TU Informatics

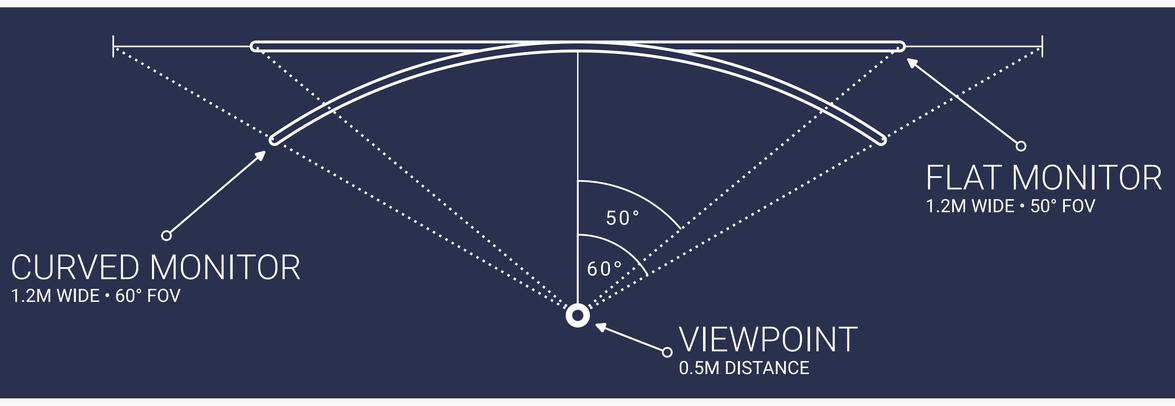
Real-Time Distortion Correction Methods for Curved Monitors

Wolfgang Rumpler, BSc
Visual Computing

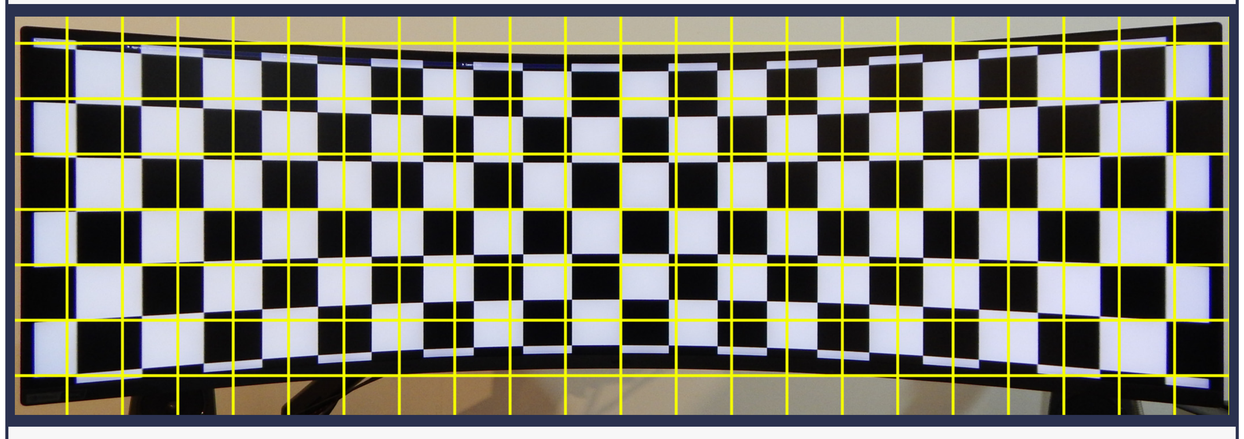
TU Wien Informatics
Institute of Visual Computing & Human-Centered Technology
Research Unit of Computer Graphics
Supervisor: Univ. Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Projektass. Dipl.-Ing. Johannes Unterguggenberger, BSc

1. Introduction & Problem Statement

In recent years, ultra-wide curved computer monitors have become an option for consumers. An advantage of such monitors is that they cover a wider horizontal field of view compared to a flat monitor with the same surface dimensions.



However, most real-time applications, such as games and simulations, expect a flat display. As a result, rendered images appear distorted and unnatural when viewed on a curved monitor. In the image below, the checkerboard displayed on the screen does not match the expected superimposed rectangular grid.



The geometry of the monitor and the user's viewpoint have to be considered during rendering to generate accurate images that preserve the impression of straight lines on curved monitors.

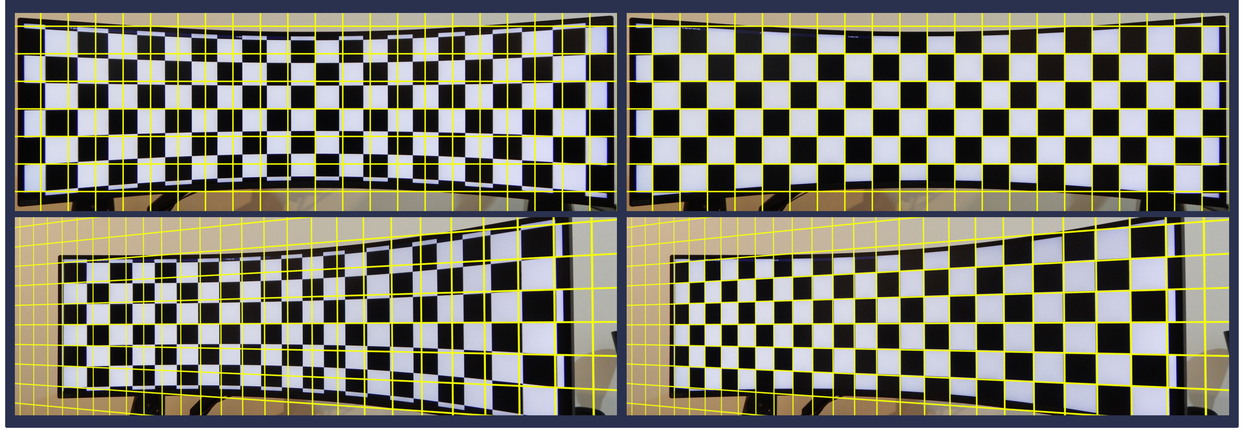
2. Existing Correction Methods

Multiple approaches exist for rendering correct images for curved monitors in real time. We implement and analyze three fundamentally different methods.

Ray tracing can generate various projections and produces correct images when the initial rays are cast accordingly[1].
Image-based methods warp a linear projection into the corrected projection in a post-processing pass[2].
Geometry-based methods approximate the corrected projection by distorting the geometry in the scene prior to rasterization[3].

RAY TRACING	IMAGE-BASED	GEOMETRY-BASED
+ Simple and accurate	+ Fastest solution	+ No resampling artifacts
− Computationally expensive	− Image resampling artifacts	− Geometric artifacts

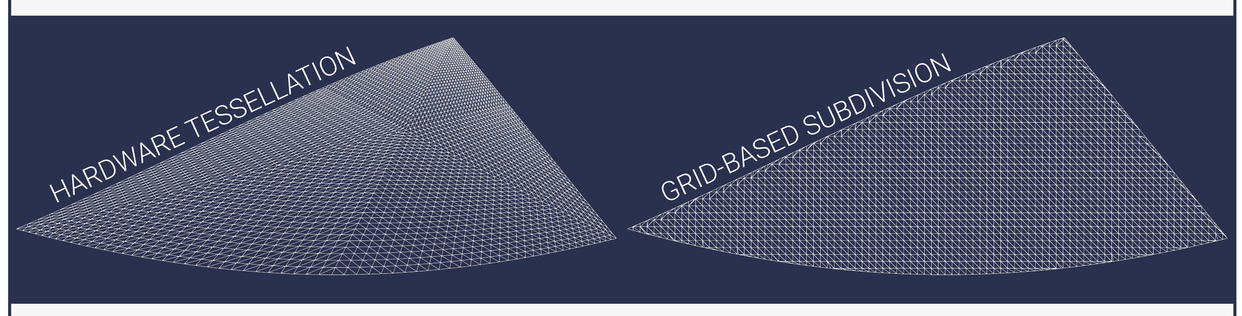
The impression of straight lines is preserved when the image is generated according to the user's viewpoint and the geometry of the monitor. As a result, the rendered checkerboard pattern pictured below appears rectangular.



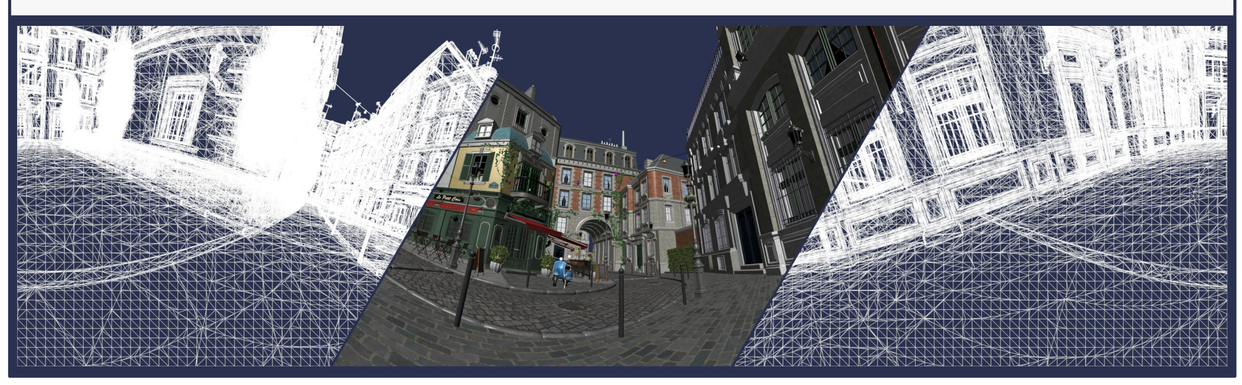
3. Contribution: Grid-Based Subdivision

The main challenge of geometry-based methods is that the scene geometry has to be very dense to avoid geometric artifacts. Otherwise, the planar triangles will not approximate the display surface well enough. The geometry can also be subdivided dynamically to ensure accurate results. In previous work, tessellation shaders have been used for this purpose[3].

Our work contributes an algorithmic description and evaluation of an alternative subdivision scheme, where triangles are subdivided along a 2D grid, as shown in the picture below. This subdivision can be implemented in software on the GPU in a single render pass using the recently introduced *graphics mesh pipeline*[4]. Our grid-based subdivision reflects the geometry of the display closely and produces fewer triangles for coarse geometry compared to hardware tessellation. Additionally, it virtually has no limit on the number of subdivisions.



Furthermore, the grid-based subdivision allows rendering 360° images without artifacts by default, as depicted below. Geometry-based solutions using tessellation shaders require an additional clipping stage to achieve the same.

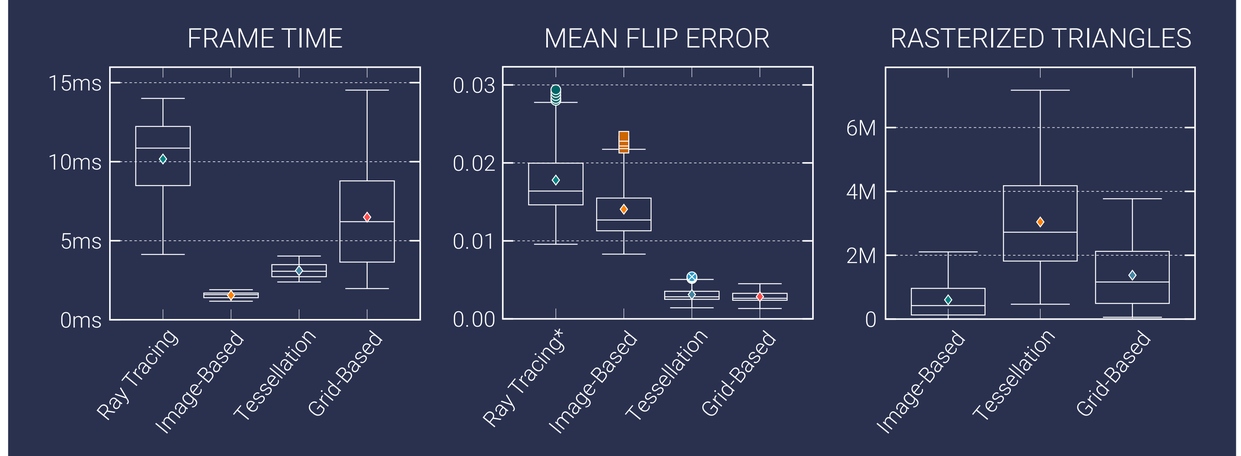


4. Results & Conclusion

We confirm that the image-based method represents the fastest solution and can produce acceptable image quality. In our implementation, it is faster by 1.3-2.6x on average compared to the geometry-based method using hardware tessellation. Thus, we recommend it for performance-critical applications.

Our implementation of the grid-based subdivision is slower than hardware tessellation by 2.1-2.3x on average. At the same time, it generates up to 72% fewer triangles for coarse geometry. We recognize the opportunity for further performance optimizations and see potential in using our approach for large structures with coarse geometry, such as buildings and streets.

The figure below shows the frame time, the perceptual difference to reference images using FLIP[5], and the number of triangles rendered for the Amazon Lumberyard Bistro[6] scene with a viewing distance of 100cm from the monitor.



*The relatively high FLIP error for ray tracing is expected and results from using a sampling pattern different from that employed for the reference images, which are generated using the rasterization pipeline. Therefore, the value does not reflect the image quality well in this case.

[1] Pietro Acquisto and Eduard Gröller. A distortion camera for ray tracing. *WIT Transactions on Information and Communication Technologies*, 5, 1993.
[2] Benjamin Allen Watson and Larry F. Hodges. Using texture maps to correct for optical distortion in head-mounted displays. In *Proceedings Virtual Reality Annual International Symposium'95*, pages 172–178. IEEE, 1995.
[3] Mariano Pérez, Silvia Rueda, and Juan M Orduña. Geometry-based methods for general non-planar perspective projections on curved displays. *The Journal of Supercomputing*, 75(3): 1241–1255, 2019.
[4] The Khronos® Vulkan Working Group. Vulkan® 1.3.212 - A Specification (with all registered vulkan extensions). https://registry.khronos.org/vulkan/specs/, 2022.
[5] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. Flip: A difference evaluator for alternating images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2), August 2020.
[6] Amazon lumberyard bistro. open researchcontent archive (orca). http://developer.nvidia.com/orca/amazon-lumberyard-bistro, July 2017. Last accessed on May 19, 2022.