Trails of Performance: A Study of Real Time Optimization Techniques in Heterogeneous Natural Environments for VR

Lucas Rangel da Cruz[1], **Nicole Cristina Vieira dos Santos**[1], **Luiz André de Souza**[1]
Andressa dos Santos Oliveira[1], **Marcelo Arêas Rodrigues da Silva**[1]
Joel André Ferreira dos Santos[1]

[1] CEFET/RJ
Rio de Janeiro – Brazil

{lucas.rangel, nicole.vieira}@aluno.cefet-rj.br

{luiz.souza, andressa.oliveira}@aluno.cefet-rj.br

marcelo.rodrigues@cefet-rj.br, jsantos@eic.cefet-rj.br

Abstract. Introduction: *Virtual reality has gained widespread adoption across training, education, and other fields, yet hardware limitations, especially in lower-end standalone devices, pose challenges for accessibility and quality. Moreover, optimization is still a slow and experimental process, often at odds with the fast pace of research, especially for small teams. Thus, effective software optimization is critical for delivering a smooth and enjoyable experience for users and developers.* ***Objectives:*** *This study evaluates various optimization techniques and proposes a hybrid approach to address limitations observed in a target VR scene.* ***Methodology:*** *Performance metrics were collected on a VR device at key points in a test scene for each technique. Results were compared to identify the most effective methods, which were then combined into a hybrid technique and further assessed.* ***Results:*** *The hybrid technique increased performance by 26%, achieving an average frame time reduction of around 8 ms compared to individual techniques, with the second-best technique, a Level of Detail system, yielding a reduction of around 4 ms on average. However, the hybrid technique's broader applicability outside the test scene requires further experimentation.*
Keywords *Optimization, Virtual Reality, Unity, Occlusion, LOD, Comparison.*

1. Introduction

Virtual Reality (VR) has emerged as an important technology with applications across diverse fields [Anthes et al. 2016], enabling the creation of interactive and immersive experiences comparable to those in the physical world. Its use spans simulated surgical training [Ntakakis et al. 2023], engineering education [Wang et al. 2018], and cognitive rehabilitation [Saleh et al. 2025], offering opportunities to experience complex scenarios with minimal risk.

Despite these advances, head mounted displays (HMD) continue to face significant hardware constraints [Anthes et al. 2016, Sanjay e Salanke 2017], primarily due to the head-mounted nature of the displays. These limitations affect user experience by reducing immersion and comfort and impose challenges on developers who must address them [Hamad e Jia 2022, Nusrat et al. 2021, Xiao et al. 2010].

Therefore, creating immersive environments for HMDs requires careful balancing of visual fidelity and real-time performance. This trade-off becomes particularly more complex in scenarios involving large numbers of objects, varying spatial scales, and transparent materials, such as in natural environments.

While ongoing improvements in mobile graphics hardware may eventually mitigate some of these issues, current limitations remain a major obstacle for VR development. It restricts the range of target platforms, increases production costs, and, in some cases, hinders broader adoption of VR technologies.

The field of real-time 3D graphics provides a wide range of optimization techniques, many of which are established in video game development, to achieve performance across different hardwares. However, these methods span from low-level to high-level approaches, involving distinct trade-offs. As a result, selecting and applying the most appropriate techniques often requires preliminary analysis and iterative testing, making this optimization process both slow and experimental.

In this context, the present work evaluates a set of selected optimization techniques applied to a complex natural environment, from an ongoing parallel research. Furthermore, it introduces a method for combining these techniques into a parametric approach, and verifies its effectiveness against the individual methods, demonstrating average improvements in performance on the target scene.

2. Background

A three-dimensional environment, as defined by [Shirley e Marschner 2009, Scratchapixel 2024a], is a space where sets of geometric models, called polygons, are represented by a series of points known as vertices. These vertices contain their spatial coordinates in three dimensions and include additional data relevant to describing other characteristics of the represented polygon, such as the color of that point. A group of vertices defines the surface of a polygon, a mesh of triangles. Triangles are the basic elements of rendering, as they can efficiently approximate complex meshes. They are constructed using three vertices.

During the rendering process of a 3D scene, Graphics Processing Unity (GPU) may face challenges related to the efficiency of the graphics pipeline. Among these challenges, overdraw and overshading stand out, as they can significantly impact rendering performance and the consumption of computational resources.

Overdraw occurs when the same pixel is drawn multiple times within a single frame due to the overlapping of multiple models. This results in a waste of computational power as the GPU processes information that will be replaced by other visual elements before the final display. Overshading occurs when the shading calculation of a pixel is performed excessively, either due to the complexity of the shaders used or the application of effects in areas that do not contribute to visual quality. It can create a performance bottleneck, as it demands significant computational power, especially when complex operations like dynamic lighting and reflections are involved [Scratchapixel 2024b, Shirley e Marschner 2009, Unity Technologies 2024].

3. Related Work

Koskela presents a framework of automated processes that deliver 3D models to low-performance mobile devices through the internet, discussing the inclusion of optimization techniques as part of the solution [Koskela e Vatjus-Anttila 2015]. Galindo reviews optimization techniques and explores the workflow in the development of optimized models [Galindo Jr 2018]. Both demonstrate opportunities for the application of optimization techniques in real-world scenarios, reinforcing the rationale for further studies, although they do not present experimental data.

On the other hand, [Nusrat et al. 2021] offers an empirical study on performance optimizations used by developers around the world, providing valuable information on the standard use of optimization techniques for VR. Furthermore, in a recent effort, [Oksanen 2022] presents a comparative evaluation of optimization in a 3D scenario that simulates the interior of a building, which is the study most closely related to the problem addressed in this work.

The above studies provide a solid foundation for future research. It is possible to observe an opportunity to conduct exploratory studies in more heterogeneous scenarios such as natural and open areas. Moreover, the present work stands out by evaluating and constructing a method based on the strengths of each presented technique. With this in mind, the present work will adopt and carry out a comprehensive exploratory analysis of the scenario and the effects of selected optimization techniques.

4. Optimization Techniques

A major challenge on VR development, as noted by [Koskela e Vatjus-Anttila 2015], lies in balancing draw calls and vertex processing on performance-constrained devices, making the understanding of the available techniques and their characteristics essential.

VR software optimization can be tailored to distinct steps of the rendering pipeline, and can be effective in re-balancing the load. Furthermore, knowledge of the balancing dynamics is relevant to better target and detect existing bottlenecks.

Draw calls are CPU instructions sent to the GPU to render elements. As shown in Figure 1, each call involves data preparation and transmission, which can bottleneck the CPU if excessive. Similarly, high vertex counts increase GPU workload, delaying frame completion and reducing performance [Luebke e Humphreys 2007].

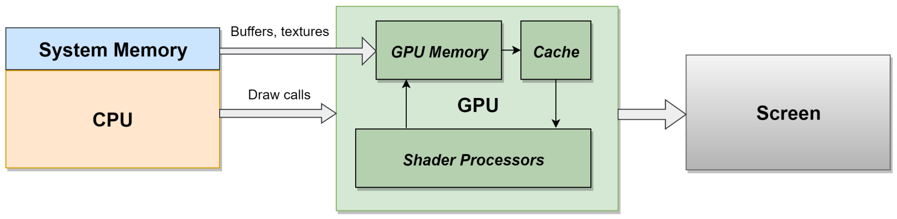
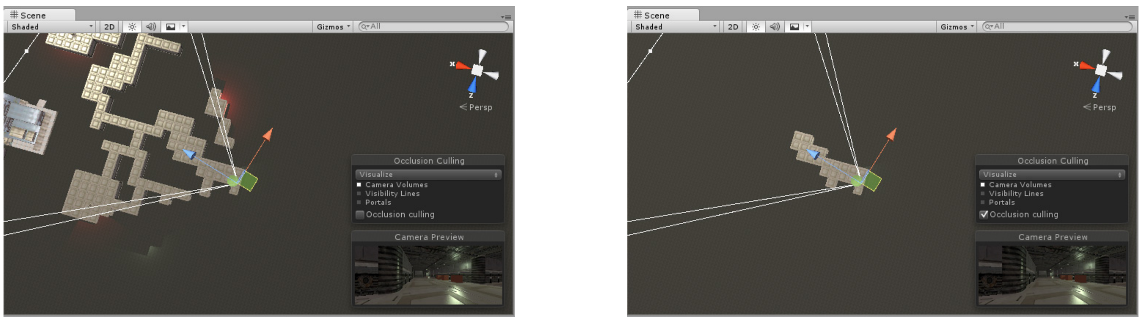


Figure 1. Connection between CPU and GPU.
Source: `http://fragmentbuffer.com/`

To mitigate this, batching techniques reduce draw calls by grouping elements with some degree of similarity. Static Batching combines non-moving models sharing a material into a single draw call and is ideal for static scenes [Unity Technologies 2024]. Other techniques, such as GPU Instancing, combine multiple instances of the same mesh after a draw call, benefiting scenes with many identical objects [Unity Technologies 2024].

Visibility optimizations prevent unnecessary rendering by first calculating the visible objects before rendering. Frustum Culling skips objects outside the camera's view (Figure 2a) [Unity Technologies 2024], while Occlusion Culling ignores objects hidden or occluded by other objects (Figure 2b) [Opsenica 2020]. Without these, all models would be drawn by the GPU, degrading performance.



(a) Frustum Culling (b) Occlusion Culling

Figure 2. Examples of culling methods.
Source: `https://docs.unity3d.com/`

However, occlusion culling struggles with transparent materials like foliage, treating them as occluders. A solution for it can be using a proxy of the scene, i.e. a simplified opaque replica of the scene and then pre-calculating visibility.

For scenes with high graphical detail, quality reduction techniques dynamically lower model and texture complexity to balance CPU/GPU workloads. Level of Detail systems (LOD) adjust model complexity based on distance from the camera, replacing distant objects with lower-poly versions [Radivojevic 2024] (Figure 3).

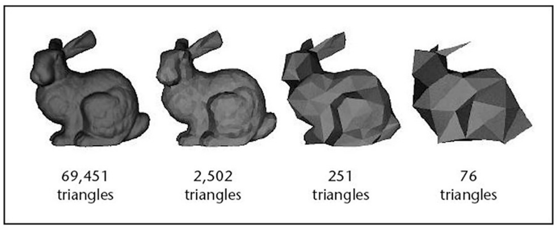


Figure 3. Example of a LOD system.
Source: `https://3dstudio.co/pt/`

5. Methodology

In this study, Unity Engine was used to build VR applications, apply the optimization techniques, and collect data. Unity is a platform that allows developers to program shaders and adjust different stages of the rendering pipeline according to the demands of each project [Unity Technologies 2024]. Understanding the structure of this pipeline can be crucial for optimizing game performance, especially on resource-constrained platforms such as mobile devices and HMDs.

5.1. Use case application

The application "Aventura na Trilha Lagoa da Mata", ATLM for short, is an educational virtual experience project used as a use case. ATLM is still in development for HMDs and is an adaptation of the educational board game of the same name. In the game, the Lagoa da Mata trail is presented to the player along a board path, featuring playful elements and interesting facts about the region, to raise awareness of the importance and richness of its fauna and flora. This trail exists in the real world and is home to a wide variety of animal and plant life, making it an engaging environment to replicate in 3D. Figure 4 presents ATLM's 3D environment [da Silva et al. 2021, Soares et al. 2021].

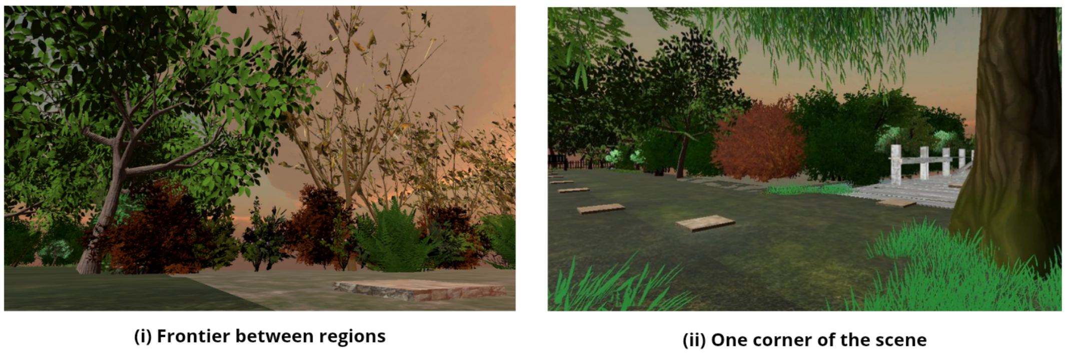


Figure 4. Types of vegetation across the virtual trail. Figure (i) shows a frontier between two regions. Figure (ii) shows another region, across the bridge.

ATLM was chosen for this study due to its features: (i) the use of several different models for plants/trees, (ii) the presence of objects created by developers using photogrammetry, (iii) the size of the scene, (iv) the movement and animation of objects, and (v) the variety of illumination throughout the scene.

5.2. Procedure

This study evaluates the ATLM environment, on Unity version 2021.3.11.f1, to identify performance bottlenecks and explore optimization strategies suitable for VR platforms, with a particular focus on the Meta Quest family. Given the relatively straightforward process of compiling for multiple Quest devices, we selected the first-generation Meta Quest as the target hardware.

To assess performance under realistic usage scenarios, we systematically sampled the scene by manually selecting discrete points across regions of varying complexity. These sampling points were distributed to capture representative areas, including vegetation zones, transitional frontiers between regions, and open spaces.

At each point, we performed a full 360-degree rotation along the Y-axis, recording performance metrics and region identifiers at chosen degree intervals. This approach enabled the collection of a dataset reflecting diverse viewing angles and environmental complexities. Figure 5 highlights the multiple regions dividing the environment.

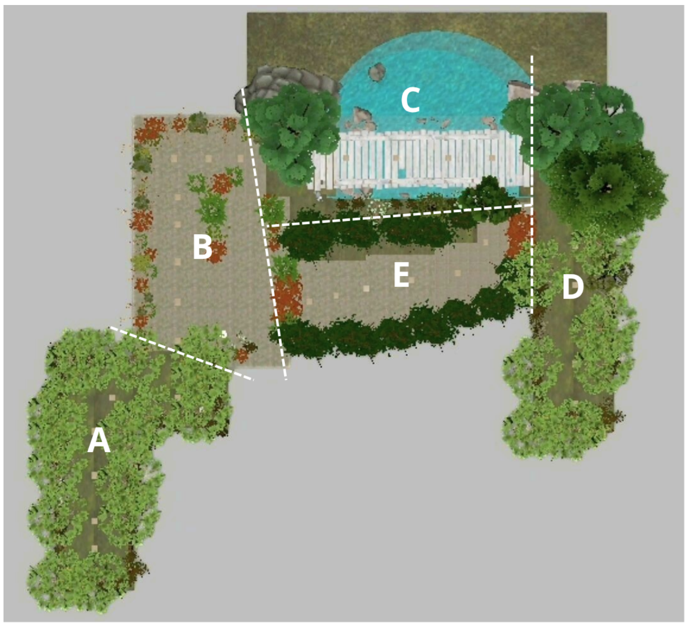


Figure 5. Frontiers of the trail regions, separating vegetation types. Regions A, B, C and D make the main route, while region E represents the extra route.

A wide array of optimization techniques is available for virtual environments, each addressing different aspects of rendering performance [Koskela e Vatjus-Anttila 2015]. However, not all are suitable for the constraints of VR development or compatible with the Unity-based pipeline of ATLM.

We curated a discrete set of techniques adhering to the following criteria: (i) minimal impact on existing scene content, (ii) no requirement for engine-level modifications, and (iii) proven compatibility with VR rendering and Unity version. These constraints led us to search for content-transparent kinds of techniques; these should avoid rework and seamlessly fit in a given pipeline when applied to the target scene.

Each selected optimization was applied individually to the target scene. Performance data were then collected on the Meta Quest to evaluate the impact of each technique in isolation. Subsequently, results were analyzed to identify the most effective subset of techniques for ATLM's environment.

Finally, we combined the parameters of the best-performing techniques into a unified configuration, aiming to achieve comparable or improved performance relative to the individual applications. To ensure a fair comparison, special attention is needed to parameter tuning and its influence on rendering outcomes.

6. Performance Analysis

Figure 6 shows a scatter plot of all models in the scene, highlighting triangle density with darker colors and clustering with contrasting tones.

An initial analysis suggests that poor performance may stem from the scene's complexity: approximately 1.6 million vertices and 1.2 million triangles distributed across 1,380 models. A secondary review of shaders revealed extensive use of translucent materials for vegetation, which dominates the scene and is dispersed in heterogeneous regions.

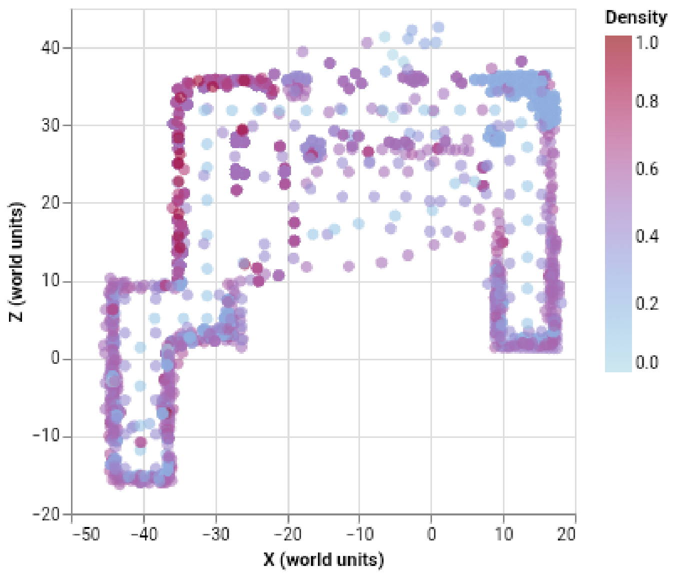


Figure 6. Distribution of models (triangles) in the scene.

This complexity suggests two bottlenecks: CPU strain from high triangle counts and GPU overload caused by overdraw from semi-transparent foliage. To address this, four techniques were implemented: Static Batching, GPU Instancing, LOD, and a custom Occlusion Culling with Proxy, adapted to handle dense, semi-transparent vegetation.

Five experiments were conducted: the unoptimized scene and four variants each applying a single optimization. For each of the 5 scene variations, 250 samples were collected across 19 capture points, yielding 23,750 records on Meta Quest.

Table 1 compares average render times across variants. Median render time in the unoptimized scene was ≈38 ms, reduced below 30 ms by Static Batching, LOD, and Occlusion Culling. GPU Instancing, however, degraded performance, likely due to increased GPU workload causing a bottleneck.

Table 1. Summarized performance overview of the hybrid technique.

Technique	**Time (ms)**	**Variation (std)**
Baseline (Unoptimized)	35,61	12,78
Static Batching	34,76	12,59
GPU Instancing	39,41	11,08
LOD	30,35*	8,86*
Occlusion Culling (w/ Proxy)	33,38	10,31
Average (all)	34,702	14,084

These findings confirm the necessity and effectiveness of scene optimization, with LOD showing particular promise for improving stability and performance on standalone devices.

7. Proposed Ensemble Method

The ensemble method developed in this study consists of assigning different optimization techniques to specific groups of scene elements, based on the characteristics of those elements and how they impact performance.

First, scene objects were grouped according to their size and function in the environment. Larger elements were treated as potential occluders, while smaller objects were considered likely to be occluded. Given that vegetation posed the most significant performance challenges, all vegetation assets were added to the LOD system, and given secondary simplified representations that always face the camera while requiring fewer resources to render.

Static batching was applied automatically to all other elements in the scene that fit the static criteria. Additionally, small vegetation assets were configured for GPU instancing, which allowed multiple instances of the same model to be rendered more efficiently in a single draw call.

The segmentation of elements was performed directly within the Unity project using hierarchy queries. These queries relied on consistent naming conventions to identify and separate sets of models by type and size. To ensure that this system remains functional, a naming convention must be applied and maintained throughout development.

In the end, the ensemble method parameters that resulted were manually tuned through iterative testing to balance performance and visual quality and minimize the occurrence of aggressive LOD transitions. At the end of the object segmentation process, the groupings were formalized as follows in Table 2.

Table 2. Final object segmentation.

Object Type	LOD	GPU Instancing	Occlusion Culling	Static Batching
Large Vegetation	Yes	No	No	No
Small Vegetation	Yes	Yes	Yes	No
Complex Meshes	Yes	No	Yes	Yes
Others	No	No	Yes	Yes

Although parameter tuning was conducted manually due to time constraints, the process could be automated or semi-automated in future iterations to improve consistency and efficiency. Finally, the final result was a visually stable scene with no significant popping artifacts and strong performance on the original Meta Quest device.

8. Results

Figure 7 provides a side-by-side comparison of the performance of the methods described in this work.

The observed performance improvements align with expectations, as the methods aim to reduce unnecessary draw calls. Notably, the LOD system yielded greater performance gains compared to the occlusion culling system. This outcome likely reflects

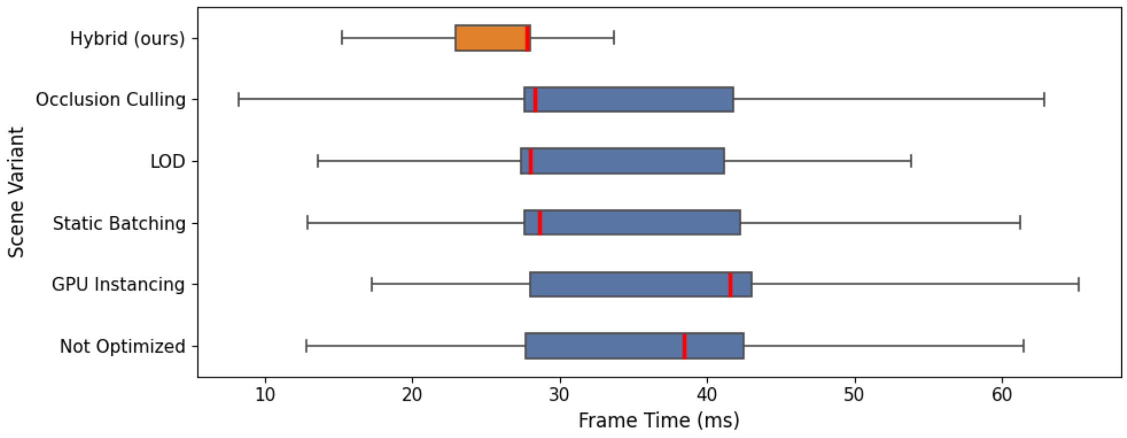


Figure 7. Box Plot for the time of each variant of the scene on Meta Quest.

characteristics of the analyzed scene: although it contains some denser regions, it still appears to be reasonably open, with relatively few objects acting as occluders.

An additional finding is that GPU instancing led to a significant performance degradation over all experiments. This result suggests that the GPU may already be operating near its processing limit. While GPU instancing reduces CPU overhead, it can inadvertently increase the workload on the GPU, exceeding its processing capabilities in this specific context.

Finally, the ensemble method delivers the best performance among the techniques. In addition to having the lowest average rendering time, it also maintains 50% of the frame-time values in a narrower range than the other methods, indicating greater overall smoothness. Table 3 presents the full comparison against the previous results.

Table 3. Performance results across the chosen techniques.

Technique	Average (ms)	Delta (ms)	Performance (%)	Variation (std)
Unoptimized scene	35,61	-9,31	+26,14%	12,78
Best of the set (LOD)	30,35	-4,05	+13,34%	8,86
Average (all)	34,702	-8,402	+24,21%	14,084
Hybrid (ours)	26,30	N/A	N/A	7,24

The ensemble method demonstrated around 26% performance improvement over the unoptimized scene and a 24% improvement compared to the other optimization methods on average, on Meta Quest hardware. However, none of the methods analyzed reached the ideal performance level of about 14 to 16 ms [Klein et al. 2024], suggesting that the hardware may be operating at its upper limit.

The original Meta Quest, due to its hardware limitations, showed to be a challenging playground for this study. However, the method suggests that the ensemble method, because of its parameter-driven approach, can be replicated on more advanced devices, such as Meta Quest 2.

In fact, the Meta Quest 2 ended up being the primary platform for the application, and undergoing tests showed that it also benefits from the technique. The Quest 2 successfully achieves about 16 ms on render time. Figure 8 shows the comparison of the hybrid technique applied in both generations of HMD to highlight the adaptability across these two devices.

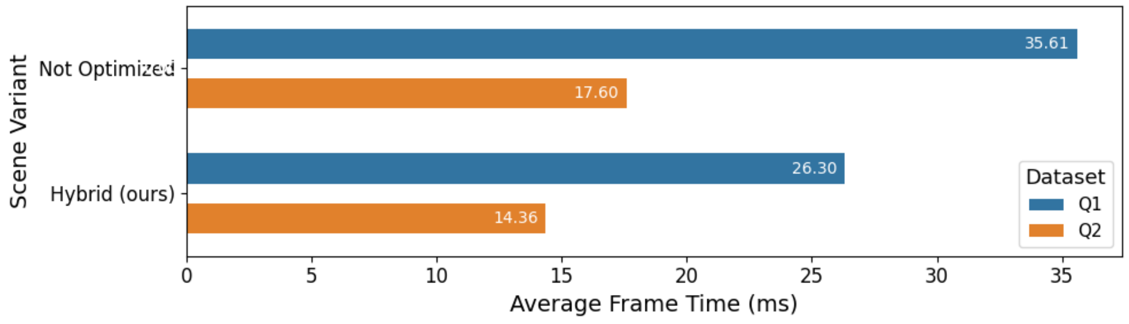


Figure 8. Comparison between the Quest 1 and Quest 2.

9. Future Work

Although widely used, traditional optimization methods rely on a margin of computational capacity, favoring devices with reasonable hardware [Koskela e Vatjus-Anttila 2015]. In the original Meta Quest, factors involving overdraw and CPU/GPU load are challenges that go beyond the presented solutions. Many developers face similar difficulties, often resorting to solutions that impact quality.

Given that the presented techniques were insufficient to achieve the ideal rendering times, future studies could explore more advanced alternatives, such as using machine learning algorithms to tune the LOD and choose occluders, or including previously unavailable techniques to the ensemble set.

Moreover, future studies on the hybrid technique should explore adaptive approaches to the parameter adjustment. Researching the possibility of capturing real-time data to provide the technique with better parameters with the goal of decreasing outliers, for example.

Additionally, fair comparison of the method is yet to be solved, since all methods discussed in this work are reasonably adjusted to the target scene. In this regard, proper comparison of the hybrid technique shall go through rigorous testing on different challenging 3D scenes of the same kind, which is suitable for future research as well.

The results of this study reinforce the potential for further advancements in VR optimization and highlight the need for ongoing efforts as VR hardware continues to evolve. Future work will focus on refining these techniques, exploring new method combinations and adaptiveness, to build low-performance friendly VR solutions and ensure better immersive experiences on standalone devices.

Acknowledgments

The authors would like to express their sincere gratitude to Professor Leonardo de Bem Lignani for invaluable support throughout this work. The appreciation is also extended to the Limnology Group of UFRJ (Brazil) for their assistance on ATLM's development.

References

Anthes, C., García-Hernández, R. J., Wiedemann, M., e Kranzlmüller, D. (2016). State of the art of virtual reality technology. In *2016 IEEE Aerospace Conference*, pages 1–19.

da Silva, C., Figueiredo, T., Rodrigues, B., Bozelli, R., e Freire, L. (2021). Em busca de uma Ética do viver: Narrativas de professores e educadores ambientais em experiÊncias didÁticas em uma trilha interpretativa na amazÔnia. *Tecné Episteme y Didaxis TED*.

Galindo Jr, G. G. T. (2018). Um conjunto de métodos de otimização e boas práticas para jogos digitais 3d na portabilidade para plataformas menos potentes. Final Project Thesis.

Hamad, A. e Jia, B. (2022). How virtual reality technology has changed our lives: An overview of the current and potential applications and limitations. *International Journal of Environmental Research and Public Health*, 19(18).

Klein, D., Spjut, J., Boudaoud, B., e Kim, J. (2024). Variable frame timing affects perception of smoothness in first-person gaming. In *2024 IEEE Conference on Games (CoG)*, pages 1–8.

Koskela, T. e Vatjus-Anttila, J. (2015). Optimization techniques for 3d graphics deployment on mobile devices. *3D Research*, 6.

Luebke, D. P. e Humphreys, G. (2007). How gpus work. *Computer*, 40.

Ntakakis, G., Plomariti, C., Frantzidis, C., Antoniou, P. E., Bamidis, P. D., e Tsoulfas, G. (2023). Exploring the use of virtual reality in surgical education. *World Journal of Transplantation*, 13(2):36–43.

Nusrat, F., Hassan, F., Zhong, H., e Wang, X. (2021). How developers optimize virtual reality applications: A study of optimization commits in open source unity projects. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 473–485.

Oksanen, M. (2022). 3d interior environment optimization for vr. Final Project Thesis.

Opsenica, B. (2020). Frustum culling. Accessed on: 17 jan. 2025. Accessed on: `http://www.lighthouse3d.com/tutorials/view-frustum-culling/`.

Radivojevic, F. (2024). Level of detail (lod) in games: Enhancing performance and visuals. Accessed on: 17 jan. 2025. Accessed on: `https://www.renderhub.com/blog/level-of-detail-lod-in-games-enhancing-performance-and-visuals`.

Saleh, N., Salaheldin, A. M., Badawi, M., e El-Bialy, A. (2025). Rehabilitative game-based system for enhancing physical and cognitive abilities of neurological disorders. *Cognitive Neurodynamics*, 19(1). Cited by: 0; All Open Access, Hybrid Gold Open Access.

Sanjay, S. e Salanke, N. S. G. R. (2017). Recent trends and challenges in virtual reality. *International Journal of Computer Applications*, 166:4–6.

Scratchapixel (2024a). Introduction to polygon meshes. Accessed on: 17 jan. 2025. Accessed on: `https://www.scratchapixel.com/`

`lessons/3d-basic-rendering/introduction-polygon-mesh/introduction.html`.

Scratchapixel (2024b). What is shading: Light-matter interaction. Accessed on: 17 jan. 2025. Accessed on: `https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/what-is-shading-light-matter-interaction.html`.

Shirley, P. e Marschner, S. (2009). *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., USA, 3rd edition.

Soares, B. R., Barbosa, A., Silva, C. d., Lopes, V., Bozelli, R., e Santos, L. d. (2021). Jogo "vida na lagoa da mata": Entrelaçando ensino de ciências e divulgação científica na floresta nacional de carajás (pa). *Anais XIII Encontro Nacional de Pesquisa em Educação em Ciências*.

Unity Technologies (2024). *Unity Manual (2022.3 LTS)*. Accessed on: 05 fev. 2025.

Wang, P., Wu, P., Wang, J., Chi, H.-L., e Wang, X. (2018). A critical review of the use of virtual reality in construction engineering education and training. *International Journal of Environmental Research and Public Health*, 15(6). Cited by: 589; All Open Access, Gold Open Access, Green Open Access.

Xiao, Y., Bhaumik, R., Yang, Z., Siekkinen, M., Savolainen, P., e Ylä-Jääski, A. (2010). A system-level model for runtime power estimation on mobile devices. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 27–34.