On Foveated Rendering

Cara Tursun[1] ✉, Steffen Frey[1], and Jiří Kosinka[1]

[1] *Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands*

Abstract

Foveated rendering is a technique in computer graphics and visual computing that leverages the biological characteristics of the human eye to optimize resource allocation during rendering. The human eye has the highest visual acuity in the central area of vision, known as the fovea, while peripheral vision is less capable in resolving visual details. By rendering high-resolution graphics where the viewer's gaze is focused (the foveal region) and progressively decreasing the visual quality towards the periphery, foveated rendering ensures the efficient use of valuable computational resources in real-time, without comprising visual quality perceived by users.

The tight spot in rendering

Computer graphics is the field of generating images and videos using computers. To create an image, a computer usually requires a geometric and visual description of the scene together with objects and light sources inside the scene as input. Then, the computer generates an image from the given description of the scene as seen by a camera that is virtually placed inside the scene. This process resembles taking a photo of a scene with a camera in the real world.

When we take a photo in real life, the camera provides us with the image it "sees" when the light rays from the scene hit the sensor or photographic film. This image results from physical interactions between the light and the scene, which defines the final color and brightness of the light when it reaches the camera. When a light ray hits a surface, it gets reflected, absorbed, or transmitted (as we see in transparent materials like glass), as illustrated in Figure 1. Furthermore, the properties of object surfaces in the scene, their position, and angle with respect to the direction of the incident light, and the camera position also play a role in the interaction.

A computer's job in generating images and videos is not easy because it has to simulate those interactions between light and scene virtually, in a process called *rendering*. Obtaining a realistic, high-quality image via rendering requires a very accurate and expensive computation of complex interactions between the light and the scene. Furthermore, our displays keep improving in ways that increase the computational cost of rendering. As resolution increases, newer displays are able to showcase finer details, putting pressure on the renderer to keep up. The use of *Virtual reality* (VR) displays is on the rise,

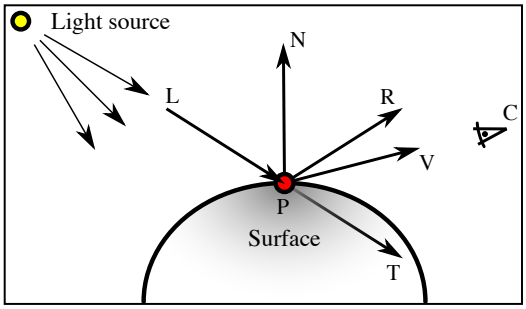


Figure 1: *Incoming light (L) from a light source can get absorbed, reflected (R), or transmitted (T) when interacting with a surface point (P) before it reaches the camera/eye (C). The normal vector (N) perpendicular to the surface at P and the vector (V) to the camera/eye can be used to estimate the resulting color observed at P.*

and this necessitates rendering the scene twice, for the left and the right eye. The use of *Augmented reality* (AR) headsets involves resource-intensive tasks, such as aligning virtual objects with real-world geometry, which is an extra computational expense. With these new developments, the task of maintaining the desired rendering quality has become even more demanding due to the increased computational requirements from our limited hardware resources.

When it comes to creating top-notch images and videos using computers, our focus is typically on utilizing advanced graphics processing units (GPUs) that are specifically designed for handling complex graphics tasks. However, this comes with challenges. First, the capabilities of GPUs are typically reflected in their corresponding purchase prices, and under a limited financial budget, we may have to explore alternative options instead of obtaining a top-of-the-

line GPU. Even with no financial constraints, deploying new hardware into existing systems may be hindered by limited physical access. Secondly, as the accuracy and complexity of rendering calculations improve, so does the energy consumption, a significant concern for battery-powered mobile devices. Finally, to accurately generate high-quality and realistic images, a greater amount of visual data must be stored, resulting in a larger storage space requirement. Due to technical constraints in transmitting and storing computer-generated images, it may not be possible to retain all necessary image details for optimal visual quality.

Our ultimate goal in computer graphics is to maintain a satisfactory level of visual quality. But, while pursuing this goal, can we simultaneously conserve resources, enhance performance, and achieve energy efficiency?

Render wars – A new hope

As computer graphics developers and researchers grapple with the balance between achieving high visual quality and managing rendering costs, they have begun exploring solutions to maximize the efficiency of limited computational resources. Recognizing that humans are the primary viewers of computer-generated images and videos, a notable strategy has emerged from the field of *visual perception*, which focuses on understanding the characteristics of the human visual system (HVS).

Visual perception provides valuable insights for optimizing graphics to match the capabilities of human vision. The process of human visual processing begins in the eyes, where light interacts with photoreceptors on the retina. These photoreceptors are distributed unevenly across the retina. The *fovea*, a region spanning about 5 degrees in the center of our visual field, is densely packed with photoreceptors that are particularly sensitive to color and intricate details. In contrast, our peripheral vision is characterized by photoreceptors that are sparser and more attuned to low light and detecting motion (Figure 2). Consequently, our ability to perceive fine details and colors diminishes as an image moves from the fovea to the periphery of our visual field. To experience this phenomenon first-hand, try a simple experiment: focus on a word in this paragraph and then attempt to read the words above it without moving your eyes. While this may be a bit challenging, you will find that you can recognize words at most 2 or 3 lines away using your peripheral vision, illustrating the variations in visual acuity at different parts of the retina.

In the realm of rendering, there is a constant pursuit for the highest possible resolution and quality in images and videos, often achieved by densely packing as many pixels as possible into our displays and cameras. However, unlike the human retina, where photoreceptors are unevenly distributed, the pixels in displays or camera sensors are arranged uniformly as a rectangular grid. Given this, devoting equal computational resources to render every part of an image or video in the highest quality seems inefficient, and a more resource-efficient approach involves the use of an *eye tracker* to determine where a viewer is looking on a display (Figure 3). By monitoring a viewer's gaze position on the display, it becomes possible to strategically allocate rendering resources

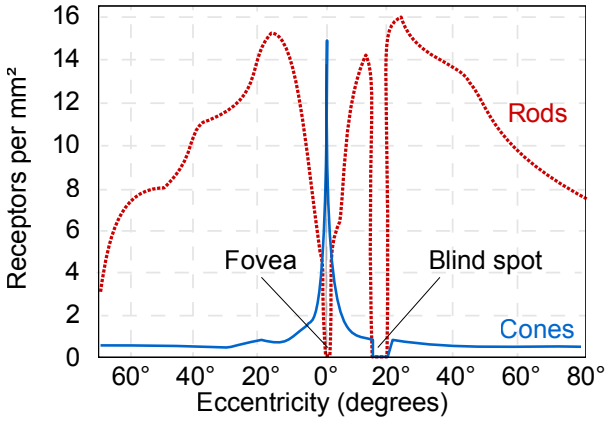


Figure 2: *The distribution density of the two types of photoreceptors in the human retina; cones and rods. The horizontal axis indicates the angular distance from the fovea, the center of our vision. Cones, which are active in well-lit conditions and enable color vision, are concentrated around fovea. Rods, which facilitate night vision are more dense in the peripheral areas of the retina. This variance in the density and distribution of photoreceptors underpins the difference in visual capabilities between central and peripheral vision [Wan95].*

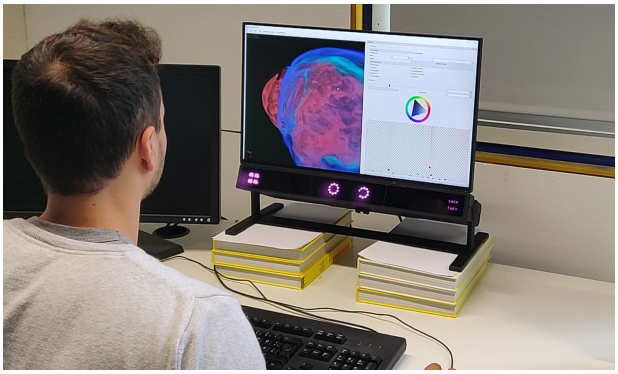


Figure 3: *A foveated rendering system consisting of a desktop display and an eye tracker positioned beneath the display.*

by reducing the quality in the peripheral vision where finer details are less discernible. This method ensures high-quality rendering where it matters most while conserving computational resources in areas less critical to the viewer's immediate focus. Inspired by the human retina, this technique is called *foveated rendering*.

Types of foveated rendering

By reducing the rendering quality in the peripheral vision, foveated rendering significantly reduces the computational load on the system. This allows for higher performance or it can enable more complex scenes or higher resolutions in the foveal region than would otherwise be possible. Many different flavors of foveated rendering have been developed over time, each employing different techniques to optimize the graphical performance. We describe several examples of this in the following paragraphs.

Standard dynamic foveated rendering Foveated rendering, in its simplest form, organizes the visual field into concentric zones around the point of gaze, progressively decreasing rendering quality (such as resolution) with increasing distance from the fovea [GFD+12]. This approach leverages a human perception model informed by the distribution of photoreceptor cells (Figure 2), making it relatively straightforward to implement. Consequently, as one moves further from the viewer's point of gaze, the rendering quality on the display diminishes. This concept is illustrated in the rendering resolution map for the standard dynamic foveated rendering technique, depicted in Figure 4 (b). Techniques such as contrast enhancement in post-processing or the introduction of synthesized noise, which replicates the spatial details lost due to reduced rendering quality, can help mitigate the visibility of this quality reduction [PSK+16, TTD22].

Content-aware foveated rendering The standard dynamic foveated rendering maintains a constant quality across different visual contents, disregarding the nature of the content itself. This approach can lead to varying levels of visibility in quality reduction, especially in areas with complex details or high contrast. On the other hand, content-aware foveated rendering adapts the quality based on both the viewer's gaze distance and the complexity of the visual content [TAKW+19]. Implementing this adaptive approach is more complex because it requires adjusting the rendering quality based on two criteria: the proximity to the gaze point and the intricacy of the content. Additionally, these adjustments must be made using a simplified version of the scene to ensure efficiency. Rendering the scene in full resolution for quality assessment would negate the benefits of reduced rendering, thus the challenge lies in making these decisions based on a representative, lower-resolution preview of the scene. A sample resolution map for this technique is shown in Figure 4 (b), corresponding to the 3D scene in Figure 4 (a).

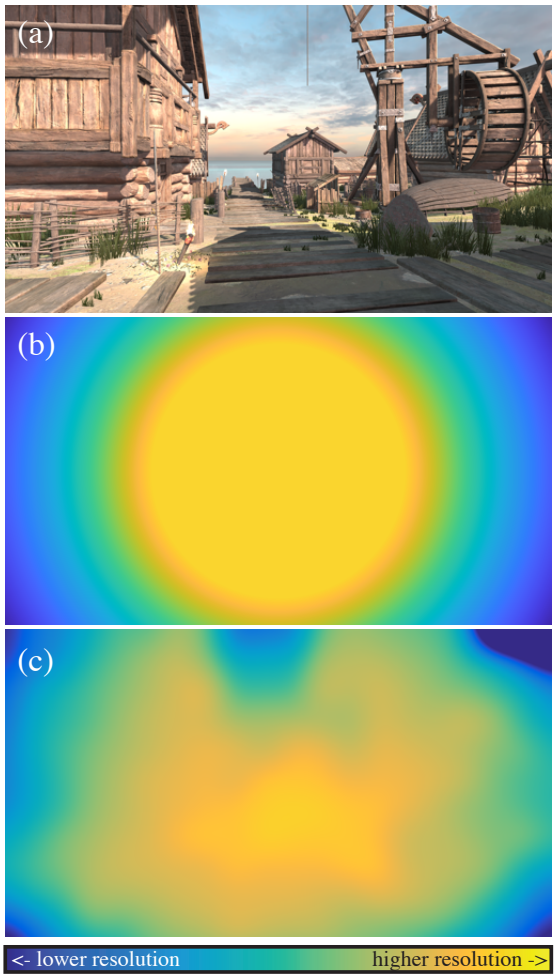


Figure 4: *For a 3D input scene (a), images (b) and (c) show the color-coded rendering resolutions achieved through the standard dynamic foveated rendering and the content-aware foveated rendering, respectively. In both cases, it is assumed that the viewer is looking at the center of the display. Content-aware foveated rendering optimizes the use of the rendering budget by considering the visual content when determining the resolution.*

Foveated tessellation Objects and characters in digital scenes, such as for computer games or animated movies, are typically described using meshes, i.e., collections of polygons such as triangles. While GPUs are increasingly more capable of quickly processing and rendering models with lots of triangles,

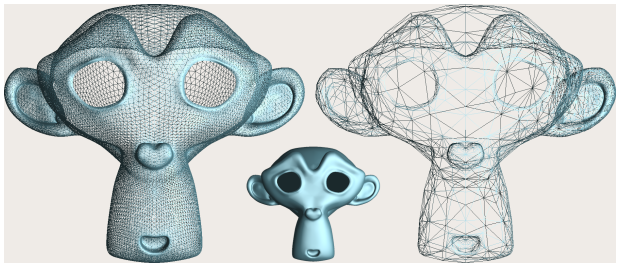
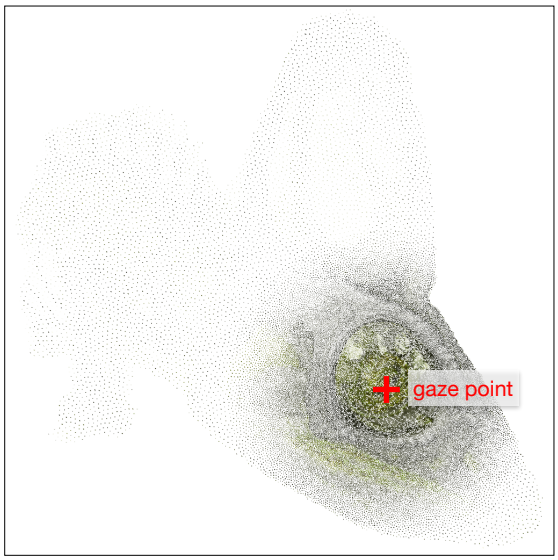


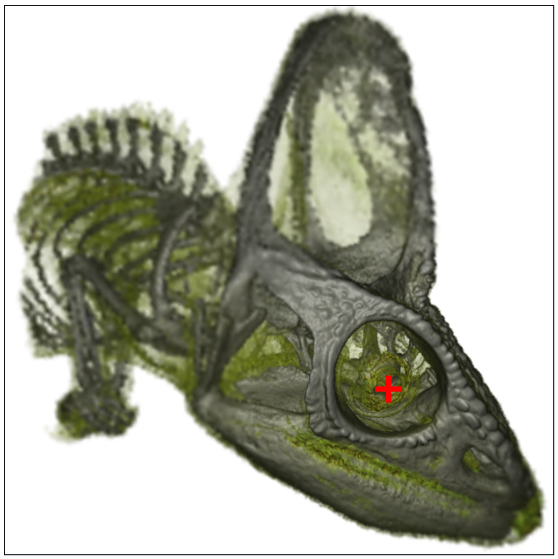
Figure 5: *Foveated tessellation is demonstrated on a monkey head model (known as Blender's Suzanne), which consists of 856 curved faces (middle image). The foveated approach utilizes only 20% of the triangles compared to the fully tessellated model. On the left, the model is globally tessellated, comprising 50k triangles. On the right, foveated tessellation is applied, resulting in 10k triangles. This technique adaptively generates triangles on the GPU, focusing only on the regions the (virtual) camera or user is observing – in this case, the right ear.*

transferring these from the CPU to the GPU every frame in real-time applications quickly becomes a bottleneck for animated models (i.e., models where the positions of the vertices/triangles keep changing). To alleviate this, a coarse (or even compressed) description of the model is sent from the CPU, and the details are then interpolated using various smooth representation methods, such as splines or subdivision (the latter being the animation's industry standard), using a technique called tessellation on the GPU. However, when used naively, it leads to (unnecessarily) dense models that need to be rendered. Enter foveated tessellation. Based on the user's gaze/-center of the field of view, we can use tessellation adaptively, inventing more triangles for the foveal area while leaving the peripheral region comparably sparse [TRK20]. The example in Figure 5 shows a five-fold reduction in the number of triangles needed to render a smooth-appearing model with foveated tessellation compared to standard (naive) model tessellation.

Foveated volume ray-casting Volume ray-casting is a technique used for rendering 3D volumetric data. When adapting this technique for foveated rendering, a unique approach is necessary due to the inherent characteristics of volume ray-casting. One effective strategy integrates the gradual decrease in human visual acuity towards the periphery with specific sampling and interpolation methods suitable for volumetric data, such as Linde-Buzo-Gray sampling and natural neighbor interpolation as illustrated in Figure 6 [BSB+19]. This approach involves rendering samples of 3D volumetric data, as depicted in Figure 6 (a), which are then used as input for the reconstruction process shown in Figure 6 (b). The interpolation technique employed for reconstruction is more computationally efficient than densely sampling the volumetric data, as is done in full-resolution rendering. This leads to significant enhancements in rendering performance with a minimal impact on visual quality.



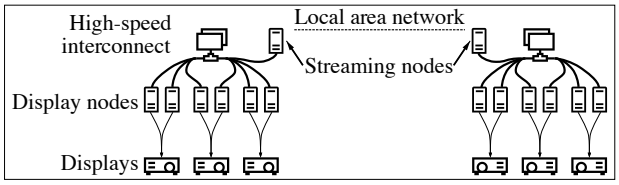
(a) *Image space sampling*



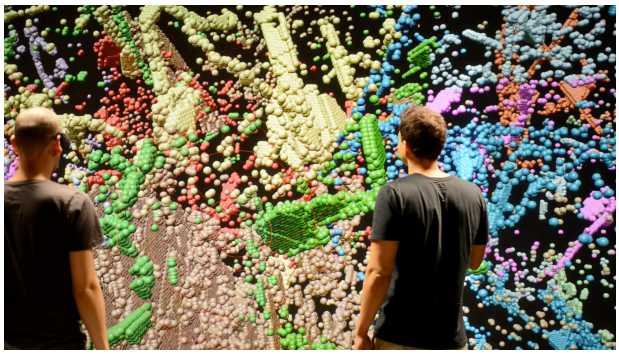
(b) *Reconstruction*

Figure 6: *Voronoi-based foveated volume ray-casting result [BSB+19].*

Foveated streaming Foveated rendering approaches can further facilitate collaborative exploration of scientific data sets across large high-resolution displays, which requires both high visual detail as well as low-latency transfer of image data



(a) Overview of the technique



(b) An application on a large display, known as powerwall

Figure 7: *Foveated streaming for large high-resolution displays [FBB+21]. The diagram in (a) shows an overview of the method, whereas the image in (b) shows the method in action with multiple viewers observing an application with particle rendering.*

(Figure 7). Using foveated rendering to dynamically adapt the encoding quality while streaming the data reduces the required bandwidth [FBB+21] (a). Different from single-user foveated rendering, this technique requires tracking the gaze of multiple viewers, and respectively adapting the quality parameter of the video encoder. This allows us to substantially reduce the required bandwidth for transferring the data with low latency, which is crucial for collaborative analysis across display walls at different physical locations. An overview and a sample use case of this technique is depicted in Figure 7.

Conclusion

We have described foveated rendering and its successes in a selected set of different applications. But the story does not end here. With the popularity of the Augmented and Virtual Reality on the rise, methods exploiting foveated methods will attain more widespread use with more challenging use cases that require creative solutions.

References

[BSB+19] Valentin Bruder, Christoph Schulz, Ruben Bauer, Steffen Frey, Daniel Weiskopf, and Thomas Ertl. *Voronoi-Based Foveated Volume Rendering*. The Eurographics Association, 2019. Accepted: 2019-06-02T18:14:41Z.

[FBB+21] Florian Frieß, Matthias Braun, Valentin Bruder, Steffen Frey, Guido Reina, and Thomas Ertl. Foveated Encoding for Large High-Resolution Displays. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1850–1859, February 2021. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[GFD+12] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM transactions on Graphics (tOG)*, 31(6):1–10, 2012.

[PSK+16] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[TAKW+19] Okan Tarhan Tursun, Elena Arabadzhiyska-Koleva, Marek Wernikowski, Radosław Mantiuk, Hans-Peter Seidel, Karol Myszkowski, and Piotr Didyk. Luminance-contrast-aware foveated rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 38(4), 2019.

[TRK20] Ankur Tiwary, Muthuganapathy Ramanathan, and Jiří Kosinka. Accelerated Foveated Rendering based on Adaptive Tessellation. In Alexander Wilkie and Francesco Banterle, editors, *Eurographics 2020 - Short Papers*. The Eurographics Association, 2020.

[TTD22] Taimoor Tariq, Cara Tursun, and Piotr Didyk. Noise-based enhancement for foveated rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022.

[Wan95] Brian A Wandell. *Foundations of vision*. Sinauer Associates, 1995.