Adaptive streaming of 3D content for web-based virtual reality: an open-source prototype including several metrics and strategies

Jean-Philippe Farrugia, Luc Billaud, Guillaume Lavoué

Adaptive streaming of 3D content for web-based virtual reality: an open-source prototype including several metrics and strategies

Jean-Philippe Farrugia
jean-philippe.farrugia@univ-lyon1.fr
Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
Villeurbanne, France

Luc Billaud
Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
Villeurbanne, France

Guillaume Lavoué
guillaume.lavoue@enise.ec-lyon.fr
Univ Lyon, Centrale Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, ENISE
Saint-Etienne, France

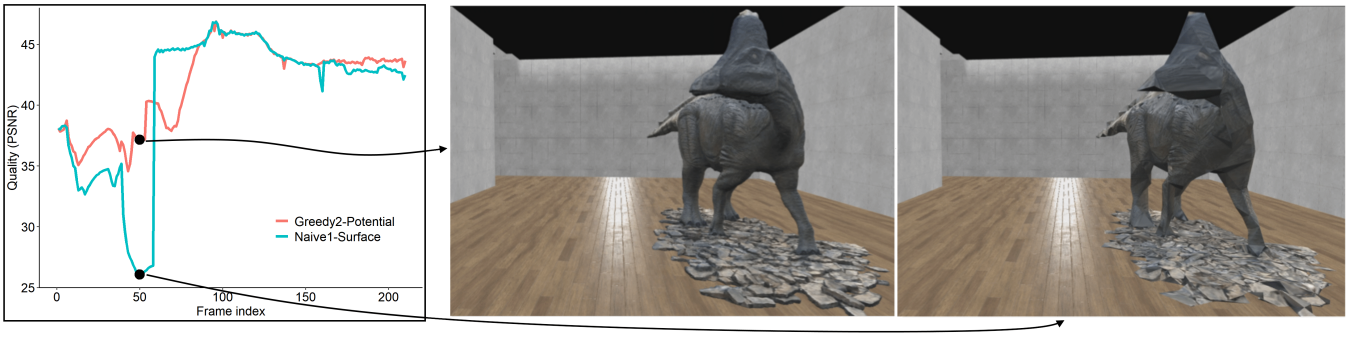

Figure 1: Our software allows us to compare different combinations of strategies and utility metrics for adaptive streaming of 3D content. This figure illustrates the performance of two combinations : *Naive1-Surface* and *Greedy2-Potential* with respect to the visual quality of the rendered streamed scenes. A frame is illustrated on the right for each of these methods.

ABSTRACT

Virtual reality is a new technology that has been developing a lot during the last decade. With autonomous head-mounted displays appearing on the market, new uses and needs have been created. The 3D content displayed by those devices can now be stored on distant servers rather than directly in the device's memory. In such networked immersive experiences, the 3D environment has to be streamed in real-time to the headset. In that context, several recent papers proposed utility metrics and selection strategies to schedule the streaming of the different objects composing the 3D environment, in order to minimize the latency and to optimize the quality of what is being visualized by the user at each moment. However, these proposed frameworks are hardly comparable since they operate on different systems and data. Therefore, we hereby propose an open-source DASH-based web framework for adaptive streaming of 3D content in a 6 Degrees of Freedom (DoFs) scenario. Our framework integrates several strategies and utility metrics from the state of the art, as well as several relevant features: 3D graphics compression, levels of details and the use of a visual quality index. We used our software to demonstrate the relevance of those tools and provide useful hints for the community for the further improvements of 3D streaming systems.

MMSys '23, June 7–10, 2023, Vancouver, BC, Canada

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**.

KEYWORDS

Virtual Reality, Streaming, Mesh compression, DASH protocol

ACM Reference Format:
Jean-Philippe Farrugia, Luc Billaud, and Guillaume Lavoué. 2023. Adaptive streaming of 3D content for web-based virtual reality: an open-source prototype including several metrics and strategies. In *Proceedings of the 14th ACM Multimedia Systems Conference (MMSys '23), June 7–10, 2023, Vancouver, BC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587819.3592555

1 INTRODUCTION

3D content is now more and more common, as technologies and techniques are developed to widespread its usage. Computer-Aided Design softwares, video games, industrial or scientific applications now all propose 3D content to their users. Hardwares like GPUs (Graphic Processing Units) allow, in conjunction with algorithms like shaders, computers to efficiently display this 3D content. Since a few years now, the rising 3D technology is Virtual Reality (VR). By immersing the users (using head-mounted displays) in a virtual 3D environment, VR proposes a seemingly realistic experience. Moreover, with the introduction of autonomous head-mounted displays, like the Oculus Quest, we got rid of constraints like a cable and a PC, as well as a restricted playground. However, this comes at the cost of performance and thus introduces new constraints on the size and complexity of the 3D world that can be displayed.

These new constraints are amplified by the fact that, in most applications, the 3D content is always more and more complex and detailed.
Another novelty is that standards such as WebGL and WebXR [16] now enables full 3D VR experience through web browsers. In those 6 DoFs (degrees of freedom) web-based VR applications, the 3D content has to be streamed from a remote web server; this may induce drops in performance and latency due to transmission, which may be both very detrimental for the user experience. In this context, several authors (e.g., [2, 11, 15]) proposed utility metrics and selection strategies to prioritize which 3D objects to fetch from the server according to the user viewport position and movement, in order to optimize at run-time the quality of what is being visualized. Those proposed algorithms use each different features (e.g., tiling, compression, levels of details), different utility metrics and different streaming strategies; it is thus difficult to compare them and evaluate the best choices in this large parameter space.
In this work, we address this problem by providing an open-source framework, which we use for the evaluation and comparison of those different features, metrics and strategies. Our contributions are the following:

- We propose and release publicly, in open source, a DASH-based web virtual reality application integrating several features, metrics and streaming strategies from the state of the art. The source code is publicly available on our GitHub repository [1].
- We provide an exhaustive evaluation and comparison of 30 combinations of metrics and strategies, under 3 different bandwidths and for 3 camera paths for a museum scene containing more than 880MB of 3D content. We explored a large set of parameters, leading to the generation of a total of 720 captured videos corresponding to user simulated navigation paths.
- We isolate and evaluate the effect of three features highly relevant for 3D streaming : 3D mesh compression, levels of details and visual quality index integration.

The remainder of this paper is organized as follows : section 2 review the related work about adaptive 3D content streaming. Section 3 describes our implemented system, while section 4 details the experimental results. Finally, in section 5, we provide conclusions and perspectives.

2 RELATED WORK

This related work section focuses on web-based adaptive streaming of 3D content, an more particularity on the algorithms introduced to optimize and prioritize the downloads in order to optimize the quality of experience. The reader can refer to [7] for a complete survey about 3D mesh compression.
With the introduction of WebGL 10 years ago, several algorithms were introduced for 3D content compression and streaming over the web [3, 4, 6, 12]. They are based on different compression strategies, compliant with a fast web decoding. Those algorithms focus on optimizing the compression rate and the streaming of one single 3D object, without any consideration on how to optimize the delivery of a whole scene composed of several objects. However, they constitute relevant tools for our problematic; the DRACO codec [3] from Google is integrated into our implemented framework.
Pioneering works related to our focus of interest were introduced only three years ago, concurrently by Forgione et al. [2], van der Hooft et al.[15] and Park et al. [11]. Those three works proposed systems and algorithms for adaptive streaming of scenes comprising multiple 3D models (either 3D point clouds or meshes). In particular, they proposed utility metrics and strategies to schedule and optimize the delivery of the 3D models (or their levels of details - LoDs) according to the network and the viewport of the user. Further methods and systems were introduced latter, based on the same or close principles [5, 13, 14]. Most of the above works intent to remain compatible with the Dynamic Adaptive Streaming over HTTP (DASH) standard, which is already widely deployed for streaming adaptive video content on the Web. These frameworks focus on 3D content represented by textured meshes [2, 5, 13] or colored point clouds [11, 14, 15]. They consider different tools, such as compression, tiling [11, 14], texture LoD [2], quality index [2], 3D object grouping [13] or manual prioritization [5].
Since these algorithms use heterogeneous data, tools and strategies then it becomes very difficult to have a clear idea of the best choices. In this work, we propose an open-source DASH compliant web-based virtual reality framework that integrates and combines a large set of tools, metrics and strategies.

3 OUR ADAPTIVE STREAMING FRAMEWORK

3.1 General description and data preparation

Our implemented framework is inspired by [2] and is compliant with DASH. It implements a scenario of an interactive visit to a virtual museum composed of high quality 3D models. On the server side, the 3D models composing the scene are simplified into 10 levels of details. Each level of details (LoD) is compressed with Google Draco and the resulting `.drc` files are stored. Each LoD is also associated with a quality score computed using the HDR-VDP2 perceptual quality metric [8]. This metric intends to predict the visual fidelity of the LoD with respect to the pristine unsimplified object. Note that HDR-VDP2 is an image-based metric; we computed its value for each 3D model by applying it to the most perceptually-relevant screenshot. On the client side, a script is charged to evaluate the current state of the scene, the user's position, rotation and trajectory and the client's bandwidth to determine the *best* objects and their levels of detail to download. Once this is determined, the corresponding files are requested, decompressed and then displayed, and the process can start again, until all the objects have their best level of detail imported.
To determine which levels of detail are the *best* and have to be imported, different combinations of strategies and metrics were implemented, taken from [2, 15] ; they are presented below.

3.2 Utility metrics from [2, 15]

Those metrics $U(m)$ reflect how much an object m is useful to the global quality of the scene. Intuitively, the closer and the bigger an object is, the more useful it is. The implemented metrics are the following :

- **Distance :** Distance between the object and the camera.
  $U_D(m) = 1/distance(camera, m)^2$

[1] https://github.com/Plateforme-VR-ENISE/AdaptiveStreaming

- **Surface :** Surface of the object divided by its squared distance to the camera.
  $U_S(m) = surface(m)/distance(camera, m)^2$
- **Visible :** Area on the screen taken by the bounding box (BB) of the object relative to the screen's area.
  $U_V(m) = screenArea(BB(m))$
- **Potential :** Area on the screen of the object if the camera were oriented towards the center of this object.
  $U_P(m) = screenArea_{CameraToward(m)}(BB(m))$
- **VisiblePotential :** It is equal to the *Visible* metric if the object is in the field of view, and is equal to *Potential* otherwise.

3.3 Strategies from Forgione *et al.* [2]

These strategies serve to select one level of detail m_l of a given object m to be imported. This m_l is the one considered as the best, i.e., the one with the best *score* as computed by the utility metrics presented above. Each strategy has a different criterion, and a different way to calculate the *score* according to the metrics above. Note at, at each selection step, candidate LoDs are those which are in the current view frustum and in a *predicted* view frustum computed using a linear predictor of the position and the rotation of the camera. The temporal horizon for this viewpoint prediction is set by default to 2 seconds in our system. The impact of this value is evaluated in the experiments.

- **Naive1 :** The score is the utility of the object multiplied by the quality index of the level of detail. This strategy is most likely to select an object that is currently visible.
- **Greedy1 :** The score is the variation of utility of an object between the moment it is imported and the temporal horizon, multiplied by the quality index of the level of detail. This strategy is most likely to select an object that will be visible after some time.
- **Optimize1 :** (called *Proposed* in the original article) The score is the integral of the utility of an object between the moment it is imported and the temporal horizon, multiplied by the quality index of the level of detail. We do it by calculating a Riemann sum. This strategy is most likely to select an object that is visible over the full period.

3.4 Strategies from Van der Hooft *et al.*[15]

These strategies select and import multiple levels of detail at a time. Here, we are filling a buffer, which corresponds to a number of bits that can be imported (downloaded and decompressed) in a certain amount of time (by default 2 seconds × the bandwidth). A loop will go through all the levels of details of all the objects and allocate bits to these LoDs until the buffer is full. The strategies loop through the levels of details differently, thus allocating the bits differently.

- **Greedy2 :** We loop through all of the levels of detail of an object before considering the next object. Objects are considered by decreasing utility.
- **Uniform2 :** We loop through a certain level of detail (e.g. level 1) for all the objects in decreasing utility order before considering the next level of detail (e.g. level 2).

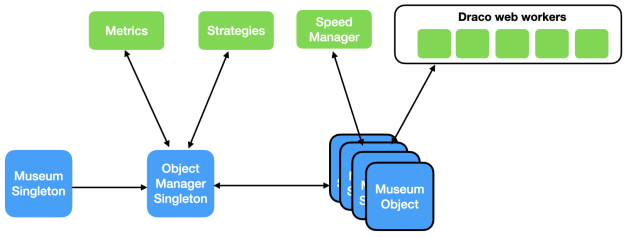

Figure 2: Architecture of our application

- **Hybrid2 :** We first loop through the visible objects in a Uniform way, then we loop through the non-visible object in a Greedy way.

As for the temporal horizon above, the impact of the buffer size (set to 2 seconds * bandwidth) is evaluated in the experiments.

3.5 Implementation

In the context of streaming use, it seemed natural to use a web-based framework for our implementation. We selected Babylon.js, which has the advantage of being WebXR compatible and allowing the use of javascript WebWorkers. WebWorkers were used to be able to run the decompression of the levels of details in parallel to the user visualization and interaction (thus preventing any freeze). Typescript, a superset of javascript, is used to allow more rigorous object programming. The architecture of our application is shown in Figure 2. It includes two singletons, an instantiated class, and service classes.

3.5.1 *Singletons.*

- The Museum Class creates the context of the application and handles its basic operations: inputs handling, scene display, asset import.
- The ObjectManager Class is the application scheduler. It manages the initialization, positioning and display of all museum objects in the scene as well as the execution of strategies.

3.5.2 *Instantiated class.*

- The MuseumObject class represents an object in the museum and is multi-instanced by the ObjectManager for each object in the scene. This class contains all the metadata (name, area, size, visual quality index) and the levels of detail generated for this object. All level of details are stored as independent .drc meshes, compressed with the Draco Framework. The MuseumObject class uses methods of the DracoCompression class, provided by BabylonJS, to decompress level of details asynchronously. In our implementation, a single pool of 5 DracoCompression web-workers is used for the whole scene.

3.5.3 *Services.*

- The SpeedManager static class gathers speed and bandwidth calculation functions.
- The Metrics class includes metric calculation functions
- The Strategies class brings together the strategy calculation functions

The overall behaviour of the application is as follows: an instance of the application is created with the Museum class. It then creates an instance of the ObjectManager singleton, which itself will dynamically create the scene's objects with the MuseumObject class. At runtime, ObjectManager applies metrics and strategies to determine the appropriate level of detail for each museum object. Each museum object instance calculates its speed and bandwidth data.

3.6 Differences with the state of the art

As stated above, our system integrates the metrics and strategies from both Forgione *et al.* [2] and Van der Hooft *et al.* [15]. However some differences remain with regards to their frameworks : Van der Hooft *et al.* [15] considered colored point clouds whereas we consider textured meshes. Our system uses multiple geometrical levels of detail (i.e., different meshes) while Forgione *et al.* [2] had only one geometrical level of detail per object but multiple texture resolutions. They use a quality index for the textures which is a simple PSNR. Our quality index (HDR-VDP2 [8]) was designed to reflect the perceived quality and demonstrated good correlation with subjective opinion on a dataset of distorted 3D models [9]. Another point is that Forgione *et al.* [2] did not use a compression algorithm. Finally, we obviously introduced differences in the algorithm implementation since research papers do not contain full implementation details.

4 EXPERIMENTS AND RESULTS

4.1 Experimental setup

To demonstrate the utility of the proposed software, we compare the different combinations of strategies and metrics presented above. We also assess the influence of the different parameters of our system. We took inspiration from Forgione *et al.*'s experimental process [2]. We created a museum scene composed of 20 high quality 3D models between 20K and 635K vertices, for a total of 880MB of uncompressed data (including 5MB for the museum building itself). Those models were taken from the Creative Common collections of Sketchfab. We created three different predefined camera movements in the scene, corresponding to realistic exploration paths. Each of these animations is 21 seconds long. We also considered three network bandwidth conditions : 2 Mbps, 4 Mbps, and 30 Mbps and two values for the temporal horizon / buffer size: 1s and 2s. Figure 3 illustrates several views of our 3D environment.

For each combination of conditions (3*animations, 6*strategies, 5*metrics, 3*bandwidths and 2*Temporal horizons/buffer size), the navigation was simulated using our adaptive streaming engine and recorded, giving us a total of 540 recordings. To objectively evaluate the quality of those resulting renderings, as in [2], we compared them to reference recordings using PSNR. Those references were *offline* versions, where each object of the scene was initially loaded with its maximum quality, thus having the best possible quality in the video. To compute the PSNR between those *test* and *reference* recordings, they were sampled each 100ms giving a total of 210 frames to compare. A particular care was taken to be sure that the frames of each pair to compare have a perfect temporal correspondence. Additional recordings were computed for different parameter settings (visual quality index on/off, levels of detail on/off) giving a total of 720 recording, i.e. 151200 test images plus 630 references ones.

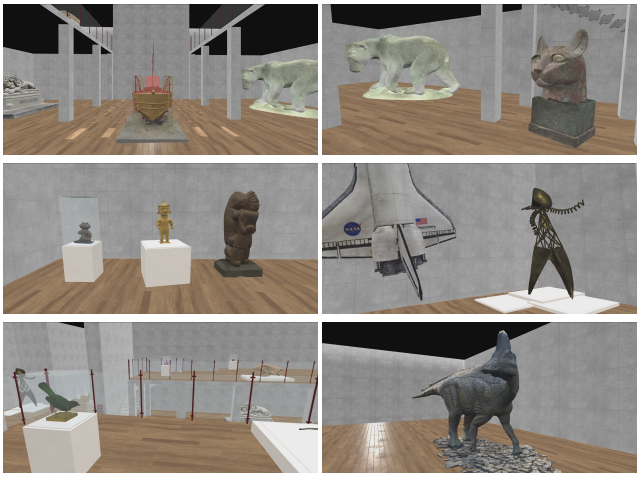

Figure 3: Several views of our virtual environment, taken from the MainRoom camera path (first row), the StatueRoom camera path (second row) and the UpperRoom camera path (third row).

4.2 Results

In this section, we evaluate and compare the performance of the different combinations of strategies and utility metrics, with regards to the bandwidth, the camera path and to the temporal parameters. We also evaluate the impact of three features of our system : the perceptual quality metric, the compression algorithm and the use of levels of details.

Best metrics and strategies according to the bandwidth

Figures 4 illustrate boxplots of PSNR values according to the metrics (top row) and according to the strategies (bottom row), for the three tested bandwidth values. For each metric (resp. strategy) each boxplot aggregates PSNR values of all frames for all the strategies (resp. metrics), and all camera paths. Note that those results are for *standard conditions* meaning that we activate the perceptual quality index, the compression and the levels of detail and the values of the temporal parameters are set to 2s.

The performances of the different utility metrics are very close to each others with a similar trend regarding the effect of bandwidth. It is interesting to observe the strong improvement between 2Mbps and 4Mbps, whereas 30Mbps brings a relatively small additional quality. The best metrics are *Surface* and *VisiblePotential*.

For the strategies, surprisingly, the simplest provide the best results : *Naive1* and *Greedy2*. One explanation is that these strategies require less operations to execute than the others. This result is a bit surprising as they are presented as bad in their respective articles. This emphasizes the need of a common public implementation, since the real-time performance of such interactive systems can be impacted by tiny details of implementation that do not necessarily appear in the research papers.

Another surprising result is the decrease in performance of the

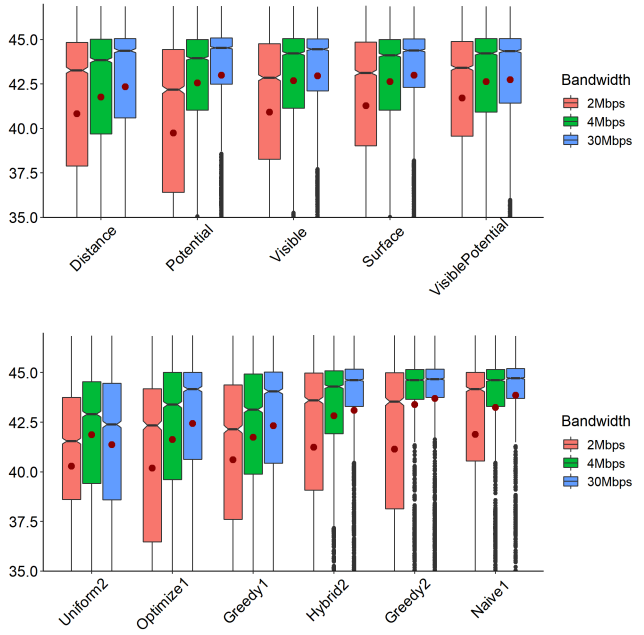

Figure 4: Boxplots of PSNR values according to the utility metrics (top) and strategies (bottom). Mean values are represented by the red dots.

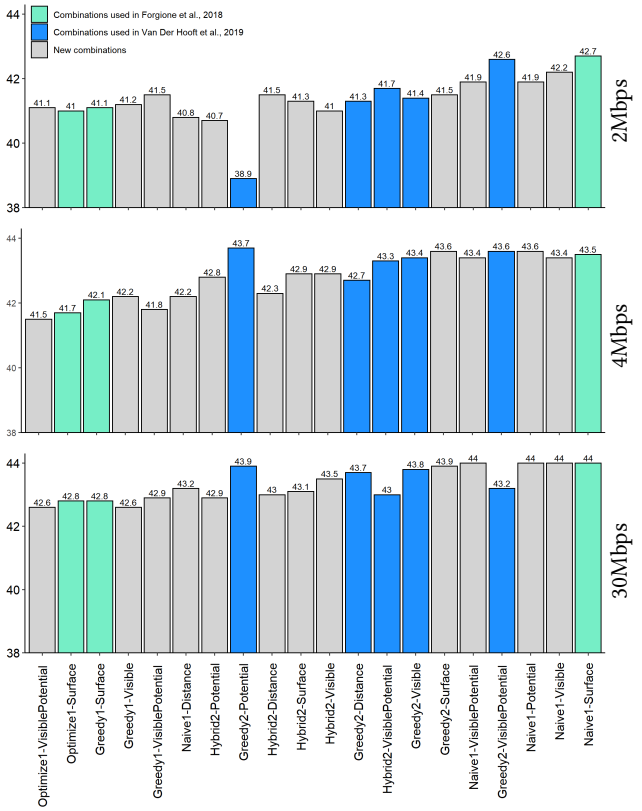

Figure 5: Performance per combination for each of the 3 bandwidths. The performance is computed as the PSNR averaged over all camera paths. Combinations are ordered by increasing average performance over the 3 bandwidths.

Uniform2 strategy when passing from 4Mbps to 30Mbps. By checking in details the results, we found that the explanation could be the time processing of the buffer filling algorithm (as described in Section 3.4). Indeed, since this algorithms loop through all objects until it fills the buffer, then a large buffer size could create latency in the downloading decisions.

Best metric/strategy combinations

Figure 5 presents the performance of each combination of metric/strategy in term of PSNR averaged over all the frames and all the animations for each bandwidth (still for the *standard conditions*). Methods are ordered by their average performance (i.e., mean over the three bandwidths). For the sake of visibility we present only the 20 best combinations (among 30). Overall, the best combinations are the following: *Naive1-Surface*, *Naive1-Visible*, *Naive1-Potential* and *Greedy2-VisiblePotential*. Moreover, none of the best strategies from Forgione's and van der Hooft's articles [2, 15], respectively *Optimize1* and *Hybrid2*, appear in our best combinations. As mentioned above, this may be probably explained by a different implementation of the algorithms and the strategies, a different test environment or even a slightly different measurement process. It is interesting to notice that the respective ranks of the combinations actually depend on the bandwidth conditions. Several methods are very stable (*Naive-Surface*), while others are significantly impacted by the decrease in bandwidth (e.g., *greedy2-potential*). It is noteworthy that for 30Mbps and 4Mbps the *greedy2-potential* method is actually the best one.

Figure 1 illustrates quality values over time and illustrations for the *Naive1-Surface* and *Greedy2-Potential* combinations. We can see how the two combinations behave differently. Some drops in quality are happening at the same time because they correspond to new objects entering the field of view. However, the spikes are not happening at the same time and not with the same amplitude, because the choices made by the combinations regarding the levels of detail to import are not the same.

Influence of the temporal horizon

The horizon and the buffer, which are parameters used by the different strategies were suspected to have a large influence on the performances of the algorithms. Van der Hooft *et al.* [15] have shown that the buffer have an impact on the number of freezes and the video quality. They showed that for higher buffer values, the prediction accuracy and then the video quality decreases, but the resilience to freezes increases. We performed our quality evaluation for two values of these parameters: 1s and 2s for the horizon and the buffer size. Statistical tests shows that the parameters have an impact on the performance, and that a buffer/horizon of 2 seconds is performing significantly better than a buffer/horizon of 1 second. Those results are confirmed by Figure 6.

Influence of the camera path The performance of the system is obviously dependent on the complexity of the camera path, its velocity and the 3D objects that are present. Figure 7 (*left*) illustrates this behavior. The camera path *UpperRoom* generates globally the worst results (this trend is general for all methods).

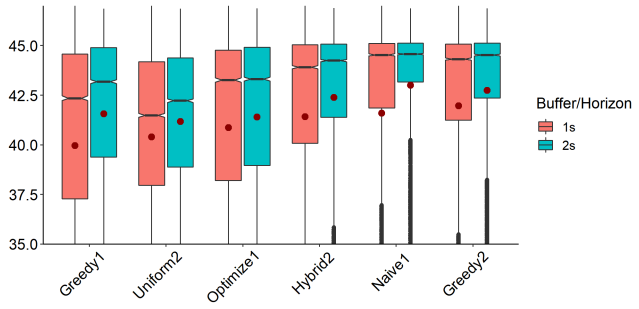

Figure 6: Boxplots of PSNR values of the strategies, according to the buffer/horizon values.

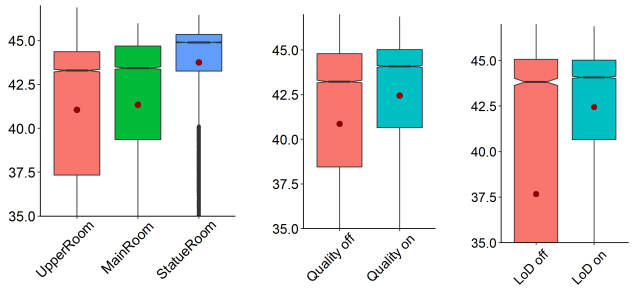

Figure 7: Boxplots of PSNR values according to the camera path (left), the use of quality metric (middle) and the use of levels of details (right).

Influence of the visual quality index

Only three strategies use a quality index. Those strategies are *Naive1*, *Greedy1* and *Optimize1*. Therefore, only these strategies are considered in the results below. In these strategies, the quality index is used to weight the utility of an object. Then, to evaluate our combinations without the use of the visual quality index, we simply replace it by 1. Figure 7 (*center*) clearly shows that the visual quality index (HDR-VDP2 [8]) brings a significant improvement of the PSNR (confirmed by statistical test).

Influence of the levels of detail

Figure 7 (*right*) clearly shows that the levels of detail have a significant impact on the PSNR (confirmed by statistical test). This result was expected as simplified meshes are always closer to the reference than the background.

Influence of compression

Table 1 shows a comparison of the download time of some `.obj` files and the import time of their `.drc` counterparts (i.e., compressed by Draco [3]). In this table, the calculations are made with a high bandwidth (30 Mbps) and an average decompression speed of 8 Mbps (the average decompression speed in our setup). In those conditions, the compressed files require around 90% less time to be imported than the uncompressed ones. This data clearly indicates that compression with Google Draco is very advantageous and reduces the import time of meshes in almost every situation. Moreover, `.obj` files might be long to parse and most OBJ Loaders are not meant to be used during runtime and are not multi-threaded, while Babylon.js's *DracoCompression* object is.

Name	Uncompressed (.obj)		Compressed (.drc)		Δ
	Size (KB)	Time (s)	Size (KB)	Time (s)	
Dinosaur	78185	20.85	794	1.01	95%
Lion	45492	12.13	794	1.01	92%
Lizard	4402	1.17	84	0.11	91%
Orchid	2221	0.59	52	0.07	89%
Shuttle	10583	2.82	250	0.32	89%

Table 1: Comparison of the import time for uncompressed and compressed meshes. Bandwidth = 30Mbps, decompression speed = 8 Mbps. Δ expresses the gain in the time provided by the compression.

5 CONCLUSION AND DISCUSSION

In this work, we present our open-source framework for streaming 3D graphics in 6DoFs scenarios. We used our framework to evaluate 6 strategies and 5 metrics from the state of the art in a virtual museum scenario. We determined the best combinations for several bandwidths and demonstrated the impact of several tools and parameter values. Those results constitute first steps toward the full understanding and development of optimized streaming systems for 6 DoFs virtual experiences. Our results and the differences we observed regarding state-of-the-art conclusions, emphasized the fact that streaming systems are difficult to compare since many details of implementation may influence the results. The release of our source code is an attempt to provide the community with a common basis for comparison.

Beyond the considerations raised above, our implemented system owns several limitations that should be improved in future work. First, levels of details should be considered also for texture, and not only for meshes. For instance, the progressive algorithm from Caillaud et al. [1] proposes a bit-allocation framework which multiplexes mesh and texture levels of details to optimize the visual quality for a given bit budget. Another way of improvement concerns the visual quality index: we used an existing image-based metric calibrated for natural image distortions; however, very recently, several authors introduced metrics specifically suited for visual quality assessment of 3D graphics [9, 10]. Finally, our performance metric is the per-frame PSNR ; this metric is obviously not directly related to the *quality of experience* which is our main interest. Note that we also computed SSIM values on top of PSNR ; however reported values were not significant since this perceptual metric is not suited for large scene renderings where only localized distortions occur. The only way to measure and evaluate the quality of experience would be to conduct subjective user tests. Such user studies would also be a mean to assess if the quality of user experience is correlated or not with our evaluation measure. We believe that our publicly-released platform may constitute an excellent basis for that, as well as for exploring further factors (e.g., including visual attention models, and so on).

6 ACKNOWLEDGMENTS

This work was partly supported by French National Research Agency as part of ANR-PISCo project (ANR-17-CE33-0005), and by Computer Science Department of IUT Lyon 1, Bourg en Bresse, France.

REFERENCES

[1] F. Caillaud, V. Vidal, F. Dupont, and G. Lavoué. 2016. Progressive compression of arbitrary textured meshes. *Computer Graphics Forum* 35, 7 (2016). https://doi.org/10.1111/cgf.13044

[2] Thomas Forgione, Axel Carlier, Géraldine Morin, Wei Tsang Ooi, Vincent Charvillat, and Praveen Kumar Yadav. 2018. DASH for 3D Networked Virtual Environment. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) *(MM '18)*. Association for Computing Machinery, New York, NY, USA, 1910–1918. https://doi.org/10.1145/3240508.3240701

[3] Google. [n. d.]. Draco. https://github.com/google/draco.

[4] G Lavoué, L Chevalier, and F Dupont. 2013. Streaming Compressed 3D Data on the Web using JavaScript and WebGL. *ACM Web3D* (2013). http://liris.cnrs.fr/guillaume.lavoue/travaux/conference/WEB3D2013.pdf

[5] Hendrik Lievens, Maarten Wijnants, Mike Vandersanden, Peter Quax, and Wim Lamotte. [n. d.]. Adaptive Web-Based VR Streaming of Multi-LoD 3D Scenes via Author-Provided Relevance Scores. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (Lisbon, Portugal, 2021-03). 488–489.

[6] Max Limper, Maik Thöner, Johannes Behr, and Dieter W. Fellner. 2014. SRC - a streamable format for generalized web-based 3D data transmission. *Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies - Web3D '14* (2014), 35–43. http://dl.acm.org/citation.cfm?id=2628588.2628589

[7] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot. 2015. 3D mesh compression: Survey, comparisons, and emerging trends. *Comput. Surveys* 47, 3 (2015). https://doi.org/10.1145/2693443

[8] Rafal Mantiuk, Kil Joong Kim, Allan G Rempel, and Wolfgang Heidrich. 2011. HDR-VDP-2 : A calibrated visual metric for visibility and quality predictions in all luminance conditions. *ACM Siggraph* (2011).

[9] Yana Nehmé, Florent Dupont, Jean-Philippe Farrugia, Patrick Le Callet, and Guillaume Lavoué. 2021. Visual Quality of 3D Meshes With Diffuse Colors in Virtual Reality: Subjective and Objective Evaluation. *IEEE Transactions on Visualization and Computer Graphics* 27, 3 (2021), 2202–2219. https://doi.org/10.1109/TVCG.2020.3036153

[10] Yana Nehmé, Florent Dupont, Jean-Philippe Farrugia, Patrick Le Callet, and Guillaume Lavoué. 2022. Textured Mesh Quality Assessment: Large-Scale Dataset and Deep Learning-based Quality metric. *arXiv preprint arXiv:2202.02397* (2022).

[11] Jounsup Park, Philip A. Chou, and Jenq-Neng Hwang. [n. d.]. Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 ([n. d.]), 149–162. https://doi.org/10.1109/JETCAS.2019.2898622

[12] Federico Ponchio and Matteo Dellepiane. 2016. Multiresolution and fast decompression for optimal web-based rendering. *Graphical Models* 88 (2016), 1–11. https://doi.org/10.1016/j.gmod.2016.09.002

[13] Carter Slocum, Jingwen Huang, and Jiasi Chen. [n. d.]. VIA: Visibility-aware Web-based Virtual Reality. In *The 26th International Conference on 3D Web Technology* (Pisa Italy, 2021-11-08). ACM, 1–9. https://doi.org/10.1145/3485444.3487641

[14] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. [n. d.]. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle WA USA, 2020-10-12). 3669–3677.

[15] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 2405–2413. https://doi.org/10.1145/3343031.3350917

[16] W3C. [n. d.]. WebXR Device API. https://www.w3.org/TR/webxr/.