Scene-aware Foveated Rendering

Runze Fan, Xuehuai Shi, Kangyu Wang, Qixiang Ma, and Lili Wang

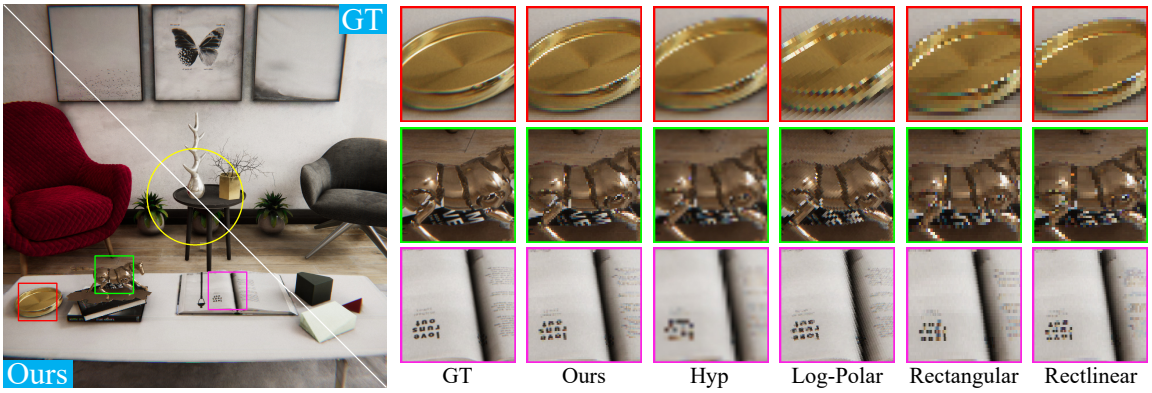


Fig. 1: Left: Comparison of the proposed foveated rendering (lower-left) and the full-resolution rendering (ground truth, GT, upper-right). Right: As illustrated in the close-ups of the rendered images, compared with the Hyp method [1], the Log-Polar method [2], the Rectangular method [3], and the Rectlinear method [4], our results are closer to the ground truth, and are able to retain clearer details in the periphery region with the same compression ratio as the above methods.

Abstract— We propose a new scene-aware foveated rendering method, which incorporates the scene awareness and characteristics of the human visual system into the mapping-based foveated rendering framework. First, we generate the conservative visual importance map that encodes the visual features of the scene, visual acuity, and gaze motion. Second, we construct the pixel size control map using a convolution kernel method. Third, we utilize the pixel size control map to guide the foveated rendering. At last, a temporal coherent refinement strategy is used to maintain the smooth foveated rendering for the adjacent frames. Compared to the state-of-the-art mapping-based foveated rendering methods using the same compression ratio, our method achieves smaller MSE, higher PSNR, and SSIM in the fovea, periphery, salient regions, and the whole image. We also conducted user studies, and the results proved that the perceptual quality of our method has a high visual similarity with the ground truth rendered with the full resolution.

Index Terms— Virtual reality, Foveated rendering, Visual importance, Pixel size control

1 Introduction

Virtual Reality (VR) is widely used in more and more fields such as entertainment and gaming, culture and tourism, and manufacturing, which puts higher demands on the realism and efficiency of rendering. Foveated rendering techniques provide a possible solution to accelerate rendering without compromising visual perceptual quality. It utilizes characteristics of the Human Visual System (HVS) to reduce computational costs and improve performance by rendering only content that is actively perceived by the user [5].

Many researchers have investigated rasterization-based and ray tracing-based foveated rendering [1, 6–15]. To avoid repeated rendering and waste of computational resources, more generalized mapping-based foveated rendering methods have been proposed [2–4]. These methods use a typical deferred shading pipeline. First, they generate the g-buffer in the regular geometry pass. Then, they employ a mapping function to map the original g-buffer to a reduced resolution mapped g-buffer. Subsequently, uniform shading is carried out on the mapped g-buffer, and the final rendering results are generated using the inverse mapping function. Different from [13–15], these methods consistently take into account the sampling rates for both rasterization and shading, and allow for continuous non-uniform compression. However, existing mapping-based foveated rendering methods use relatively simple mapping and inverse mapping functions that are constructed only based on characteristics of the HVS without being aware of the scene, i.e., they do not consider the visual features of the scene, which results in poor rendering quality in locations with high visual features in the periphery region. Recent studies show that peripheral vision is fundamental for many visual tasks, including walking, driving, and aviation. Many visual features, such as salient features, are constantly processed in the periphery region [16]. To further improve the quality of visual perception, it is crucial for foveated rendering to maintain the high-quality visual features in the periphery region [5, 17, 18].

In this paper, we propose a scene-aware foveated rendering method, which incorporates the scene awareness and characteristics of the HVS into the mapping-based foveated rendering framework. Scene awareness means the detection and processing of visual features of the scene, and HVS is characterized by visual acuity and gaze motion. We use the conservative visual importance map to encode the visual importance distribution. Unlike the previous mapping-based foveated rendering methods, we use the pixel size control map to guide the allocation of computing resources for each pixel. First, we generate the conservative visual importance map, which encodes the visual features of the scene, visual acuity, and gaze motion and keeps the total cumulative importance of the map constant. Second, we construct the pixel size control map with a convolution kernel method based on the conservative visual importance map. Third, foveated rendering is executed with the guide of the pixel size control map. At last, a temporal coherent refinement strategy is used to maintain the smooth foveated rendering for the adjacent frames.

Compared to the state-of-the-art methods, our method achieves smaller MSE, higher PSNR, and SSIM in the fovea, periphery, salient regions, and the whole image. Our method has a similar performance as the previous methods and achieves 1.65× speedup with similar visual perceptual quality compared to the full-resolution rendering. We also conducted user studies, which proved that the perceptual quality of our method has a high visual similarity with the results of the full-resolution rendering. Fig. 1 shows the rendering results of our method. The close-ups illustrate the rendering details in the periphery region of our method compared to the full-resolution(ground truth, GT), the Hyp method [1], the Log-Polar [2], the Rectangular [3] and the Rectlinear methods [4].

In summary, the contributions of this paper are: 1) a new scene-

- *Lili Wang is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China; Peng Cheng Laboratory, Shengzhen, China. Lili Wang is the corresponding author. E-mail: wanglily@buaa.edu.cn.*
- *Runze Fan, Kangyu Wang and Qixiang Ma is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China. E-mail: by2106131@buaa.edu.cn, sy2306314@buaa.edu.cn, sycamore_ma@outlook.com.*
- *Xuehuai Shi is with Nanjing University of Posts and Telecommunications. E-mail: xuehuai@njupt.edu.cn.*

Manuscript received 14 March 2024; revised 4 June 2024; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

aware foveated rendering pipeline; 2) a conservative visual importance map, which encodes the visual features of the scene, visual acuity, and gaze motion and keeps the total cumulative importance of the map constant; 3) a pixel size control map to guide the allocation of computing resources and a convolution kernel method to construct this map; 4) a temporal coherent refinement strategy to maintain the temporal consistency of the visual importance for the adjacent frames, which leads to smooth foveated rendering.

2 Related Work

In this section, we first give a brief review of the research on foveated 3D rendering, and then discuss the methods to improve the visual quality of periphery region. For a more comprehensive review of foveated 3D rendering, we recommend the readers refer to the reviews [19, 20].

2.1 Foveated 3D rendering

Early research in the foveated 3D rendering focused on ray tracing based methods [6–11]. To further improve rendering efficiency, some researchers focused on rasterization-based foveated rendering. Guenter et al. [1] rendered three image layers around the gaze point with discrete sampling rates. The fovea region is repeatedly rendered 3 times, which lowers the efficiency of the rendering. Stengel et al. [17] linearly expanded the fovea region based on the gaze motion vector, resulting in smooth following of gaze motion. This method also suffered from wasted computational resources. Turner et al. [21] used multiple low-resolution results and one high-resolution result to generate the final result by aligning the rendered pixel grid in the periphery region to the virtual scene during rasterization.

Many researchers worked on avoiding repeated rendering of the fovea region. Inspired by the Coarse Pixel Shading technique [22] (CPS), Patney et al. [5] decoupled shading and visibility by quantizing the shading rate into a finite set of screen-aligned grids and achieved foveated rendering by sampling coarse pixels in the periphery region. Similiar, Variable Rate Shading [14] (VRS) increased the rendering performance and quality by varying the shading rate for different regions of the image. These CPS or VRS-based methods can shade the scene at a variety of rates, including the non-square fovea region. However, these methods require the adaptive shading feature, which is not yet supported on some commodity GPUs.

Inspired by that the excitation of the cortex can be approximated by a Log-Polar mapping of the eye's retinal image [23, 24], Meng et al. [2] proposed a two-pass foveated rendering pipeline that parameterizes foveated rendering by introducing a polynomial kernel function in the Log-Polar coordinate mapping. Similarly, Ye et al. [3] proposed a rectangular mapping-based foveated rendering method, which processed the horizontal and vertical directions independently. Li et al. [4] expanded the Log-Polar mapping to the field of foveated 360-degree video streaming, and proposed a rectilinear foveated mapping method, which also processed the horizontal and vertical directions independently. While these mapping-based foveated rendering methods avoid repeated rendering of the fovea region, none of them take into account the quality decrease of the periphery region where the visual features of the scene are located.

Our scene-aware foveated rendering method belongs to the mapping-based foveated rendering framework. It takes into account both the user's visual acuity and gaze motion, as well as the visual features of the scene.

2.2 Improve the Visual Quality in Periphery Region

In the foveated rendering, blurred rendering results for the periphery region with visual features can make users perceive a decrease in the visual quality. Therefore, many methods have been proposed to improve the visual perceptual quality of the periphery region.

Stengel et al. [17] allocated computational resources to pixels based not only on visual acuity but also on gaze direction, eye movements, contrast, and brightness. Okan et al. [25] and Shi et al. [26] improved visual quality in the periphery region by analyzing the local luminance contrast of the image. Walton et al. [27] computed a feature pyramid consisting of steerable filter responses on multiple scales and generated a new image based on the extracted statistics by collapsing this pyramid, resulting in high visual quality in the periphery region. Krajancich et al. [28] proposed an attention-aware foveated rendering method. It analyzes the factors affecting periphery visual perception, such as illumination and attention, and adaptively adjusts the spatial resolution of the rendered images in the periphery region according to these factors, thus improving visual quality in the periphery region. Lisboa et al. [29] analyzed the effect of movement on the perception of periphery region in VR and gave a relationship between movement speed and foveated rendering parameters.

Many methods have been proposed to adjust the resolution of the rendered images in the periphery region by using frequency analysis. Jindal et al. [30] proposed a Just-noticeable-difference resolution based on frequency domain analysis, and determined the resolution of the periphery region based on the content-adaptive metric of judder, aliasing, and blur. Denes et al. [31] proposed a visual perceptual model that predicts the just-noticeable-difference resolution by giving an object velocity and predictability of motion. Taimoor et al. [32] gave the range of frequencies that are detectable but not distinguishable at a given eccentricity. These frequencies were replaced by noise generated in a less costly post-processing step to improve the visual quality in the periphery region.

Besides, with the development of deep learning techniques [33, 34], many researchers have applied them in foveated rendering to improve the quality in the periphery region. Deng et al. [35] proposed FoV-NeRF, which improves the visual quality in the periphery region by adjusting the sampling density of the 3D neural representation. Bauer et al. [36] proposed FovolNet, a fast-foveated deep neural network, and the visual quality in the periphery region could be improved easily with the reconstruction network. Kerbal et al. [37] proposed 3DGS which has greatly improved the performance and quality compared with previous NeRF-base methods, which provides a basis for further improving the efficiency and quality of foveated rendering.

Our method focuses on improving the visual quality in the periphery region for the mapping-based foveated rendering method.

3 Method

3.1 Pipeline

We propose a scene-aware foveated rendering method. Our method adopts a typical deferred shading pipeline. The pipeline is shown in Fig. 2. There are three main steps in our foveated rendering process: Step 1: generating the conservative visual importance map (Section 3.2); Step 2: constructing the pixel size control map (Section 3.3); Step 3: foveated rendering guided with the pixel size control map (Section 3.4).

3.2 Conservative Visual Importance Map Generation

3.2.1 Definition

The conservative visual importance map $cvim_{H,W}$ is a 2D map, where H and W are the vertical and horizontal resolutions of the g-buffer. The value of each pixel in $cvim$ represents the visual importance of the scene at the corresponding location. The visual importance ranges from 0 to 1 and can be calculated based on the visual features of the scene, visual acuity, and gaze motion. Conservation means that the sum of the visual importance values of all pixels in $cvim_{H,W}$ is always constant $h \times w$, where h and w are the vertical and horizontal resolutions of the mapped g-buffer.

3.2.2 Generation

Weier et al. [20] summarized the factors that influence human visual perception, such as eccentricity, visual features, and gaze motion. Inspired by this idea, we first construct a visual acuity map based on eccentricity. Then, we extract a visual feature map according to the scene. Finally, we combine them to generate the visual importance map with consideration of gaze motion. The visual acuity map, the visual feature map and the visual important map are generated for each frame.

Visual Acuity Map Construction. A visual acuity map $vam_{H,W}$ is a 2D map of the same size as the g-buffer, in which the pixel value indicates the user's visual acuity (measured in terms of the minimum angle of resolution) according to the gaze point. Given the gaze point, the acuity a of each pixel in vam is computed with the visual acuity model [1]. The model uses acuity limit w_0 and acuity slope m to characterize the user's visual acuity.

Visual Feature Map Extraction. A visual feature map $vfm_{H,W}$ is a 2D map of the same size as the g-buffer, in which the pixel value indicates the intensity of the scene's visual features. We take the low-level visual features, such as saliency, into account to generate the visual feature map. Given a g-buffer, we detect the object saliency, silhouette, and highlight and combine them to get the visual feature value f ($[0,1]$) of each pixel with the perceptual filter method [17].

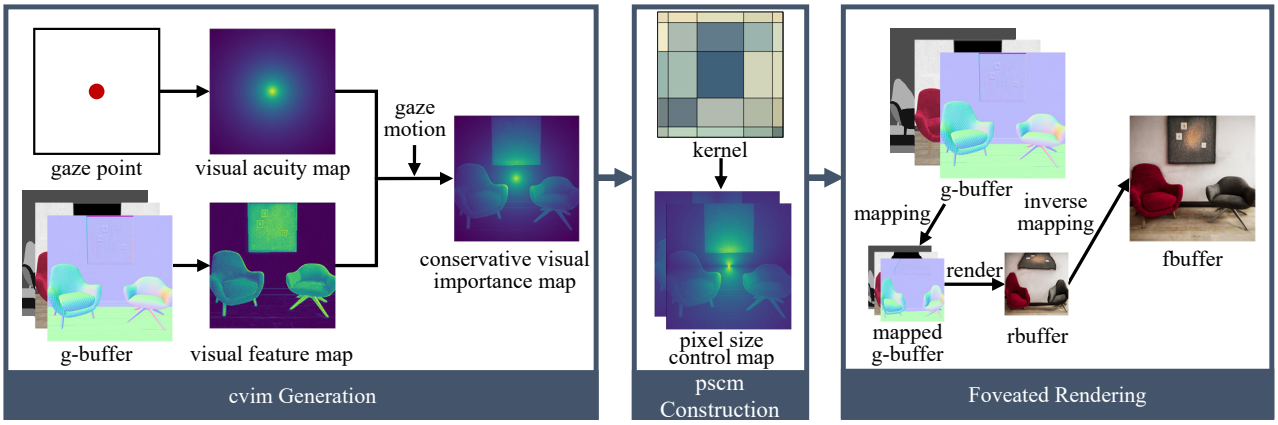


Fig. 2: There are three main steps in our foveated rendering process. In the cvim generation step, the visual acuity map is generated according to the gaze point, and the visual feature map is extracted based on the g-buffer. Then, the conservative visual importance map is constructed by combining the visual feature map and the visual acuity map according to the gaze motion. In the pscm construction step, the convolutional kernel is first built according to HVS. Next, the convolution is operated with the kernel to construct the pixel size control map. In the foveated rendering step, g-buffer is first transformed to a mapped g-buffer. Then, reduced-resolution rbuffer is rendered based on the mapped g-buffer. The final full-resolutuon fbuffer is generated based on the inverse mapping.

Visual Importance Computation. The visual importance vi in $cvim$ indicates the visual perceptual importance of each pixel to the HVS. Visual acuity a rates human's ability to recognize small details with precision. When measured in terms of the minimum angle of resolution, the larger a, the larger the smallest detail that can be distinguished, and details of the same size are less visually important to the HVS. Thus vi is inversely proportional to visual acuity a,i.e.,$vi \propto {}^{1}/_{a}$.

Visual feature f indicates the intensity of the scene's visual features, which attracts people's attention. The larger f, the more attention is attracted to the pixels with visual features, i.e., the greater visual importance of these pixels to the HVS. Thus vi is positively related to visual feature f. Strasburger et al. [38] used the spotlight of attention to describe the phenomenon in HVS where attention is attracted to a specific area caused by visual features. Marzecov et al. [39] characterized this mechanism as a gain-control model with attention amplifying the response to the stimulus, the amplification factor is gain. Wang et al. [40] and Sam et al. [41] proposed the linear or nearly linear relationship between gain and visual feature. According to the previous work, vi is directly proportional to the intensity of visual features f,i.e.,$vi \propto 1+k\cdot f$. 1 indicated the original vi caused by other factors, such as visual acuity. k is a weight of f and can be obtained with a pilot user study.

Reddy et al. [42] pointed out the relationship between visual acuity and the velocity at which details crossed the retina (gaze motion), where visual acuity was inversely proportional to the gaze motion influence function $G(v)$, and v represents the gaze motion velocity.

Based on the above analysis, the visual importance is calculated with Eq. 1.

$$vi = \frac{G(v)}{a} + \frac{k\cdot f}{a} \tag{1}$$

The reason the molecule of the second term of the Eq. 1 is not multiplied by $G(v)$ is that $G(v)$ is used to describe the effect of the gaze motion (velocity) on visual acuity (the 1 part in $1+k\cdot f$). While $k\cdot f$ is used to describe the attention effect of visual features on visual acuity, which is independent of $G(v)$.

For the mapping-based foveated rendering methods, the computational resources are fixed, which requires that the conservation of the visual importance map, i.e., the sum of the visual importance must remain constant. Therefore, the final visual importance is normalized with the function $NorVi$ (Eq. 2).

$$NorVi(vi) = hw\cdot vi/\sum(vi) \tag{2}$$

3.3 Pixel Size Control Map Construction

3.3.1 Definition.

The pixel size control map $pscm_{H,W}$ is a 2D map with the same size as $cvim$. $pscm$ is used to control the corresponding pixel size in the g-buffer after mapping. Each element of $pscm$ has two components representing the vertical and horizontal size of the pixels. These sizes are calculated according to the visual importance.

We choose a map-based approach to control the size of the mapped pixel instead of the conventional mapping function-based approach to generate the mapped g-buffer. This is because existing methods, such as [2–4], only consider the visual acuity associated with the gaze point, so it is relatively easy to find the mapping function. Our scene-aware method not only considers the visual acuity but also the visual features of the scene. Since the density and location of the visual feature vary with the scene, it is difficult to fit with a single function.

3.3.2 Construction

In this section, we first introduce how to compute the mapped pixel size based on the characteristics of visual acuity. Then, we introduce how to add the influence of the visual feature into the pixel size computation, i.e., how to compute the pixel size based on the visual importance. Next, to speed up the computation, based on the above computation method, we build a logarithmic-convolution kernel and propose a convolution construction method for the pixel size control map based on the visual importance map. Finally, to perform inverse mapping after rendering, we propose a construction method for the inverse pixel size control map.

Visual Acuity based Pixel Size Computation. The real world projected in the retinas of eyes is reconfigured onto the cortex by a topological transformation process before it is examined by the brain [43], and this topological transformation can be approximated by a Log-Polar mapping, which transforms a circular region in the Cartesian coordinate system into a rectangular region in the polar coordinate system. The area of the mapped pixels can be calculated by the integral of $a(e)$ over the original pixel, where $a(e)$ denotes the visual acuity calculated based on the eccentricity e.

As shown in Fig.3, (a) illustrates a g-buffer with resolution 7×7, the red dot in the upper left corner with coordinates $(0,0)$ denotes the gaze point, the yellow block denotes the visual feature. (b) illustrates the $cvim$ computed based on Eq. 2. The lighter the pixel's color, the greater the visual importance. It can be seen that visual importance is higher close to the gaze point and the location of the visual feature. (c) illustrates the g-buffer representing each pixel in a distinct color for clear viewing. (d) illustrates the area of each mapped pixel calculated according to the Log-Polar mapping. The black region in (d) indicates no pixels are mapped to it. We stretch the circular region in (d) into a square region in (e) to utilize the buffer completely. Since the mapping function between (c) and (e) needs to be subjective and injective, this mapping needs to follow two constraints. Constrain 1: for a square region, the mapped region of it remains square (subjective). Constrain 2: the adjacency between pixels within the region and on the boundary remains unchanged (injective).

For constrain 1, the square in Fig.3 (c) framed in red with size s_i is mapped to the square region in (e) with size $s_i^{'}$ using the Eq. 3.

$$s_i^{'} = sqrt(\int_0^{s_i}\int_0^{s_i} \frac{1}{a(e)}dxdy) \approx \sqrt{\frac{\pi}{2m}\ln(1+\frac{m}{w_0}s_i)} \tag{3}$$

For constraint 2, the adjacency of all pixels is guaranteed as long as the pixels on the edges of the square are adjacent. We

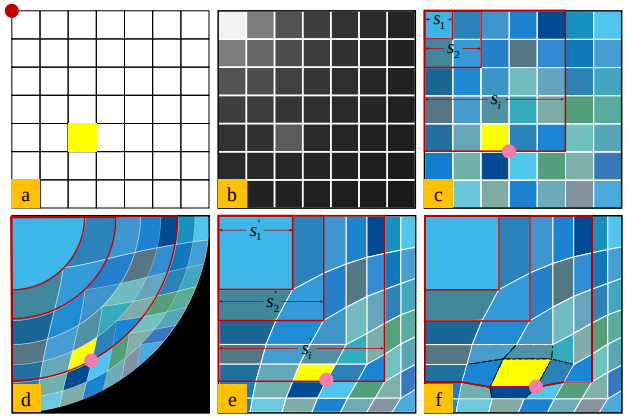


Fig. 3: (a) illustrates a g-buffer with resolution 7×7, the red dot in the upper left corner denotes the gaze point, and the yellow mark denotes the visual feature. (b) illustrates the *cvim* computed based on Eq. 2. (c) illustrates the g-buffer representing each pixel in a distinct color for clear viewing. (d) illustrates the size of each mapped pixel calculated based on Log-Polar mapping. We stretch the circular region in (d) into a square region in (e) to maximize computational resources. (f) illustrates how the size of neighboring pixels changes when the pixel size of the yellow pixel changes accordingly.

ensure pixel adjacency by computing the lengths of the 4 edges of the mapped pixels, so that the pixels may not be rectangles, but rather resemble parallelograms or trapezoids. Take the yellow pixel in Fig.3 (c) for example, we only need to calculate the mapped coordinates of the lower right corner(pink point) of this pixel. The upper left corner is determined by its upper left pixel, the upper right corner is determined by its upper pixel, and the lower left corner is determined by its left pixel. For the pink point with coordinates $(x, y = s_i)$, the vertical coordinate v of the mapped pink point equals the size $s_i^{'}$ of the mapped square region. The horizontal coordinate u is determined by the ratio of the sum of the visual acuity of the pixels at the left of the pink point to the sum of the acuity of all the pixels located on the edges of the square region (Eq. 4).

$$u = s_i^{'} \times \frac{\int_{s_i}^{\sqrt{x^2+s_i^2}} 1/a(e)dx}{\int_{s_i}^{\sqrt{2}s_i} 1/a(e)dx} \approx s_i^{'} \times \frac{\ln\left(1+mx/\left(w_0+ms_i\right)\right)}{\ln\left(1+ms_i/\left(w_0+ms_i\right)\right)} \tag{4}$$

Equations for pixels on the right edge of the square can be derived similarly. For simplicity, we only list the mathematical derivation of the pixels located to the lower right of the gaze point, and set the gaze point as $(0,0)$. When the coordinates of the gaze point is not 0, equations for the remaining pixels can be derived similarly.

For any pixels (x,y) located to the lower right of the gaze point on the g-buffer, the coordinates (u,v) of mapped pixels on the mapped g-buffer are computed with Eq. 5.

$$\begin{cases} u(x,y) = \frac{\ln(1+mx/(w_0+my))}{\ln(1+my/(w_0+my))}\sqrt{\frac{\pi}{2m}\ln\left(1+my/w_0\right)} \\ v(x,y) = \sqrt{\frac{\pi}{2m}\ln\left(1+my/w_0\right)} \end{cases} \tag{5}$$

Based on our mapping function (Eq. 5), the horizontal and vertical sizes of mapped pixels can be calculated as Eq. 6.

$$\begin{cases} \Delta u(x,y) = u(x+1,y) - u(x,y) \approx \frac{\partial u}{\partial x} \cdot \Delta x \\ \Delta v(x,y) = v(x,y+1) - v(x,y) \approx \frac{\partial v}{\partial y} \cdot \Delta y \end{cases} \tag{6}$$

The partial derivatives are calculated by Eq. 5. Inversely, for mapped pixels (u,v), the coordinates (x,y) of the corresponding original pixels on the g-buffer are computed with Eq. 7.

$$\begin{cases} x(u,v) = \frac{w_0}{m}\exp\left(\frac{2m}{\pi}v^2\right)\left(\left(2-1/\exp\left(\frac{2m}{\pi}v^2\right)\right)^{u/v}-1\right) \\ y(u,v) = \frac{w_0}{m}\left(\exp\left(\frac{2m}{\pi}v^2\right)-1\right) \end{cases} \tag{7}$$

Visual Importance based Pixel Size Computation. As discussed in section 3.2, the ideal visual importance is related to visual acuity, gaze motion, and visual features. In this section, we use the area of the mapped pixel calculated with Eq. 6 and the visual importance to adjust the size of the mapped pixel to take into account the effects of the visual features of the scene and gaze motion. For the mapped pixels, we approximate their shape as a parallelogram (where $x \neq y$) or two adjacent triangles (where $x = y$). Given the visual importance vi, the optimized horizontal and vertical sizes $\Delta(u/v)_{vi}$ of mapped pixels are calculated with Eq. 8.

$$\Delta(u/v)_{vi} = \sqrt{\frac{\Delta(u/v)}{\Delta(v/u)}} \cdot vi \tag{8}$$

Convolution Construction Method for Pixel Size Control Map. The size of the mapped pixel is not only related to its visual importance but also to that of its neighboring pixels because the adjacency of the mapped pixels needs to be ensured. If we change the size of just one pixel, it would result in overlapping pixels or gaps between pixels. Thus, we vary the size of the pixel and its neighboring pixels. The most straightforward computational strategy is to traverse all pixels and compute the horizontal and vertical change in pixel size $c_{\Delta u/\Delta v}$ for itself and its neighboring pixels. Because the 'spotlight of attention' is typically assumed to decay monotonically around its center [38], we use a logarithmic function to control the changes in the size of neighboring pixels, rather than uniformly scaling the size of neighboring pixels. Therefore $c_{\Delta u/\Delta v}$ is proportional to $ln(1+dis)$, where dis is the horizontal and vertical distance between the adjacent pixel and the center pixel. Fig.3 (f) shows a pixel of higher visual importance with a yellow block. $c_{\Delta u/\Delta v}$ of its neighbor pixel ($n \times n$, $n = 2l+1$, inside the black dotted frame) are calculated based on Eq. 9.

$$c_{\Delta u/\Delta v}(i,j) = \frac{\ln(1+|i/j-l|)}{\sum_{i/j\in[0,2l]}^{i/j\neq l}\ln(1+|i/j-l|)}\left(\Delta(u/v) - \Delta(u/v)_{vi}\right) \tag{9}$$

To speed up the computation, we propose a convolution construction method to avoid repeated computation of individual pixel size change. First, we construct the kernel with Eq. 10.

$$K(i,j,u/v) = \begin{cases} 1 & if \quad i/j = l \\ \frac{-\ln(1+|i/j-l|)}{\sum_{i/j\in[0,2l]}^{i/j\neq l}\ln(1+|i/j-l|)} & else \end{cases} \tag{10}$$

Then, the convolution is operated and *pscm* is constructed with Eq. 11.

$$pscm(x,y,u/v) = Conv\left(K, \Delta(u/v)_{vi} - \Delta(u/v)\right) + \Delta(u/v) \tag{11}$$

Inverse Pixel Size Control Map Construction. Similar to *pscm*, we define the inverse pixel size control map $pscm_{inv}$ to control the corresponding pixel size in the mapped g-buffer after inverse mapping. The inverse pixel size control map $pscm_{inv}(h,w)$ is a 2D map with the same size as the mapped g-buffer. Each element of the $pscm_{inv}$ has two components representing the vertical and horizontal sizes. For pixels of mapped g-buffer, we compute their size $\Delta(x/y)$ in the same way as Eq.6. We approximate the relationship between partial derivatives as $\frac{\partial u}{\partial x}\cdot\frac{\partial x}{\partial u} \approx \frac{\partial v}{\partial y}\cdot\frac{\partial y}{\partial v} \approx 1$. Then, the $pscm_{inv}(h,w)$ can be calculated based on Eq. 12.

$$pscm_{inv}(u,v,x/y) = \frac{\Delta(u/v)}{pscm(x,y,u/v)}\Delta(x/y) \tag{12}$$

3.4 Pixel Size Control Map guided Foveated Rendering

We propose a pixel size control map guided foveated rendering method. This method takes the g-buffer and the pixel size control map *pscm*, $pscm_{inv}$ as input and outputs the final full-resolution output fbuffer. First, we map the g-buffer to a mapped g-buffer with the guidance of *pscm*. Second, we render the reduced resolution buffer rbuffer with a pixel shader according to the mapped g-buffer. Third, we do the inverse mapping to transform the rbuffer to the fbuffer with the guidance of $pscm_{inv}$.

Mapping guided with *pscm*. The g-buffer in the deferred shading pipeline contains the world-space coordinates of objects, normals, depth, texture coordinates, and material-related information. In mapping, we transform the contents of the g-buffer with size $H \times W$ to the mapped g-buffer $h \times w$. For each pixel (x,y) on the g-buffer, the information of this pixel is stored in the pixel on the mapped g-buffer with coordinates (u,v) based on Eq. 13.

$$\begin{cases} u = \sum_{i=0}^{x-1} pscm(i,y,0)/\sqrt{\frac{\pi}{2m}\ln(1+mW/w_0)} \times w \\ v = \sum_{j=0}^{y-1} pscm(x,j,1)/\sqrt{\frac{\pi}{2m}\ln(1+mH/w_0)} \times h \end{cases} \tag{13}$$

Rendering. In the rendering pass, a pixel shader computes the direct and indirect lighting for each pixel using the information from the mapped g-buffer and renders the result to rbuffer.

Inverse Mapping guided with $pscm_{inv}$. Finally, the rendered result stored in rbuffer is mapped to fbuffer, which is the inverse of the above process. Specifically, for each pixel (u,v) on rbuffer, the color of that pixel is stored in the pixel with coordinates (x,y) on fbuffer, where (x,y) can be calculated in the same way as Eq. 13.

3.5 Temporal Coherent Refinement

When the user's gaze point, viewpoint, or scene changes, the visual features within the user's view will also change, which will result in an abrupt change in the visual importance map corresponding to that view. Since the pixel size control map is computed based on the visual importance map, excessive pixel size changes may result in visual flickering between frames of the final rendered results. There are two ways to mitigate this problem. One is to smooth the pixel size changes between frames by weighted averaging the visual importance maps of neighboring frames. The other approach is to detect the locations where the visual importance jumps between frames of the visual importance map and eliminate the excessive changes by adjusting the visual importance of these locations. Based on the above ideas, we propose a method to maintain the temporal coherence of the visual importance map, which enables smooth interframe changes of the pixel size control map and realizes real-time smooth foveated rendering in the presence of gaze point changes, viewpoint changes, or scene changes. We utilize Eq. 14 to generate the temporal coherent visual importance map $cvim^*$.

$$\begin{aligned} cvim^*(p_i) =& \gamma cvim_{i-1}(p_{i-1}) + (1-\gamma)cvim_i(p_i) \\ &+ \eta\,|cvim_i(p_i) - cvim_{i-1}(p_{i-1})| \end{aligned} \tag{14}$$

where γ and η can be obtained with a pilot user study. $cvim_i$ denotes the visual importance map of the current frame, $cvim_{i-1}$ denotes the visual importance map of the previous frame. p_i denotes the point on $cvim_i$ and p_{i-1} is the correspond point on $cvim_{i-1}$. After this, $civm^*$ is normalized with Eq. 2 in Section 3.2 and the foveated rendering is processed as in Section 3.3 and 3.4.

4 User Studies

We conduct two pilot user studies to determine the parameters of our method. The first one is used to estimate the parameters of the visual acuity model and the compression parameter. The second one is used to optimize the weight of the visual feature, kernel size of the pixel size control map construction, and the weight coefficient of temporal coherent refinement for the conservative visual importance map. We also conduct a user study to evaluate the visual perceptual quality of our method and the comparison methods.

4.1 Pilot User Study 1: estimation of w_0, m and t

We conducted the first pilot user study to empirically estimate the most suitable parameter values for acuity limit w_0, acuity slope m, and compression parameter $t = h/H = w/W$ in Section 3.2 that result in perceptually acceptable foveated rendering. The visual acuity model and compression parameter are scene-independent, thus we ignore the visual feature and the temporal coherence, i.e., we only use Eq. 5 and Eq. 7 for mapping and inverse mapping.

4.1.1 Pilot User Study Design

Conditions. For w_0, visual acuity of $6/6$ ($w_0 = 1'$) is frequently used as an estimate of average foveal acuity for adults [44]. However, most observers may have a binocular acuity superior to $6/6$, and Brian et al. [1] used $w_0 = 1/48^\circ = 1.25'$. We set w_0 ranging from 1.0 to 1.4 with a step 0.1. Many researches have been conducted to find the optimal m, and Brian et al. [1] reported optimal m of different compression parameters ranging from 1.32 to 1.65. We set m ranging from 1.3 to 1.7 with a step 0.1. For t, $t = 1$ denotes rendering with full-resolution. Meng et al. [2] reported when $t > 2.8$ the users found foveated rendering results significantly different from the full-resolution rendering results. We set t ranging from 1.2 to 2.8 with a step size of 0.2. Based on the steps of the 3 parameters, we get $5 \times 5 \times 9 = 225$ conditions in total.

Participants and Setup. 15 participants (10 males and 5 females, aged between 21-30) were recruited in this study, and all of them have had experiences in VR HMDs. We used an HTC Cosmos HMD with a Droolon aGlass to track the gaze point of the user. The HMD was connected to a PC workstation with a 3.8 GHz Intel(R) Core(TM) i7-10700KF CPU, 64 GB of memory, and an NVIDIA GeForce GTX 3080 Ti graphics card.

Task and Procedure. There are 2 outdoor scenes(City [45] and Bistro [46]) and 3 indoor scenes(Livingroom, Temple [47], and Bedroom) used in the experiment. Each scene used 5 acuity limit levels, 5 acuity slope levels, and 9 compression parameter levels for generating foveated rendering results. We asked each participant to participate in the two-alternative-force-choice experiment on 5 scenes with 225 conditions each, 1125 tests in total. For each scene, we presented the participant with two frames: the full-resolution rendering and the foveated rendering with t, w_0, m, where t, w_0, m are selected from the shuffled parameter array. The two frames were presented in a random order for 2 seconds each and were separated by a black-screen interval of 0.75 second. Then, we asked the participants to score the difference between the two frames they observed by pressing one of two buttons. The score $Score(scene, t, w_0, m)$ contains 2 confidence levels: 1 represents perceptually identical; 0 represents perceptually different.

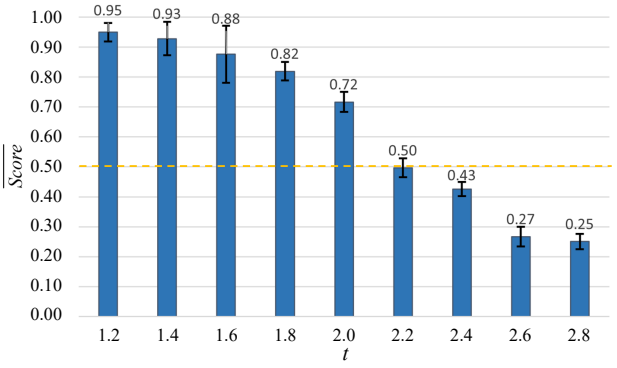


Fig. 4: The average values and standard deviations of $\overline{Score}$ for each t across all w_0, m and scenes, where the yellow dotted line indicates the score threshold of 0.5. The averaged score $\overline{Score}$ decreases with t as expected.

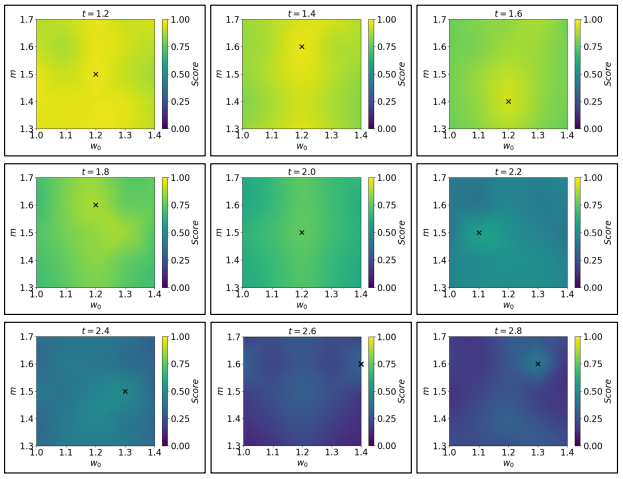


Fig. 5: The average values of $Score$ for each w_0, m, t across all scenes, where the black cross indicates the highest score. Yellow indicates high score while blue indicates low score.

4.1.2 Results and Discussion

We calculate the average score $\overline{Score}$ for each t across all w_0, m and scenes to estimate the overall optimal t that is applicable to general cases. Fig.4 gives average values and standard deviations of $\overline{Score}$ for each t. As shown in Fig.4, the $\overline{Score}$ is inversely related to t, which is reasonable. As t increases, the mapped g-buffer gets smaller, thus reducing the overall computing resources used in foveated rendering which leads to a decrease in the visual perceptual quality. To achieve visually acceptable results for foveated rendering, we use a threshold of 50% responses (yellow dotted line) considering foveated rendering to be visually indistinguishable from full-resolution rendering. To achieve the highest rendering acceleration, we look for the highest t that meets this

threshold. We therefore choose $t = 2$ as our desired compression ratio for foveated rendering.

We also calculate the average values of *Score* of all conditions and show the smoothed score *Score* of each t across the w_0 and m grid in Fig.5.

As shown in Fig.5, in general, the *Score* increases with w_0 ranging $1.0 - 1.2$. This is reasonable because w_0 denotes the computing resources allocated to the fovea. Since the resolution of the mapped g-buffer is reduced, the visual perceptual quality of the foveated rendering results is naturally lower than that of the full-resolution rendering. However, human perception is mainly allocated in the fovea region, so allocating more computational resources in the fovea region (bigger w_0) will improve the quality of visual perception of foveated rendering results. However, this does not mean that a larger w_0 is better, because too large w_0 will make the visual perceptual quality of the periphery region have a decrease. Besides, with the increase of m, *Score* decreases with m ranging $1.3 - 1.5$. This is reasonable because m represents the rate at which visual acuity declines. A larger m means that fewer computational resources are allocated to the periphery region, which reduces the quality of visual perception in the periphery region. Overall, w_0 and m together determine the allocation of computational resources in the fovea and the periphery regions, and the optimal w_0 and m are different under different t. With $t = 2$, the optimal acuity limit and acuity slope are set as $w_0 = 1.2$ and $m = 1.5$.

4.2 Pilot User Study 2: optimization of k, n, γ and η

We conduct the second pilot user study to take the visual feature into account with the scene-aware foveated rendering method and the temporal coherent refinement method. We optimize the weight k of the visual feature, kernel size n of the pixel size control map construction in Section 3.3 and the weight coefficient γ and η in Section 3.5, so that the foveated rendering results have the highest visual perceptual quality.

4.2.1 Pilot User Study Design

Conditions. Based on Pilot User Study 1, we set $t = 2, w_0 = 1.2, m = 1.5$ as the parameters for the foveated rendering. Then, we set the k ranging from 0.0 to 0.8 with a step size of 0.2, n ranging from 1 to 9 with a step size of 2, γ ranging from 0.0 to 0.4 with a step size of 0.1 and η ranging from 0.0 to 0.4 with a step size of 0.1. When $n = 1$, the visual feature can not affect the calculation of *pscm*, so the rendering results are the same for different k, γ, η, and there is one condition when $n = 1$. Based on the steps of the 4 parameters, we get $5 \times 4 \times 5 \times 5 + 1 = 501$ conditions in total. The participants and setup are the same as those in Pilot User Study 1.
Task and Procedure. The scenes used in this experiment are the same as those used in Pilot User Study 1. Before the experiment, participants were introduced to the scenes and how to explore them. For each participant, we randomly selected one scene and asked the participant to participate in the experiment with 501 conditions and 501 tests in total. The probability of each scene being selected was the same. In each test, participants were free to explore the scene for 10 seconds, exploring as many places as possible. At the end of each test, participants were required to rate the visual perceptual quality of the test and take a 5-second break. The visual perceptual quality score *Sco* consists of 5 confidence levels: 5 means that no artifacts were perceived at all, 4 means they perceived acceptable artifacts for a few very short moments, 3 means they perceived acceptable artifacts, 2 means they perceived noticeable artifacts, 1 means they perceived obvious artifacts.
Statistical analysis. We compared the values of different conditions. First, the normality of the data was assessed using the Shapiro-Wilk test. Then the comparison was performed with a repeated-measures ANOVA if the values showed a normal distribution. When values did not follow a normal distribution, the comparison was performed using a Wilcoxon signed-rank test. In addition to the p-value of the statistical test, we also estimate the size of the effect using Cohen's d. The d values are translated to qualitative effect size estimates of Huge ($d > 2.0$), Very Large ($2.0 > d > 1.2$), Large ($1.2 > d > 0.8$), Medium ($0.8 > d > 0.5$), Small ($0.5 > d > 0.2$), and Very Small ($0.2 > d > 0.01$).

4.2.2 Results and Discussion

We calculate the average values of *Sco* of all conditions. Fig.6 visualizes average values of *Sco* for all conditions except when $n = 1$. The color and size of each point in Fig.6 represent the value of *Sco*. Big size and yellow indicate high *Sco* while small size and blue indicate low *Sco*. We also calculate the average value, standard deviations of *Sco* when changing only one parameter while setting the others to their optimal values, calculate the statistical significance between different sets of parameters, and visualize them in Fig. 7.

When $n = 1$, the value of *Sco* is 3.62 ± 0.11. In other conditions in Pilot User Study 2, the value of *Sco* is greater than 3.62 (as shown in Fig. 6), which demonstrates that k, n, γ, η are useful, i.e., allocating more computing resources to pixels with the visual feature can improve the visual perceptual quality of the foveated rendering results. As shown in Fig. 6, the value of *Sco* is maximized to 4.4, when $k = 0.4$, $n = 5$, $\gamma = 0.3$ and $\eta = 0.2$. Thus, we choose this set of parameters as the optimal estimation.

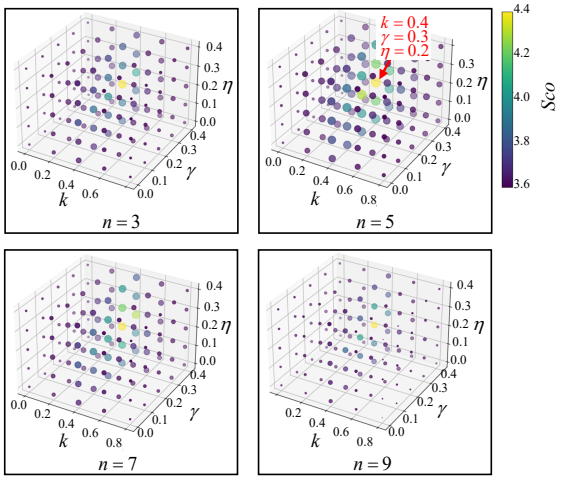


Fig. 6: The average values *Sco* for each k, n, γ, η across all scenes except when $n = 1$. The color and size of each point represent the value of *Sco*. The size of each point is *Sco* in the corresponding condition minus *Sco* when $n = 1$. Big size indicates high *Sco*. Yellow indicates high *Sco* while blue indicates low *Sco*.

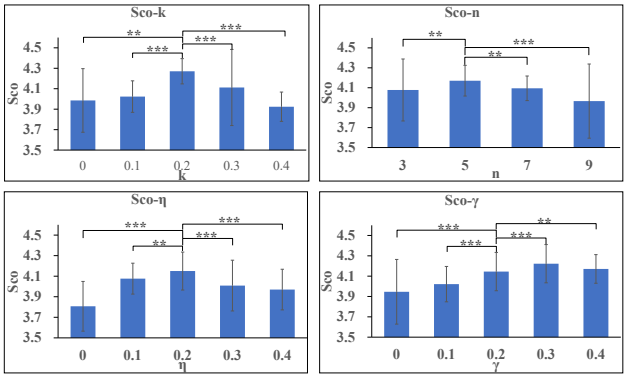


Fig. 7: The average value, standard deviations of *Sco* when changing only one parameter while setting the others to their optimal values. Error bars indicate standard deviation. Asterisks denote statistical significance between different sets of parameters.

As shown in Fig. 7, the average value of *Sco* when $k = 0.0$ is 3.98 which is bigger than 3.62. Because we approximate the shape of mapped pixels as a parallelogram and approximate the area of the mapped pixel. The calculated area is an approximation of the ideal visual acuity, and therefore fine-tuning the pixel size according to the visual acuity improves the visual perceptual quality. The average values of *Sco* when $k = 0.2, 0.4, 0.6, 0.8$ are $4.02, 4.27, 4.11, 3.92$, which are bigger than 3.98. Compared with conditions when $k = 0.4$, $p - values$ of $k = 0.0, 0.2, 0.6, 0.8$ are $0.002, < 0.001, < 0.001, < 0.001$, and effect sizes are from large to huge. There exists a significant difference across the multiple k, and the *Sco* is maximized when $k = 0.4$. The average values of *Sco* when $n = 3, 5, 7, 9$ are $4.07, 4.17, 4.09, 3.96$, which are bigger than 3.62. Compared with conditions when $n = 5$, $p - values$ of $n = 3, 7, 9$ are $0.005, 0.002, < 0.001$, and effect sizes are from large to very large. There exists a significant difference across the multiple n, and the *Sco* is maximized when $n = 5$. The average value of *Sco* when $\gamma = 0.0$ is 3.94. The average values of *Sco* when $\gamma = 0.1, 0.2, 0.3, 0.4$ are $4.02, 4.14, 4.22, 4.17$, which are bigger than 3.94. Compared with

conditions when $\gamma = 0.3$, $p-values$ of $\gamma = 0.0, 0.1, 0.2, 0.4$ are $< 0.001, < 0.001, < 0.001, 0.002$, and effect sizes are from large to huge. There exists a significant difference across the multiple γ, and the *Sco* is maximized when $\gamma = 0.3$. The average value of *Sco* when $\eta = 0.0$ is 3.81. The average values of *Sco* when $\eta = 0.1, 0.2, 0.3, 0.4$ are $4.07, 4.15, 4.01, 3.96$, which are bigger than 3.81. Compared with conditions when $\eta = 0.2$, $p-values$ of $\eta = 0.0, 0.1, 0.3, 0.4$ are $< 0.001, 0.001, < 0.001, < 0.001$, and effect sizes are from large to huge. There exists a significant difference across the multiple η, and the *Sco* is maximized when $\eta = 0.2$.

4.3 User Study

4.3.1 User Study Design

Hypotheses. With the same compression parameter t, our method has better visual perceptual quality than the comparison methods.
Conditions. We have 5 conditions, which are our method(Ours), Hyp method [1](Hyp), Log-Polar method [2](Log-Polar), Rectangular method [3](Rectangular), and Rectlinear method [4](Rectlinear). The parameters of the comparison methods are set to the optimal parameters reported in their papers. The participants and setup are the same as those in Pilot User Studies.
Task and Procedure. The scenes used in this experiment are the same as those used in Pilot User Studies. We asked each participant to participate in the experiment with 5 conditions in 5 scenes, and $5 \times 5 = 25$ tests in total. Before the experiment, we introduced users to scenes and how to explore them. For each participant, the order of tests was random. The task was to explore the scenes freely for 30 seconds in each test, exploring as many places as possible. At the end of each test, participants were required to rate the visual perceptual quality of the test and take a 10-second break. The visual perceptual quality score *Sco* contains 5 confidence levels which are the same as those in Pilot User Study 2. The statistical analysis is the same as that in Pilot User Study 2.

Table 1: Score comparison between our method and the previous methods.

Scene	Method	$Avg \pm std$	p	d	effect size
Livingroom	Ours	4.60 ± 0.15	-	-	-
	Hyp	4.25 ± 0.24	<0.001	1.23	very large
	Log-Polar	3.55 ± 0.22	<0.001	3.91	huge
	Rectangular	4.25 ± 0.21	<0.001	1.35	very large
	Rectlinear	3.86 ± 0.27	<0.001	2.41	huge
City	Ours	4.36 ± 0.29	-	-	-
	Hyp	3.48 ± 0.29	<0.001	2.16	huge
	Log-Polar	3.70 ± 0.32	<0.001	1.54	very large
	Rectangular	3.46 ± 0.23	<0.001	2.45	huge
	Rectlinear	3.98 ± 0.21	<0.001	1.07	large
Bistro	Ours	4.12 ± 0.11	-	-	-
	Hyp	3.72 ± 0.33	0.003	1.17	large
	Log-Polar	3.38 ± 0.40	<0.001	1.78	very large
	Rectangular	3.51 ± 0.39	<0.001	1.52	very large
	Rectlinear	3.77 ± 0.37	<0.001	1.68	very large
Temple	Ours	4.86 ± 0.16	-	-	-
	Hyp	4.12 ± 0.32	0.003	2.08	huge
	Log-Polar	3.32 ± 0.27	<0.001	4.82	huge
	Rectangular	4.05 ± 0.19	<0.001	3.22	huge
	Rectlinear	4.04 ± 0.33	<0.001	2.21	huge
Bedroom	Ours	4.61 ± 0.21	-	-	-
	Hyp	4.35 ± 0.22	0.004	0.84	large
	Log-Polar	3.86 ± 0.26	<0.001	2.24	huge
	Rectangular	4.19 ± 0.32	0.009	1.12	large
	Rectlinear	3.07 ± 0.49	<0.001	2.86	huge

4.3.2 Results and Discussion

As shown in Table 1, we calculate the average score for different scenes of all conditions, and use the p-value and Cohen's d to estimate the difference between two conditions. The results show that there is a significant difference between our method and other methods. Compared with the 4 previous methods, our method has the highest visual perceptual quality in all five scenes. This demonstrated that allocating more computational resources to locations with higher visual feature improves the visual perceptual quality of the foveated rendering results. City is an empty outdoor scene, our method can better reduce the allocated computational resources for non-important objects and then increase the computational resources for important objects, while comparison methods do not consider the visual features of the objects and allocate the computational resources only according to the visual acuity. Bistro is a complicated outdoor scene, although the region of the scene without visual feature is small, our method can still compress this region, so as to allocate more computational resources to regions with high visual feature. Temple is a scene with complex lighting changes, our method can better handle the light variation between frames and thus achieve better visual perceptual quality. Living room and bedroom are different indoor scenes, objects with high visual feature in the former are evenly distributed around the fovea region, whereas objects with high visual feature in the latter are concentrated on the left side of the fovea region. Our method gets the rendering results with high visual perceptual quality without any requirements on the distribution of objects with high visual feature in the scene.

5 Comparison Experiment

5.1 Implementation

The scenes and setup used in this experiment are the same as those used in the User Studies. The rendering methods used to generate results include the full-resolution rendering method (GT), our method, Hyp method [1], Log-Polar method [2], Rectangular method [3], and Rectlinear method [4]. The parameters of the comparison methods are set to the optimal parameters reported in their papers. GT is rendered with resolution 2000×2000. The compression parameter t of the foveated rendering methods is set as 2. The parameters of our method are set as $w_0 = 1.2, m = 1.5, k = 0.4, n = 5, \gamma = 0.3, \eta = 0.2$. The fovea region is defined with $e < 10^\circ$, which is the same as [1], the periphery region is defined with $e > 10^\circ$.

5.2 Quality

The quality of our results is compared with those of ground truth and the comparison methods in Fig.8. The first column of images shows the comparison between our method(lower-left) and the full-resolution rendering method (upper-right). The yellow circles on the image indicate fovea regions. The second column of images shows the close-ups of the rendered images for comparison (red, green, and purple rectangles).

Our results are closer to the ground truth results and generally preserve sharper details with the same compression ratio t. Some artifacts are shown in the rectangular regions rendered by the comparison methods. In City scene, the insurance company brand, bus stop sign, and fire hydrant rendered by our method are clearer, while the corresponding areas of comparison methods are blurred. In Bistro scene, our method renders the street lights, the menu, and the motorcycle more clearly, while the corresponding areas of comparison methods have visible jagged edges. In Temple scene, our method renders statues more clearly. For these statues, our rendering shows the shape of statues more clearly, and the highlight details of statues are also clearly rendered, while the other rendering results have artifacts of different sizes and shapes. In Bedroom scene, the teddy bears and toy cars rendered by our method are clearer. Our method renders the highlight areas on the front windshield of the black car closest to GT, while the results rendered by the other methods have significant artifacts. The numbers on the roof and the body of the red car rendered by our method are clearly visible, while the numbers in the results rendered by the other methods are jagged.

Table 2 shows the comparison of Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity(SSIM) of our method and comparison methods for the images in Fig. 8.

To distinguish the error in different regions, we quantify the image quality in 4 regions: the whole image (whole), the fovea region (fovea), the periphery region (periphery), and the salient region (salient). In Fig. 9, the yellowish color indicates the fovea region and the white color indicates the salient region.

Our method achieves larger PSNR, and larger SSIM in the whole image, fovea, periphery, and salient regions than those of comparison methods.

Fig. 10 illustrates the visualizations of the *vam*, vfm, *cvim* and the probability map (*pm*) used in [17] of the Temple Scene. We combine *vam* and vfm to generate *cvim* with a control parameter k. The parameter k controls the allocation of computing resources and the rendering quality of different regions. The probability map is constructed by a simple max function in [17] and cannot be used directly in our methods. The u and v channels of *pscm*, the rbuffer of our method (rbuffer_{our}), the rbuffer of the Log-Polar method (rbuffer_{LP}), the rbuffer of the Rectangular method (rbuffer_{RT}) and

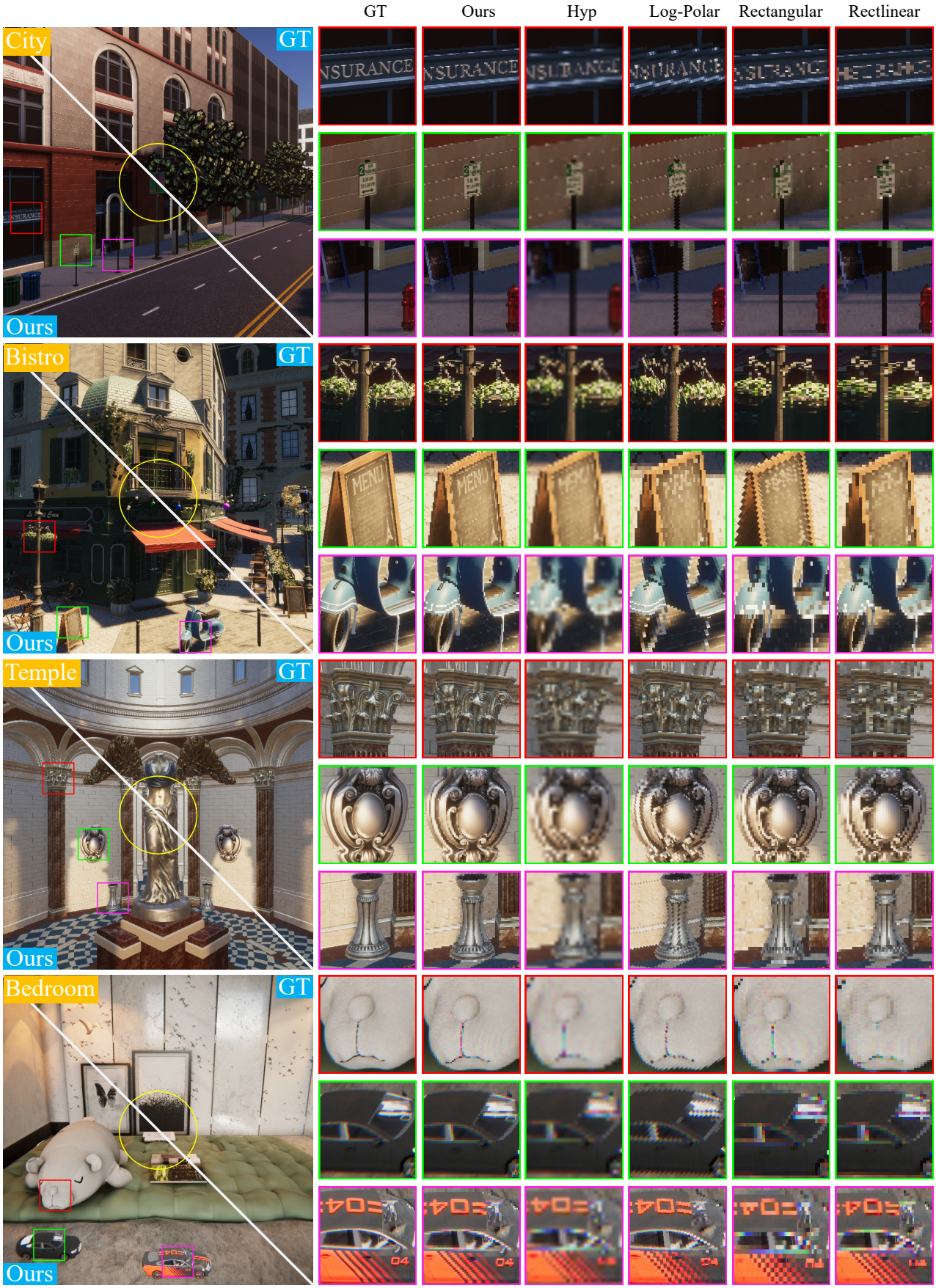
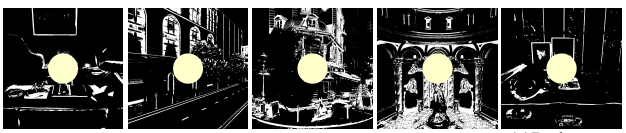


Fig. 8: Left: Comparison of the proposed foveated rendering (lower-left) and the full-resolution rendering (upper-right). Right: As illustrated in the close-ups of the rendered images, compared with Hyp method [1], Log-Polar method [2], Rectangular method [3], Rectlinear method [4], our results are closer to the results of the ground truth and generally preserves sharper details with the same compression ratio.



(a)Livingroom (b)City (c)Bistro (d)Temple (e)Bedroom

Fig. 9: Yellowish areas indicate the fovea region, and white areas indicate the salient region. The fovea region is defined with $e < 10^{\circ}$ which is the same as [1]. The salient region is defined with $cvim > 0.3$.

the rbuffer of the Rectlinear method (rbuffer$_{RL}$) are also visualized in Fig. 10.

Table 3 shows the comparison of the average interframe error (MSE', the average of all interframe errors) of GT, our method, and the comparison methods in different regions for the images in Fig.8, where 'NonInter' denotes setting $\gamma = 0$ and $\eta = 0$. 'NonInter' gets larger interframe errors than the comparison methods in the whole

Table 2: Image Quality (PSNR, SSIM) in the whole image, fovea region, periphery region, and salient region with different methods.

Region	whole					fovea					periphery					salient				
Method	Ours	Hyp	Log-Polar	Rect-angular	Rect-linear	Ours	Hyp	Log-Polar	Rect-angular	Rect-linear	Ours	Hyp	Log-Polar	Rect-angular	Rect-linear	Ours	Hyp	Log-Polar	Rect-angular	Rect-linear
Metric	PSNR																			
Livingroom	**27.84**	26.08	27.44	26.14	27.58	**36.13**	35.94	34.49	34.61	32.44	**27.59**	25.72	27.12	25.80	27.58	**27.87**	24.12	25.42	23.98	25.84
City	**25.92**	24.16	25.01	24.07	25.49	**32.40**	31.67	30.46	30.46	30.83	**25.01**	23.90	24.00	24.06	24.62	**23.47**	21.13	22.36	21.44	22.98
Bistro	**25.52**	24.31	23.25	23.60	24.93	**35.26**	35.20	33.36	33.90	33.77	**25.32**	23.98	22.96	23.36	24.72	**26.04**	22.25	23.01	21.77	23.66
Temple	**28.95**	28.32	28.27	28.32	28.80	**36.68**	36.33	35.95	36.38	35.19	**28.67**	27.94	27.98	28.02	28.62	**29.64**	26.33	27.31	26.98	28.14
Bedroom	**30.44**	30.00	30.05	29.30	30.15	**39.75**	39.55	37.04	36.65	38.28	**30.07**	28.65	27.24	28.97	28.81	**30.08**	24.11	24.45	22.91	24.72
Metric	SSIM																			
Livingroom	**0.688**	0.475	0.574	0.471	0.621	**0.986**	0.973	0.977	0.976	0.976	**0.662**	0.491	0.599	0.498	0.646	**0.630**	0.446	0.480	0.442	0.493
City	**0.679**	0.463	0.550	0.442	0.556	**0.995**	0.995	0.992	0.994	0.994	**0.625**	0.476	0.574	0.470	0.577	**0.614**	0.409	0.412	0.411	0.424
Bistro	**0.674**	0.471	0.485	0.448	0.557	**0.996**	0.995	0.994	0.993	0.992	**0.625**	0.483	0.507	0.469	0.577	**0.614**	0.464	0.486	0.457	0.490
Temple	**0.690**	0.530	0.530	0.540	0.600	**0.990**	0.980	0.970	0.980	0.970	**0.660**	0.550	0.550	0.570	0.610	**0.640**	0.540	0.490	0.460	0.590
Bedroom	**0.702**	0.578	0.648	0.669	0.674	**0.992**	0.992	0.991	0.990	0.992	**0.696**	0.587	0.558	0.580	0.662	**0.659**	0.512	0.511	0.480	0.543

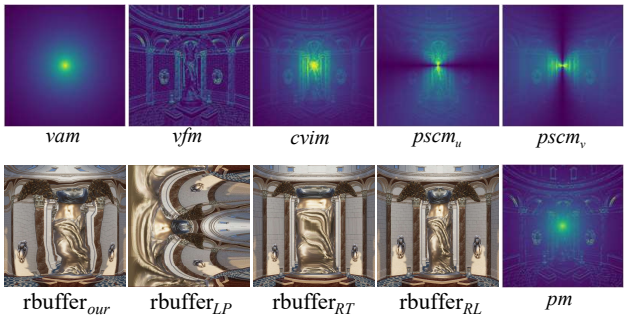


Fig. 10: Visualization of the vam, vfm, $cvim$, $pscm$, rbuffers of different mapping-based methods and the probability map (pm) used in [17]

image and the fovea region in Bistro scene. This is because, in Bistro scene, there are large variations in brightness and darkness in different areas. When the gaze point changes or the user's viewpoint changes, the visual feature also changes, which in turn leads to a large change in the visual importance and ultimately to image flickering. By applying the temporal coherent refinement, our method greatly reduces the interframe error and outperforms the comparison methods. This is because our method mitigates the problem of rendering flicker between frames by smoothing the pixel size changes between frames and eliminating the excessive changes by adjusting the visual importance.

In most cases, the interframe error of our method is close to that of GT. While in the whole image and the periphery region in the Bistro Scene, our method gets larger errors than GT. This is because there are too much visual feature in the periphery region of this scene, and these visual feature is densely distributed, thus additional computational resources allocated to each pixel with high visual feature are small, which finally leads to bigger interframe errors.

Table 3: Interframe Image Quality(MSE') in the whole image, fovea region, periphery region, and salient region with different methods.

Method	GT	Ours	NonInter	Hyp	Log-Polar	Rectangular	Rectilinear
				whole			
Livingroom	26.63	**29.69**	46.91	58.74	92.86	65.73	68.07
City	48.47	**91.62**	101.79	137.62	181.94	145.51	153.99
Bistro	29.38	**116.52**	159.58	195.75	174.40	128.87	125.36
Temple	15.59	**26.01**	32.29	53.70	56.43	52.08	49.55
Bedroom	17.88	**28.77**	33.00	35.46	73.80	54.61	62.78
				fovea			
Livingroom	18.21	**19.45**	21.04	20.06	28.90	28.90	25.68
City	40.89	**47.54**	52.27	54.73	100.52	90.41	75.74
Bistro	21.77	**21.80**	24.39	21.92	52.89	40.02	40.07
Temple	11.10	**11.76**	12.03	15.88	26.76	22.28	19.67
Bedroom	19.06	**22.55**	24.50	24.55	31.33	28.71	25.59
				peri			
Livingroom	20.56	**30.61**	49.57	93.45	97.06	68.29	71.17
City	59.11	**94.99**	103.29	199.56	189.38	150.64	161.14
Bistro	31.33	**125.58**	132.95	193.77	186.01	137.36	133.51
Temple	19.98	**31.19**	32.32	46.36	59.27	54.93	52.40
Bedroom	15.18	**38.23**	42.84	72.84	77.64	57.06	66.14
				salient			
Livingroom	42.33	**65.87**	128.23	275.22	376.63	267.35	269.19
City	129.29	**213.76**	216.72	454.20	622.71	468.83	508.91
Bistro	71.49	**124.83**	171.95	211.45	363.85	275.85	265.06
Temple	59.77	**67.76**	104.71	199.55	239.68	330.40	207.46
Bedroom	30.82	**78.12**	82.88	237.05	317.56	219.87	237.64

5.3 Performance

For the mapping-based foveated rendering framework, the theoretical speedup of rendering is proportional to $1/t^2$. We implement our method on a Nvidia GTX3080ti graphics card with $t = 2$ using the deferred shading pipeline. We report the rendering time of the segments and the total time between the full-resolution rendering, our method, and the comparison methods in Table 4. Our method achieves similar rendering acceleration compared with the previous methods, reduces the rendering time by 39.33%, and achieves a speedup of 1.65 with a 2000×2000 resolution.

Table 4: Performance(ms) of the full-resolution rendering, our method, and the comparison methods

Procedure	Timing(ms)					
	GT	Hyp	Rectangular	Rectlinear	Log-Polar	Ours
Depth Pass	1.34	1.35	1.33	1.32	1.32	1.32
Shadow Pass	5.87	7.83	6.01	6.12	5.46	5.67
Defer Pass	5.14	4.76	5.41	5.14	5.12	5.14
Skybox	0.08	0.08	0.08	0.08	0.08	0.08
cvim Gen	-	-	-	-	-	0.93
pscm Gen	-	-	-	-	-	0.91
Shading/Pass 1	25.78	12.28	7.69	7.99	7.99	7.75
Pass 2	-	-	0.97	0.97	0.97	1.39
Total	38.23	26.31	21.50	21.64	20.96	23.19
Speed Up	-	1.45×	1.78×	1.77×	1.82×	1.65×

6 Conclusion

We have proposed a scene-aware foveated rendering method. First, a conservative visual importance map is constructed by combining a visual feature map generated based on the scene and a visual acuity map generated based on the user's gaze point. A convolution-based construction method is used to construct the pixel size control map based on the conservative visual importance map. Next, the foveated rendering is executed with the guidance of the pixel size control map. At last, a temporal coherent refinement strategy is adopted to refine the temporal conservative visual importance map, which leads to smooth foveated rendering in real-time. Compared to existing state-of-the-art methods, our method achieves smaller MSE, larger PSNR, and larger SSIM in the fovea, periphery, salient regions, and the whole image with similar performance. User study also demonstrated that our method achieves the best visual perceptual quality.

Our method has several limitations. The first limitation is that our method has some scene-dependent parameters, such as weight of visual feature, kernel size, and weight coefficient, which in our implementation is determined using Pilot User Study 2 that employed multiple scenes. A more reasonable approach would be to compute them scene-by-scene, or even frame-by-frame. In future work, we will consider adaptive real-time computation methods for these parameters. The second limitation is that our method does not take into account the motion of the object and the interaction between the object and the users. Human visual perception of moving objects is more complex than that of stationary objects. Some rendering algorithms in recent years tend to allocate fewer computational resources to moving objects. Besides, when there are action-based tasks, such as interactive tasks, the attention distribution will be different from the ideal state and semantic-based features needed to be take into account. In the future, we intend to combine object motion information, user interacting information, low-level visual features and semantic-based features, allocating fewer computational resources to objects that move at high speeds and have simple surface textures, and more computational resources to the interacting objects. The third limitation is that for scenes with dense visual features in the periphery region, our method can control the allocation of computational resources in the fovea region and the salient region by weight of visual feature, but the quality improvement in the salient region has limited effect compared to other mapping-based methods. In future work, we intend to introduce deep neural networks, etc., for quality improvement.

Acknowledgments

This work is supported by the National Natural Science Foundation of China through Projects 61932003 and 62372026, Beijing Science and Technology Plan Project Z221100007722004, and the National Key R&D plan 2019YFC1521102.

REFERENCES

[1] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM transactions on Graphics (tOG)*, 31(6):1–10, 2012. 1, 2, 5, 7, 8

[2] Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018. 1, 2, 3, 5, 7, 8

[3] Jiannan Ye, Anqi Xie, Susmija Jabbireddy, Yunchuan Li, Xubo Yang, and Xiaoxu Meng. Rectangular mapping-based foveated rendering. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 756–764. IEEE, 2022. 1, 2, 3, 7, 8

[4] David Li, Ruofei Du, Adharsh Babu, Camelia D Brumar, and Amitabh Varshney. A log-rectilinear transformation for foveated 360-degree video streaming. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2638–2647, 2021. 1, 2, 3, 7, 8

[5] 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. 1, 2

[6] Masahiro Fujita and Takahiro Harada. Foveated real-time ray tracing for virtual reality headset. *Light Transport Entertainment Research*, 2014. 1, 2

[7] Martin Weier, Thorsten Roth, Ernst Kruijff, André Hinkenjann, Arsène Pérard-Gayot, Philipp Slusallek, and Yongmin Li. Foveated real-time ray tracing for head-mounted displays. In *Computer Graphics Forum*, volume 35, pages 289–298. Wiley Online Library, 2016. 1, 2

[8] Matias K Koskela, Kalle V Immonen, Timo T Viitanen, Pekka O Jääskeläinen, Joonas I Multanen, and Jarmo H Takala. Instantaneous foveated preview for progressive monte carlo rendering. *Computational Visual Media*, 4:267–276, 2018. 1, 2

[9] Youngwook Kim, Yunmin Ko, and Insung Ihm. Selective foveated ray tracing for head-mounted displays. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 413–421. IEEE, 2021. 1, 2

[10] Nicholas T Swafford, José A Iglesias-Guitian, Charalampos Koniaris, Bochang Moon, Darren Cosker, and Kenny Mitchell. User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception*, pages 7–14, 2016. 1, 2

[11] Fabio Policarpo and Manuel M Oliveira. Relief mapping of non-height-field surface details. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 55–62, 2006. 1, 2

[12] Xuehuai Shi, Lili Wang, Xiaoheng Wei, and Ling-Qi Yan. Foveated photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4183–4193, 2021. 1

[13] NVIDIA. Lens matched shading, July 2017. `http://developer.nvidia.com/orca/amazon-lumberyard-bistro`. 1

[14] Lei Yang, Dmitry Zhdan, Emmett Kilgariff, Eric B Lum, Yubo Zhang, Matthew Johnson, and Henrik Rydgård. Visually lossless content and motion adaptive shading in games. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(1):1–19, 2019. 1, 2

[15] NVIDIA. Multi-res shading, July 2017. `http://developer.nvidia.com/orca/amazon-lumberyard-bistro`. 1

[16] Christian Vater, Benjamin Wolfe, and Ruth Rosenholtz. Peripheral vision in real-world tasks: A systematic review. *Psychonomic bulletin & review*, 29(5):1531–1557, 2022. 1

[17] Michael Stengel, Steve Grogorick, Martin Eisemann, and Marcus Magnor. Adaptive image-space sampling for gaze-contingent real-time rendering. In *Computer Graphics Forum*, volume 35, pages 129–139. Wiley Online Library, 2016. 1, 2, 7, 9

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

[19] Lili Wang, Xuehuai Shi, and Yi Liu. Foveated rendering: A state-of-the-art survey. *Computational Visual Media*, 9(2):195–228, 2023. 2

[20] Martin Weier, Michael Stengel, Thorsten Roth, Piotr Didyk, Elmar Eisemann, Martin Eisemann, Steve Grogorick, André Hinkenjann, Ernst Kruijff, Marcus Magnor, et al. Perception-driven accelerated rendering. In *Computer Graphics Forum*, volume 36, pages 611–643. Wiley Online Library, 2017. 2

[21] Eric Turner, Haomiao Jiang, Damien Saint-Macary, and Behnam Bastani. Phase-aligned foveated rendering for virtual reality headsets. In *2018 IEEE conference on virtual reality and 3D user interfaces (VR)*, pages 1–2. IEEE, 2018. 2

[22] Karthik Vaidyanathan, Marco Salvi, Robert Toth, Theresa Foley, Tomas Akenine-Möller, Jim Nilsson, Jacob Munkberg, Jon Hasselgren, Masamichi Sugihara, Petrik Clarberg, et al. Coarse pixel shading. In *Proceedings of High Performance Graphics*, pages 9–18. 2014. 2

[23] ARAUJO Helder. An introduction to the log-polar mapping. In *II Workshop on Cibernetic Vision, December, 1996*, 1996. 2

[24] V Javier Traver and Alexandre Bernardino. A review of log-polar imaging for visual perception in robotics. *Robotics and Autonomous Systems*, 58(4):378–398, 2010. 2

[25] 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 (TOG)*, 38(4):1–14, 2019. 2

[26] Xuehuai Shi, Lili Wang, Jian Wu, Runze Fan, and Aimin Hao. Foveated stochastic lightcuts. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3684–3693, 2022. 2

[27] David R Walton, Rafael Kuffner Dos Anjos, Sebastian Friston, David Swapp, Kaan Akşit, Anthony Steed, and Tobias Ritschel. Beyond blur: Real-time ventral metamers for foveated rendering. *ACM Transactions on Graphics*, 40(4):1–14, 2021. 2

[28] Brooke Krajancich, Petr Kellnhofer, and Gordon Wetzstein. Towards attention–aware foveated rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–10, 2023. 2

[29] Thállys Lisboa, Horácio Macêdo, Thiago Porcino, Eder Oliveira, Daniela Trevisan, and Esteban Clua. Is foveated rendering perception affected by users' motion? In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1104–1112, 2023. 2

[30] Akshay Jindal, Krzysztof Wolski, Karol Myszkowski, and Rafał K Mantiuk. Perceptual model for adaptive local shading and refresh rate. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 2

[31] Gyorgy Denes, Akshay Jindal, Aliaksei Mikhailiuk, and Rafał K Mantiuk. A perceptual model of motion quality for rendering with adaptive refresh-rate and resolution. *ACM Transactions on Graphics (TOG)*, 39(4):133–1, 2020. 2

[32] Taimoor Tariq, Cara Tursun, and Piotr Didyk. Noise-based enhancement for foveated rendering. *ACM Trans. Graph.*, 41(4), jul 2022. 2

[33] Jiawei LI Piaopiao YU Jie GUO Mingqiang WEI Yanwen GUO Dayong REN, Zhengyi WU. Point attention network for point cloud semantic segmentation. *SCIENCE CHINA Information Sciences*, 65(9):192104–, 2022. 2

[34] Liang CHANG Ke LU Tongtong WU, Fuqing DUAN. Human-object interaction detection via interactive visual-semantic graph learning. *SCIENCE CHINA Information Sciences*, 65(6):160108–, 2022. 2

[35] Nianchen Deng, Zhenyi He, Jiannan Ye, Budmonde Duinkharjav, Praneeth Chakravarthula, Xubo Yang, and Qi Sun. Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3854–3864, 2022. 2

[36] David Bauer, Qi Wu, and Kwan-Liu Ma. Fovolnet: Fast volume rendering using foveated deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):515–525, 2022. 2

[37] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 2

[38] Hans Strasburger, Ingo Rentschler, and Martin Jüttner. Peripheral vision and pattern recognition: A review. *Journal of vision*, 11(5):13–13, 2011. 3, 4

[39] Anna Marzecová, Antonio Schettino, Andreas Widmann, Iria SanMiguel, Sonja A Kotz, and Erich Schröger. Attentional gain is modulated by probabilistic feature expectations in a spatial cueing task: Erp evidence. *Scientific Reports*, 8(1):54, 2018. 3

[40] Yixue Wang, James R. Miller, and Taosheng Liu. Suppression effects in feature-based attention. *Journal of vision*, 15 5:15, 2015. 3

[41] Sam Ling, Taosheng Liu, and Marisa Carrasco. How spatial and feature-based attention affect the gain and tuning of population responses. *Vision Research*, 49(10):1194–1204, 2009. Visual Attention: Psychophysics, electrophysiology and neuroimaging. 3

[42] M. Reddy. Perceptually optimized 3d graphics. *IEEE Computer Graphics and Applications*, 21(5):68–75, 2001. 3

[43] Eric L Schwartz. Anatomical and physiological correlates of visual computation from striate to infero-temporal cortex. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):257–271, 1984. 3

[44] Robert J Cooling. Dictionary of visual science. *The British Journal of Ophthalmology*, 74(8):511, 1990. 5

[45] Kate Anderson Nicholas Hull and Nir Benty. Nvidia emerald square, open research content archive (orca), July 2017. `http://developer.nvidia.com/orca/nvidia-emerald-square`. 5

[46] Amazon Lumberyard. Amazon lumberyard bistro, open research content archive (orca), July 2017. `http://developer.nvidia.com/orca/amazon-lumberyard-bistro`. 5

[47] Epic Games. Unreal engine sun temple, open research content archive (orca), October 2017. `http://developer.nvidia.com/orca/epic-games-sun-temple`. 5