Foveated Stochastic Lightcuts

Xuehuai Shi, Lili Wang, Jian Wu, Runze Fan, and Aimin Hao

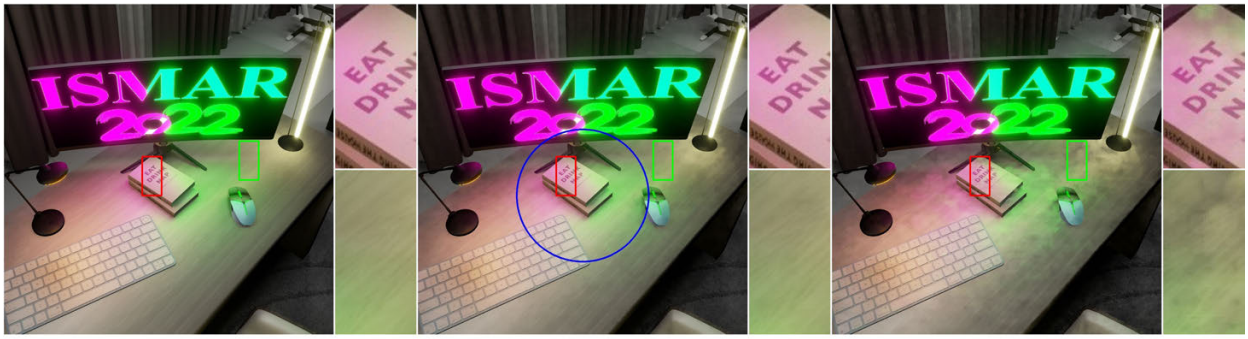

Fig. 1: Many-lights illumination effects rendered by stochastic lightcuts method with 2048 light samples per pixel (left), our method with 5.6 average light samples per pixel (middle), and stochastic lightcuts method using the same rendering time as our method (right) in *Study*. Compared with stochastic lightcuts (SL_e) using the same rendering time as our method, our method achieves 5.3× smaller mean squared error (*MSE*) in the foveal region (*Ours* vs SL_e, 2.47×10^{-2} vs 13.21×10^{-2}), and 2.3× smaller *MSE* in the peripheral region (*Ours* vs SL_e, 1.67×10^{-2} vs 3.86×10^{-2}).

Abstract—Foveated rendering provides an idea for accelerating rendering algorithms without sacrificing the perceived rendering quality in virtual reality applications. In this paper, we propose a foveated stochastic lightcuts method to render high-quality many-lights illumination effects in high perception-sensitive regions. First, we introduce a spatiotemporal-luminance based lightcuts generation method to generate lightcuts with different accuracy for different visual perception-sensitive regions. Then we propose a multi-resolution light samples selection method to select the light sample for each node in the lightcuts more efficiently. Our method supports full-dynamic scenes containing over 250k dynamic light sources and dynamic diffuse/specular/glossy objects. It provides frame rates up to 110fps for high-quality many-lights illumination effects in high perception-sensitive regions of the HVS in VR HMDs. Compared with the state-of-the-art stochastic lightcuts method using the same rendering time, our method achieves smaller mean squared errors in the fovea and periphery. We also conduct user studies to prove that the perceived quality of our method has a high visual similarity with the results of the ground truth rendered by using the stochastic lightcuts with 2048 light samples per pixel.

Index Terms—Virtual Reality, Foveated Rendering, Lightcuts, Many-lights Rendering

1 Introduction

Virtual reality (VR) technology gives us a new way to perceive the world. By wearing VR near-eye display devices, such as VR head-mounted displays (HMDs), users can immersively explore and interact with the virtual environment. Since the visual latency tolerance threshold of the Human Visual System (HVS) is around 13*ms* [23], excessive rendering latency in the HMD can lead to inconsistencies between what the user observes and the interaction, causing discomfort. Therefore, improving the performance of rendering algorithms is a critical factor in promoting the practicality of VR technology.

Foveated rendering is an accelerated rendering technology that assigns computing resources to different visual perception-sensitive regions of the HVS. To speed up rendering without sacrificing the perceived visual rendering quality, it allocates more computing resources to high perception-sensitive regions of the HVS and fewer computing resources to low perception-sensitive regions. In addition, recent studies show that high perception-sensitive regions in foveated rendering contain not only the foveal region but also some regions with high salient features [26] and high luminance-contrast features [29] in the periphery.

The existing foveated rendering algorithms can render dynamic scenes with multiple materials, but can not render scenes with a large number of dynamic light sources at high frame rates. Stochastic lightcuts method [41] can render the scene that contains a large number of light sources at interactive frame rates. However, this method fails to meet the visual latency tolerance threshold of HVS in HMDs, i.e., the rendering time per frame for both eyes is about 13*ms*. This is because in order to render high-quality illumination effects, the stochastic lightcuts method needs to generate accurate lightcuts for each pixel, and each node in the lightcuts needs to perform light tree traversal to select the light sample, which is time-consuming, especially for rendering in HMDs. To further improve the rendering performance of lightcuts, the foveated rendering framework can be used. However, it brings two challenges. The first one is how to generate lightcuts with different accuracy for different visual perception-sensitive regions without sacrificing the perceived rendering quality. The second one is how to select the light sample for each node in the lightcuts more efficiently without performing the light tree traversal for each node.

In this paper, we propose a foveated stochastic lightcuts method to address these two challenges. For the first one, we propose a spatiotemporal-luminance based lightcuts generation method. Our

- *Lili Wang is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China; Peng Cheng Laboratory, Shengzhen, China; and Beijing Advanced Innovation Center for Biomedical Engineering, Beihang University, Beijing, China. Lili Wang is the corresponding author. E-mail: wanglily@buaa.edu.cn.*
- *Xuehuai Shi, Jian Wu, Runze Fan, and Aimin Hao is with State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China. E-mail: shixuehuaireal@buaa.edu.cn, lanayawj@buaa.edu.cn, BY2106131@buaa.edu.cn, ham@buaa.edu.cn.*

method first extracts the foveated spatiotemporal feature map (FSFM) and the foveated luminance feature map (FLFM) according to the current view. Then it uses a 3D Gaussian filter to simulate the reduction of the spatiotemporal and luminance features for each pixel without perceived loss, and constructs a spatiotemporal-luminance based foveated perceptual sensitivity map (FPSM) to store the result of each pixel's standard deviation of the 3D Gaussian filter. After that, it generates the lightcuts with different accuracy for each pixel based on the corresponding value in the FPSM. To solve the second problem, we introduce a multi-resolution light samples selection method, which uses the low-resolution light samples selected in low-resolution rendering and chooses new high-resolution light samples to illuminate the scene according to the FPSM.

We compare the monocular images rendered by stochastic lightcuts with 2048 light samples per pixel, our method, and stochastic lightcuts (SL_e) which uses the same rendering time as our method. The result shows that our method achieves 1.9-5.3$\times$ smaller mean squared error (MSE) in the foveal region, and 1.2-2.3$\times$ smaller MSE in the peripheral region compared with SL_e. Our method achieves 53-110fps for both eyes with an HTC Cosmos and supports full-dynamic scenes with diffuse/specular/glossy objects and over 250k dynamic light sources. We also conduct user studies to prove that the perceived quality of our method has a high visual similarity with the result of the ground truth rendered by using the stochastic lightcuts with 2048 light samples per pixel. We refer readers to our supplementary video. Figure 1 shows the comparison in $Study$. Our method shows better illumination effects than SL_e in both the foveal and peripheral regions. The region in the blue circle is the foveal region. Details in rectangular regions are magnified to the right of each rendered image, the magnification of the red rectangular region in the foveal region is placed in the upper right, and the magnification of the green rectangular region in the peripheral region is placed in the lower right. In the red rectangular region, the green and red lighting from the screen onto the book is not smooth in the result of SL_e. In the green rectangular region, the illumination effects from the yellow light that shade on the desktop have many noisy points in the result of SL_e.

In summary, the contributions of our method are as follows:

- A foveated stochastic lightcuts method to render high-quality illumination in high perception-sensitive regions at high frame rates in HMDs, supporting full-dynamic scenes with over 250k dynamic light sources and diffuse/specular/glossy objects;
- A spatiotemporal-luminance based lightcuts generation method, which generates lightcuts with different accuracy for different visual perception-sensitive regions without sacrificing the perceived rendering quality;
- A multi-resolution light samples selection method, which selects the light sample for each node in the lightcuts more efficiently.

2 Related Work

In this section, we first introduce the prior work of foveated 3D rendering (FR) in recent years, then discuss the lightcuts generation methods, and discuss the light samples selection methods.

2.1 Foveated 3D Rendering

Guenter et al. [12] introduced a rasterization-based foveated rendering system to improve rasterization's rendering performance, demonstrating that users cannot perceive the degradation of rendering quality from foveated rendering in this system by very detailed perceptual experiments. It is an essential milestone in FR, and research in FR after this work tended to improve the rendering performance without visual perceived loss. Research in FR can be grouped into several categories according to the acceleration strategies: multi-spatial resolution, level of details (LOD), multi-color resolution, and multi-illumination resolution.

Multi-spatial resolution based FR reduces the pixel density in the output image from the high visual perception-sensitive region to the low visual perception-sensitive region of the HVS. Guenter et al. [12] generated the 3-layer image with different resolutions in one rasterization pipeline and composited them to generate the final foveated rendering result. Patney et al. [21] designed a foveated rendering system based on coarse pixel shading (CPS), which can reduce the number of shadings by up to 70%, and then introduced a novel anti-aliasing algorithm. This anti-aliasing algorithm can help recover the details of the peripheral region that are resolvable by human eyes but are degraded by filtering. As CPS pipelines require adaptive shading features which are not yet commonly available on commodity GPUs, Meng et al. [17] presented a simple two-pass kernel foveated rendering pipeline that maps well onto modern GPUs. Stengel et al. [26] introduced a foveated sampling method for rasterization, and then integrated the sampling method into the deferred shading pipeline. Turner et al. [28] aligned the rendered pixel grid to the virtual scene content during rasterization and upsampling, which reduced the detectability of motion artifacts in the periphery without complex interpolation or anti-aliasing algorithms. Friston et al. [11] presented a rasterization pipeline that achieved the foveated rendering in one rasterization pass with per-fragment ray-casting. Meng et al. [16] accelerated the foveated rendering in HMDs with more aggressive foveation based on the theory of ocular dominance. Bastani et al. [2] rendered an intermediary image of the 3D scene in the intermediary compressed space and unwarped the intermediary image to generate the foveated image. Franke et al. [10] presented a foveated rendering method that comprised recycling pixels in the periphery by spatiotemporally reprojecting them from the previous frames to accelerate rendering performance.

LOD-based FR adapts the complexity of 3D models to different visual perception-sensitive regions of the HVS, which renders the high visual perception-sensitive region with the high-quality geometry and the low visual perception-sensitive region with the simplified geometry. Recent research on LOD-based FR focused on designing user studies to optimize or select the various parameters involved in the previous methods or refine the previous methods instead of proposing new methods. Swafford et al. [27] proposed practical rules for LOD-based FR methods to achieve significant performance gains with user studies and the newly proposed rendering quality metrics. Lindeberg et al. [15] proposed a gaze-contingent depth of field tessellation that applies tessellation to all objects within the focal plane, gradually decreasing tessellation levels as applied blur increases. Zheng et al. [43] adaptively adjusted tessellation levels and culling region based on visual sensitivity. Young et al. [39] adjusted the size and shape of the foveal region for correcting the gaze tracing error or state parameters and combined this technique with LOD to render foveated images. Stafford et al. [25] selectively filtered the images in the peripheral region to reduce visual artifacts due to contrast resulting from the lower LOD before compositing foveated images for presentation.

Multi-color resolution based FR degrades the chromatic fidelity in the low visual perception-sensitive region to accelerate the rendering performance. Duchowski et al. [8] proposed a foveated chromatic degradation framework, which constructed the color degradation mapping for a gaze-contingent display to evaluate the spatio-chromatic peripheral sensitivity. Multi-illumination resolution based FR reduces the quality of illumination in the low visual perception-sensitive region to simplify the work per pixel from the high visual perception-sensitive region to the low visual perception-sensitive region. Stengel et al. [26] presented a luminance map to adjust the sampling probability in the peripheral region to obtain the shading samples that can effectively shade the important features of the image. Tursun et al. [29] proposed a luminance-contrast-aware foveated rendering technique that improves the computational savings of foveated rendering by analyzing the local luminance contrast of the image. Wang et al. [34] proposed the foveated instant radiosity method that casts more VPLs to illuminate the foveal region so that more accurate global illumination effects in the foveal region and less accurate global illumination in the peripheral region can be rendered. Yang et al. [38] improved the method proposed by Wang et al. [34] and created a CMF-based perceptual probability map to manage virtual point lights more accurately to further improve the rendering quality in the fovea. Shi et al. [24] adopted the photon mapping method

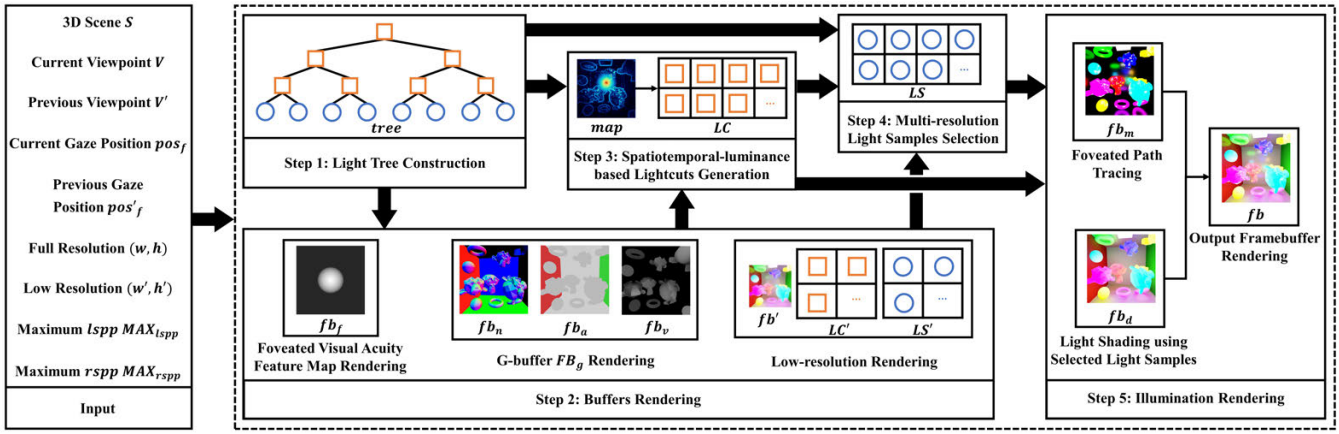

Fig. 2: Pipeline of Our Method

to foveated rendering framework, which can render high-quality global illumination effects in the foveal region at interactive frame rates for the scenes that include diffuse, specular, glossy, and transparent materials.

Since the high visual perception-sensitive region in the HVS includes not only the foveal region but also some regions with high salient features [26] and high luminance-contrast features [29] in the periphery. The above multi-illumination resolution based foveated rendering methods did not consider performing high-quality illumination rendering in the high perception-sensitive region in the periphery, which can further improve the quality of foveated rendering. We propose a foveated stochastic lightcuts method, which renders high-quality many-lights illumination in high perception-sensitive regions at high frame rates.

2.2 Lightcuts Generation Methods

Walter et al. [32] firstly proposed the general lightcuts for many-lights rendering. To improve the quality and accelerate rendering, researchers worked on lightcuts generation methods to generate lightcuts based on various evaluation criteria.

Walter et al. [31] introduced the multi-dimensional lightcuts method, which generates the multi-dimensional lightcuts for each pixel, and each dimension of the multi-dimensional lightcuts is used to achieve different rendering effects, such as temporal blur, depth of field, volumetric effects, and anti-aliasing, etc. They also [33] extended the multi-dimensional lightcuts method, and proposed the bidirectional lightcuts method that uses recursive eye tracing to connect more virtual point lights for generating lightcuts, and the new path weighting strategy to support rendering more materials, such as glossy, volumetric, and subsurface materials. Davidovič et al. [6] progressively generated the lightcuts to converge the illumination shaded by lightcuts to the correct solution with a bounded memory footprint. Wang et al. [35] used an out-of-core GPU implementation that generates lightcuts more efficiently for rendering large scenes. Yuksel et al. [42] introduced the lighting grid hierarchy method that generates multiple illumination representations of the scene with different resolutions, and used an evaluation criterion combining distance and resolution to generate lightcuts for rendering explosions with self-illumination. Yuksel [40, 41] removed the representative lights of all nodes in the traditional lightcuts [32], and randomly picked a light source in the leaf node within the given subtree using an importance sampling scheme to evaluate the node in the step of lightcuts generation. Compared with the traditional lightcuts, this method can perform many-lights rendering more efficiently and temporally stably.

However, the above methods did not consider combining the visual perceptual features of the HVS with the lightcuts generation. To accelerate the performance of illumination rendering with lightcuts without visual perceived loss, we propose a spatiotemporal-luminance based lightcuts generation method that generates lightcuts with different accuracy in different perception-sensitive regions.

2.3 Light Samples Selection Methods

Generating lightcuts by evaluating the importance criteria is not accurate enough, especially when the lightcuts contain nodes near the root. In order to improve the rendering quality, light samples selection for generated lightcuts is exploited for lightcuts methods.

Conty et al. [5] proposed the adaptive splitting heuristic method, which uses stochastic traversal from nodes in each lightcuts to leaves to select the actual lights, and determines the splitting during the traversal by a variance based on intensity and distance. Moreau et al. [18] extended this method with a two-level hierarchy for fast updates in dynamic scenes. Yuksel [40, 41] proposed the hierarchical importance sampling method to select the light for each node in the lightcuts, and introduced a new node probability formulation based on the geometry, material, intensity, and distance terms to obtain better temporal stability. However, if the shading point and the node are in the same bounding box, the distance term part of the node probability formulation and the expected illumination of the node will change from a correct positive relationship to an inverse relationship, thus reducing the sampling quality. Lin et al. [14] proposed a new distance term formulation that can mitigate this weakness and thus improve the sampling quality.

To further improve the performance of the light samples selection, we propose a multi-resolution light samples selection method. It uses the low-resolution light samples selected in the low-resolution rendering step and chooses new high-resolution light samples to illuminate the scene according to the FPSM.

3 Foveated Stochastic Lightcuts

In order to improve the performance of rendering scenes with many dynamic lights for VR applications without sacrificing the perceived rendering quality, we propose a foveated stochastic lightcuts method. Stochastic lightcuts [41] is an important algorithm for many-lights rendering, but it is not efficient enough for VR applications. Our method differs from the stochastic lightcuts method in two aspects: one is that our method has lightcuts with different accuracy for different perception-sensitive regions to improve the performance without sacrificing the perceived rendering quality, while the stochastic lightcuts method has lightcuts with the same accuracy for all regions; the other is that our method selects the light sample for each node in the lightcuts more efficiently without performing the light tree traversal for each node, while the stochastic lightcuts method needs to traverse the light tree for all nodes in the lightcuts.

Figure 2 visualizes the pipeline of our method. There are five steps in our method. Step 1 is the light tree construction, which constructs the light tree for the scene. Step 2 is the buffers rendering, which renders the buffers that are used in the next steps. Step 3 is the spatiotemporal-luminance based lightcuts generation, which generates lightcuts with different accuracy for different regions. Step 4 is the multi-resolution light samples selection, which selects light samples for rendering the diffuse illumination. Step 5 is the illumination rendering, which renders the diffuse, glossy and specular illumination effects for the current frame and outputs the final rendering result. For demonstrating the details of the steps, we give Algorithm 1.

The inputs of Algorithm 1 are the 3D scene S, the viewpoint of the current frame V, the viewpoint of the previous frame V', the gaze position of the current frame pos_f and the previous frame pos'_f, the

full resolution of the output framebuffer (w,h), the low-resolution (w',h'), the maximum number of light samples per pixel MAX_{lspp}, the maximum number of ray samples per pixel MAX_{rspp}. The output of Algorithm 1 is the framebuffer fb.

In Algorithm 1, we first use the perfect binary light tree construction method proposed in Lin et al. [14] to construct the light tree *tree* to represent the illumination in the 3D scene S (line 2). Then we use the visual acuity fall-off function proposed by Stengel et al. [26] to construct the foveated visual acuity feature map fb_f (line 3). We use the method proposed in Stengel et al. [26] to construct the spatiotemporal feature map from the G-buffer FB_g based on S, the viewpoint of the current frame V, the viewpoint of the previous frame V', and the full resolution of the output framebuffer (w,h) (line 4). FB_g includes the normal buffer fb_n, the albedo buffer fb_a, and the velocity buffer fb_v of the current frame. Each pixel in fb_n stores the normal of the corresponding pixel in the current framebuffer fb, so as to fb_a stores the albedo, and fb_v stores the velocity of the pixel's hitpoint moving through space. Our method also needs the luminance feature to guide lightcuts generation and light samples selection. We find that extracting the luminance feature from the low-resolution framebuffer is effective and efficient, and the low-resolution lightcuts and light samples can be reused in the light samples selection. Thus we perform the low-resolution rendering by directly using the method proposed by Yuksel et al. [41] to generate the low-resolution framebuffer fb', the low-resolution lightcuts set LC', and the low-resolution light samples set LS' based on S, *tree*, V, the low-resolution (w',h') (line 5). (w',h') is set as $(\frac{w}{8},\frac{h}{8})$ in our method.

Algorithm 1: Foveated Stochastic Lightcuts

input : 3D scene S, viewpoint of current frame V, viewpoint of previous frame V', gaze position of current frame pos_f, gaze position of previous frame pos'_f, full resolution of the output framebuffer (w,h), low resolution (w',h'), maximum number of light samples per pixel MAX_{lspp}, the maximum number of ray samples per pixel MAX_{rspp}

output : output framebuffer fb

1 **for** *each frame* **do**
2 // Light Tree Construction
3 $tree \leftarrow constructLightTree(S)$
4 // Buffers Rendering
5 $fb_f \leftarrow FBufferRender(V, V', pos_f, pos'_f, w, h)$
6 $FB_g \leftarrow GBufferRender(S, V, V', w, h)$
7 $fb', LC', LS' \leftarrow lowResRender(S, tree, V, w', h')$
8 // Spatiotemporal-luminance based Lightcuts Generation
9 $map \leftarrow genPerMap(S, FB_g, fb_f, fb', w, h)$
10 $LC \leftarrow genLightcuts(S, tree, map, MAX_{lspp}, V, w, h)$
11 // Multi-resolution Light Samples Selection
12 $LS \leftarrow selectLights(S, tree, map, LC', LS', LC, V, w, h)$
13 // Illumination Rendering
14 $fb, fb_d, fb_m \leftarrow initFrameBuffer(w, h)$
15 $fb_d \leftarrow lightShading(S, LS, V, w, h)$
16 **if** *hasMetallicObj(S)* **then**
17 $fb_m \leftarrow FoveatedPT(S, map, MAX_{rspp}, V, w, h)$
18 **end**
19 $fb \leftarrow fb_d + fb_m$
20 **end**

Then we perform the spatiotemporal-luminance based lightcuts generation method to construct the FPSM *map* and generate the lightcuts for each pixel with different accuracy (lines 6-7), and the details are in Section 3.1. After that, we perform the multi-resolution light samples selection method to select light samples for all lightcuts in LC and stores them in the light samples set LS based on S, *tree*, *map*, LC', LS', V, and (w,h) (line 8), and the details are in Section 3.2.

Finally, we perform the rendering step to generate the output framebuffer fb for the current frame (lines 9-13). Firstly, we initialize fb, the diffuse framebuffer fb_d, and the metallic framebuffer fb_m as empty based on (w,h) (line 9). Then we perform the traditional light shading method [32] to shade each pixel's diffuse illumination and store the result in fb_d based on S, LS, V, and (w,h) (line 10). If there are glossy or specular objects in S, we implement the foveated path tracing to render the glossy reflection effects of each pixel and store the result in fb_m based on S, *map*, the maximum number of ray samples per pixel MAX_{rspp}, V, and (w,h) (lines 11-12). The foveated path tracing is the one-bounced path tracing [13] with the guidance of *map*. Specifically, for each pixel px in fb_m, the ray samples per pixel ($rspp$) of px implemented in the foveated path tracing is $MAX_{rspp} \cdot map[px]$. The value of each pixel in fb is obtained by adding the value of the corresponding pixel in fb_d to the value of the corresponding pixel in fb_m (line 13).

3.1 Spatiotemporal-luminance based Lightcuts Generation

The motivation of the spatiotemporal-luminance based lightcuts generation method is to generate the lightcuts with different accuracy for different visual perception-sensitive regions according to the guidance of the spatiotemporal and luminance sensitivity of the HVS. Using the lightcuts with low accuracy for rendering in the low visual perception-sensitive region can improve the rendering performance without sacrificing the perceived rendering quality. Therefore, our method is different from the stochastic lightcuts method, which uses the lightcuts with the same accuracy to render all regions.

Using the HVS to guide lightcuts generation directly will lead to low-quality rendering in high perception-sensitive regions in the periphery of the HVS, which reduces the perceived rendering quality of users. Our spatiotemporal-luminance based lightcuts generation method identifies different perception-sensitive regions of the HVS, and then generates the lightcuts with high accuracy for high perception-sensitive regions to improve the perceived rendering quality of users. Our spatiotemporal-luminance based lightcuts generation method has two steps, the first step is to construct the FPSM, and the second step is to generate the lightcuts according to the FPSM. In the step of the FPSM construction, we first extract FSFM and FLFM from the G-buffer and the low-resolution framebuffer. Then we use a 3D Gaussian filter to merge FSFM and FLFM, and simulate the reduction of the merged feature value within the maximum imperceptibility threshold α. Finally, the standard deviation of the 3D Gaussian filter is used to calculate the value in the FPSM. In the lightcuts generation step, for each pixel in the output framebuffer, we multiply the corresponding pixel value in the FPSM by a constant value to get the limited number of nodes required for this pixel's lightcuts. Then we use the criterion introduced in [41] to approximate lightcuts with the limited number of nodes.

The adaptive foveated sampling method proposed in Stengel et al. [26] can efficiently extract the spatiotemporal feature based on the integration of the buffers in G-buffer. Thus, we use the adaptive foveated sampling method [26] to extract FSFM, but we don't use it to guide foveated rendering directly as Stengel et al. [26] did. Since HVS not only has the spatiotemporal sensitivity but also has the luminance sensitivity, we further extract FLFM based on the luminance of the current scene. We combine FSFM and FLFM to construct FPSM, which is used to guide the lightcuts generation.

After obtaining the spatiotemporal and luminance features, a straightforward method is to use the maximum value between spatiotemporal and luminance features to guide the generation of lightcuts with different accuracy. However, this may lead to the generation of high-accuracy lightcuts in the regions with very low spatiotemporal sensitivity but very high luminance sensitivity or very high spatiotemporal sensitivity but very low luminance sensitivity in the periphery, which may reduce the rendering performance. We use the 3D Gaussian filter to simulate the reduction of the foveated spatiotemporal and luminance features, which brings two advantages. The first one is that the 3D Gaussian filter can naturally merge the features in the first and second dimensions to generate the filtered feature in the third dimension. So we use the 3D Gaussian filter to merge the foveated spatiotemporal and luminance features and generate the filtered spatiotemporal-luminance feature. The second one is that the standard deviation of a Gaussian

filter means the difference between the filtered feature and the original features, i.e., the degree of the original features' reduction. So we seek a 3D Gaussian filter with the maximum standard deviation ρ to simulate the maximum imperceptible reduction of the features on each pixel within the maximum imperceptibility threshold α. Then, ρ of the 3D Gaussian filter of each pixel is used to construct FPSM. FPSM describes the maximum reduction of the foveated spatiotemporal and luminance features that are acceptable to HVS on each pixel. For each pixel, more reduction of the features leads to more aggressive rendering quality reduction, so the lower precise lightcuts can be used to shade this pixel.

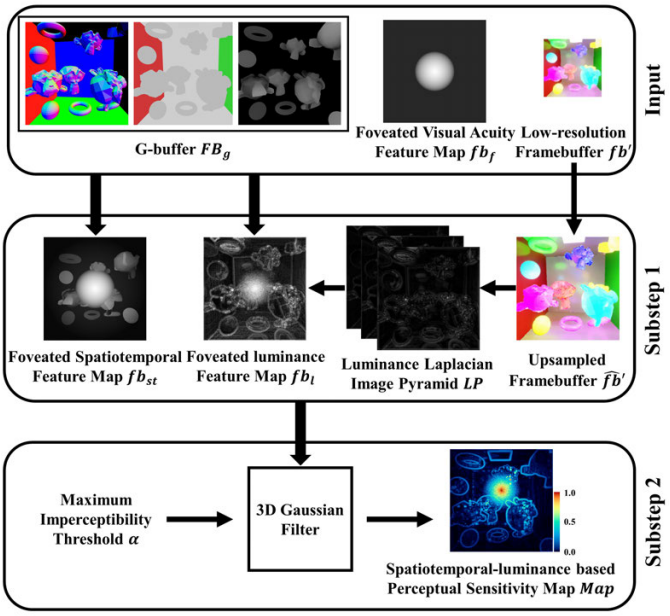

Fig. 3: Pipeline of FPSM Construction

Figure 3 describes the pipeline of FPSM construction. It inputs the G-buffer FB_g, the foveated visual acuity feature map fb_f, and the low-resolution framebuffer fb' of the current frame, outputs the FPSM Map. FB_g includes the normal buffer fb_n, the albedo buffer fb_a, and the velocity buffer fb_v of the current frame. Each pixel in the foveated visual acuity feature map fb_f stores the visual acuity value of the eccentricity corresponding to this pixel on the visual acuity fall-off function proposed by Stengel et al. [26]. There are two substeps in the FPSM construction.

In substep 1, we extract two foveated feature maps: FSFM fb_{st} and FLFM fb_l. For each pixel p, we use Equation 1 introduced in [26] to extract $fb_{st}(p)$. Because with Equation 1 we can extract the spatiotemporal feature of p efficiently by integrating the buffers in G-buffer, and combining it with the foveated visual acuity feature map directly. The feature value of each pixel p in fb_{st} is calculated by the following equation:

$$fb_{st}(p) = max(fb_f(p), \nabla fb_n(p), \nabla fb_a(p), \nabla fb_v(p)) \tag{1}$$

where $fb_f(p)$ is the visual acuity feature map that gives the visual acuity of p, $\nabla fb_n(p)$ calculates the first order derivative of p in fb_n by using Sobel filter [30] with the kernel size 3×3, $\nabla fb_a(p)$ calculates the first order derivative of p in fb_a, and $\nabla fb_v(p)$ calculates the first order derivative of p in fb_v. Since the optimized foveated acuity fall-off model proposed in [26] makes it more suitable for VR HMDs with wide field-of-view, we use the model to get the foveated visual acuity feature map fb_f. For each pixel p in fb_f, the calculation of $fb_f(p)$ is shown in Equation 2. [26] 2.

$$\begin{cases} fb_f(p) = clamp(f_f(p, pos_f), 0.02, 1) \\ f_f(p, pos_f) = 1.1715 - 2.45 \cdot e(p, pos_f) \end{cases} \tag{2}$$

where pos_f is the gaze position, $clamp(f_f(p, pos_f), 0.02, 1)$ clamps the value of visual acuity in the range $[0.02, 1]$, and $e(p, pos_f)$ calculates p's eccentricity based on the position of p and pos_f. Equation 3 introduced in [22] connects the luminance of an image with the perceptual sensitivity of the HVS, and effectively simulates the threshold of the perceptual sensitivity under different luminance in a nonlinear way. Thus, for each pixel p, we calculate the value of p in fb_l by Equation 3. [22]

$$fb_l(p) = max(fb_f(p), \frac{LP_1(p)}{LP_3(p) + \varepsilon}) \tag{3}$$

We first use the bicubic upsampling method [7] to upsample the low-resolution framebuffer fb' to produce the upsampled low-resolution framebuffer $\hat{fb'}$ of size (w, h). Then we build a 3-level luminance Laplacian image pyramid LP [3] for $\hat{fb'}$. $LP_1(p)$ denotes the value of p at the 1st level of the Laplacian image pyramid, and $LP_3(p)$ denotes the value of p at the 3rd level. ε is used to prevent mathematical singularities in the regions with low luminance [29], and we set ε to 0.01.

In substep 2, for each pixel p, we seek a 3D Gaussian filter with maximum standard deviation ρ to merge the two foveated feature maps fb_{st} and fm_{lt}, and simulate the maximum reduction of the feature values on each pixel within the maximum imperceptibility threshold α. For a Gaussian filter, the larger the standard deviation, the more significant difference between the filtered feature and the original feature, and the more reduction of the feature values. More reduction of the feature values can lead to more aggressive rendering quality reduction. Therefore, for each pixel, the greater the standard deviation of the 3D Gaussian filter within the maximum imperceptibility threshold α range, the sharper the rendering quality degradation that this pixel can tolerate. Equation 4 describes ρ_p of the 3D Gaussian filter for the pixel p that simulates the maximum reduction of the foveated spatiotemporal and the foveated luminance feature values within α.

$$\begin{aligned} &maximize \ \ \rho_p, \\ &s.t. \ \ max(fb_{st}(p), fb_l(p)) - G_{\rho_p}(p, fb_{st}, fb_l) \le \alpha \end{aligned} \tag{4}$$

where $max(fb_{st}(p), fb_l(p))$ is the larger feature value between $fb_{st}(p)$ and $fb_l(p)$. $G_{\rho_p}(p, fb_{st}, fb_l)$ is the reduced feature value of p convolved by the 3D Gaussian filter with the standard deviation ρ_p based on fb_{st} and fb_l. The larger ρ_p is, the larger the difference between the maximum feature value and the reduced feature value, so the reduced feature value $G_{\rho_p}(p, fb_{st}, fb_l)$ convolved by the 3D Gaussian filter with ρ_p can be expressed by Equation 5.

$$G_{\rho_p}(p, fb_{st}, fb_l) = max(fb_{st}(p), fb_l(p)) - \alpha \tag{5}$$

Meanwhile, $G_{\rho_p}(p, fb_{st}, fb_l)$ can also be expressed by Equation 6.

$$G_{\rho_p}(p, fb_{st}, fb_l) = e^{-\frac{(fb_{st}(p) - \overline{fb_{st}(p)})^2 + (fb_l(p) - \overline{fb_l(p)})^2}{2\pi\rho_p{}^2}} \tag{6}$$

where $\overline{fb_{st}(p)}$ is the Gaussian average value of the kernel of size 5×5 centered at $fb_{st}[p]$, and $\overline{fb_l(p)}$ is the Gaussian average value of the kernel of size 5×5 centered at $fb_l[p]$ [1]. Thus, ρ_p can be calculated by Equation 7:

$$\rho_p = \sqrt{-\frac{(fb_{st}(p) - \overline{fb_{st}(p)})^2 + (fb_l(p) - \overline{fb_l(p)})^2}{2ln(max(fb_{st}(p), fb_l(p)) - \alpha)}} \tag{7}$$

Then the value of each pixel in the FPSM map is calculated by Equation 8:

$$map[p] = 1 - \frac{\rho_p - \rho_p^{min}}{\rho_p^{max} - \rho_p^{min}} \tag{8}$$

where ρ_p^{min} is the minimum ρ_p among all pixels, and ρ_p^{max} is the maximum ρ_p among all pixels.

After constructing the FPSM map, we use it to guide lightcuts generation. For each pixel p in the output framebuffer, we multiply $map[p]$ by the maximum number of light samples per pixel MAX_{lspp} to get the number of nodes $\#L$ required for this pixel's lightcuts. Then we use the criterion introduced in [41] to refine the lightcuts until the number of the lightcuts' node becomes $\#L$.

3.2 Multi-resolution Light Samples Selection

We propose a multi-resolution light samples selection method to improve the performance of light samples selection. The motivation of the multi-resolution light samples selection is to select the light sample for each node in the lightcuts more efficiently without performing the light tree traversal for each node.

Multi-resolution light samples selection method accelerates the light samples selection by selecting light samples directly from the low-resolution light samples for nodes in the lightcuts without traversing the light tree. Given the perfect binary light tree *tree*, FPSM *map*, the low-resolution lightcuts LC', the low-resolution light samples set LS', the spatiotemporal-luminance based generated lightcuts LC, the full resolution of the output framebuffer (w,h), and the low resolution (w',h'), the lights samples ls for each pixel p are selected by using the Algorithm 2.

Algorithm 2: Multi-resolution Light Samples Selection

input : pixel of selecting light samples p, perfect binary light tree *tree*, FPSM *map*, low-resolution lightcuts LC', low-resolution light samples set LS', spatiotemporal-luminance based generated lightcuts LC, the full resolution of the output framebuffer (w,h), low resolution (w',h')

output : p's light samples ls

```
1  ls ← ∅
2  for node ∈ LC[p] do
3  |   rad ← random(0,1)
4  |   p' ← p·(w'/w, h'/h)
5  |   if rad ≤ (1.0-map[p]) and lrLS(node, LS'[p']) then
6  |   |   l ← nLight(node, LS'[p'], LC[p'])
7  |   end
8  |   else
9  |   |   l ← hierarchicalIS(tree, node)
10 |   end
11 |   ls ← ls ∪ l
12 end
```

Firstly, we initialize the light samples ls for the current pixel p as empty (line 1). Then we select the light sample for each node *node* in the lightcuts $LC[p]$ (lines 2-9).

In the light samples selection loop (line 2), we first generate a random number *rad* in the range of 0 to 1 (line 3). Then we calculate p', which is the corresponding pixel of p in the low-resolution rendering framebuffer fb' (line 4).

If *rad* is smaller than (1.0-$map[p]$), and there are low-resolution light samples (i.e. leaf nodes) in the low-resolution light samples set $LS'[p']$ that are the leaf nodes of *node*(line 5), we select the optimal low-resolution light sample l in $LS'[p']$ (line 6). The function *lrLS* is used to quickly judge if there are low-resolution light samples in $LS'[p']$ that are the leaf nodes of *node*. Since each node in *tree* can be quickly indexed by its offset value *lId* in the perfect binary light tree *tree*, and the nodes are stored consecutively from the smallest to the largest according to their offset value. Therefore, for each pixel p', nodes in the low-resolution lightcuts $LC'[p']$ are stored according to their offset values from the smallest to the largest, and light samples in $LS'[p']$ are stored according to their offset values from the smallest to the largest. *lrLS* calculates the minimum offset lId_{min} and the maximum offset lId_{max} of the *node*'s leaf nodes, and returns true if lId_{min} is less than the minimum offset of the light samples in $LS'[p']$ or lId_{max} is greater than the maximum offset of the light samples in $LS'[p']$, otherwise it will return false. lId_{min} and lId_{max} are calculated by Equation 9:

$$\begin{cases} lId_{min} = node \cdot 2^{n-\lceil log_2(node+1)\rceil} \\ lId_{max} = lId_{min} + 2^{n-\lceil log_2(node+1)\rceil} - 1 \end{cases} \tag{9}$$

where n is the number of *tree*'s level. If there are low-resolution light samples in $LS'[p']$ that are the leaf nodes of *node*, we use the function *nLight* to choose the optimal light sample l from $LS'[p']$ for *node*. *nLight* first calculates the distance *dist* between *node* and the node $node'$ in $LC'[p']$ corresponding to the light sample l in $LS'[p']$ by Equation 10, and then selects the light sample l that has the smallest distance *dist*.

$$\begin{aligned} dist = {} & \lfloor log_2(node'+1)\rfloor + \lfloor log_2(node+1)\rfloor \\ & -2\lfloor log_2(gcd(2\lfloor\frac{node'}{2}\rfloor, 2\lfloor\frac{node}{2}\rfloor)+1)\rfloor \end{aligned} \tag{10}$$

where $gcd(x,y)$ is the function used to find the greatest common divisor of x and y.

If *rad* is not smaller than (1.0-$map[p]$), or there are no low-resolution light samples in $LS'[p']$ that are the leaf nodes of *node*, we use the hierarchical importance sampling method [41] to traverse *tree* from *node* until it gets the leaf node l (lines 7-8). Finally, we merge l into ls (line 9).

4 Results and Discussion

We test our method in five scenes: *Study* (2110.1k tris., Figure 1, Figure 4 row 1), *Room* (5103.9k tris., Figure 4 row 2), *Cornellbox* (110.5k tris., row 3), *Yard* (2575.1k tris., row 4), and *Sponza* (314.1k tris., row 5). *Study*, *Room*, *Cornellbox*, and *Yard* contain diffuse and glossy objects, all of which are illuminated by mesh light sources. *Study* contains 58.1k light sources, *Room* contains 30.4k light sources, *Cornellbox* contains 110.5k light sources, and *Yard* contains 250.6k light sources. To more fully compare the quality and performance of our method, we also test our method in the virtual point light (VPL) scene *Sponza*. *Sponza* is a diffuse scene illuminated by a point light. We generate 120.0k VPLs to represent indirect lighting from the point light, and use different methods to sample these VPLs and compare the illumination quality and performance of these methods. We use an HTC Cosmos HMD with a Droolon aGlass to track the head motion and the foveated point of the user. The HMD is 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.

4.1 Implementation

To accelerate diffuse rendering for HMD binocular rendering, we shade the full left-eye view, and then render the right-eye view by sampling from the completed left-eye view using reprojection [20]. The right-eye view only has to shade new pixels in the case that no valid sample is found, rather than recalculating everything [19]. The maximum number of light samples per pixel MAX_{lspp} and the maximum number of ray samples per pixel MAX_{rspp} are set to 48 in our implementation.

4.2 Quality

The quality of our results is compared with those of ground truth (*GT*), the stochastic lights (SL_e) with the same rendering time as our method, and the stochastic lights using Stengel's adaptive foveated sampling method [26] (SL_{as}) in Figure 4. Since stochastic lightcuts can only render diffuse illumination effects, we also add the one-bounced path tracing [13] to each method for rendering glossy reflection effects. The ground truth images (*GT*, column 1) are generated by the stochastic lightcuts with 2048 light samples per pixel (*lspp*) and the one-bounced path tracing with 2048 ray samples per pixel (*rspp*). The number of *lspp* and *rspp* used in the first compared method (SL_e, column 3) makes SL_e the same frame rates as our method. *lspp* and *rspp* for each pixel in the other compared method (SL_{as}, column 4) is calculated by MAX_{lspp} and MAX_{rspp} times the corresponding value in the adaptive foveated sampling map (AFSM) generated by Stengel's adaptive foveated sampling method. The blue circles on the image of *Ours* and SL_{as} indicate foveal regions. We use a large foveal region in experiments, which is the same as the foveal region in [24]. Because a larger foveal region makes users less aware of the low-quality rendering effects in the periphery. Even in the current large foveal region, our method achieves the rendering quality in the foveal region which is similar to the ground truth. The rendering quality of our method in the foveal region will be further improved and closer to the ground truth if a smaller foveal region is given. We also crop and magnify the details in both the foveal and peripheral regions on the right of each rendering

Table 1: MSE, MSE_d and MSE_g ($\times 10^{-2}$) in the foveal regions (*Fove*) and peripheral regions (*Peri*).

Scene	MSE in *Fove*			MSE in *Peri*			MSE_d in *Fove*			MSE_d in *Peri*			MSE_g in *Fove*			MSE_g in *Peri*		
	Ours	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}
Study	**2.01**	7.33	2.70	**4.65**	6.94	5.56	**1.91**	6.87	2.62	**4.51**	6.35	5.49	**2.44**	9.32	3.03	**5.23**	9.56	5.84
Room	**2.41**	10.20	2.92	**6.28**	10.02	7.88	**1.73**	9.85	2.61	**3.92**	9.34	4.53	**4.33**	11.67	6.25	**11.26**	12.98	11.49
Cornellbox	**4.44**	8.93	4.61	**3.78**	5.72	5.13	**2.38**	4.64	2.84	**1.28**	4.72	2.13	**10.47**	21.69	10.64	**8.01**	15.02	12.74
Yard	**3.09**	5.84	4.27	**4.49**	5.36	5.30	**2.27**	4.83	3.28	**3.40**	4.03	3.90	**7.63**	10.84	9.53	**8.41**	13.04	12.57
Sponza	**3.29**	10.42	5.69	**5.78**	8.92	9.34	**3.29**	10.42	5.69	**5.78**	8.92	9.34	/	/	/	/	/	/

image for comparison (up: details in the red rectangle, down: details in the green rectangle).

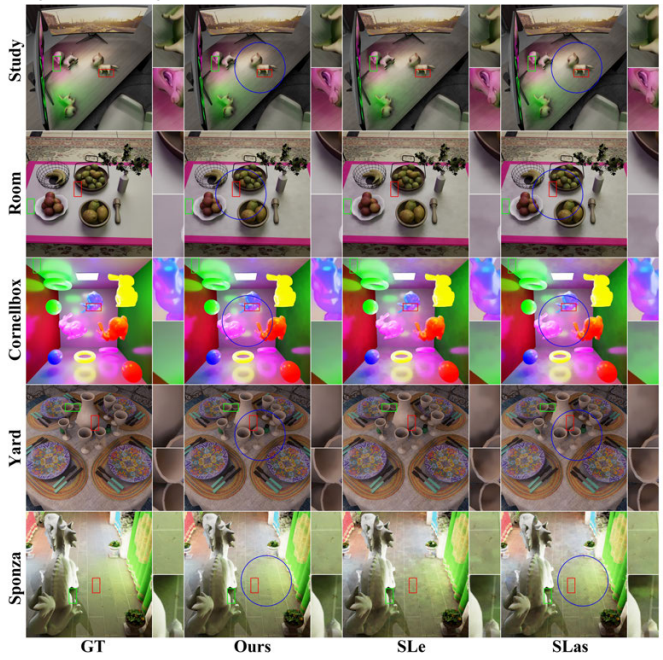

Fig. 4: Comparison between *GT* (column 1), *Ours* (column 2), SL_e (column 3) and SL_{as} (column 4) in different scenes. The details in the rectangular regions are magnified and placed on the right of each image.

Table 2: $\overline{spp}$ of our method, SL_e and SL_{as} in different regions.

Scene	$\overline{spp}$			$\overline{spp}$ in *Fove*			$\overline{spp}$ in *Peri*		
	Ours	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}	*Ours*	SL_e	SL_{as}
Study	5.5	**5.0**	6.3	22.4	**5.0**	20.1	**3.1**	5.0	4.4
Room	10.7	**10.0**	15.9	27.2	**10.0**	30.3	**8.2**	10.0	13.9
Cornellbox	8.4	**8.0**	11.8	28.0	**8.0**	27.9	**6.0**	8.0	9.3
Yard	5.5	**5.0**	8.2	22.6	**5.0**	22.7	**4.4**	5.0	6.1
Sponza	7.4	**7.0**	9.3	23.3	**7.0**	22.6	**5.2**	7.0	7.4

Our results are closer to the results of the ground truth than those of SL_e and SL_{as}. Some artifacts are shown in the rectangle regions rendered by SL_e and SL_{as}. In row 1, there is wrong red shading at the front legs of the toy cat in the red rectangle region in the result of SL_e, and the shading of the toy dog's head should be more reddish rather than greenish in the green rectangle region in the result of SL_e and SL_{as}. In row 2, the shadow under the fruit basket is not smooth in the red rectangle region in the result of SL_e, and the color shading on the table is noised in the green rectangle region in the result of SL_e and SL_{as}. In row 3, the glossy effects on the blue bunny's head are coarse in the red rectangle region in the result of SL_e and SL_{as}, and the color shading on the ceiling is noised in the green rectangle region in the result of SL_e and SL_{as}. In row 4, there are noisy points on the jar in the red rectangle region in the result of SL_e, and there are noisy points on the inner side of goblets in the green rectangle region in the result of SL_e and SL_{as}. In row 5, the color bleeding on the floor reflected by the green curtain is not smooth in the red rectangle region in the result of SL_e, and there are obvious green sparks on the floor near the feet of the dragon in the green rectangle region in the result of SL_e and SL_{as}.

To distinguish between errors caused by foveated lightcuts and errors caused by foveated path tracing, we quantify the quality with three metrics: MSE for all pixels of the rendering results compared with the ground truth, MSE_d for the pixels of the diffuse parts in the rendering results compared with the ground truth, MSE_g for the pixels of the glossy parts in the rendering results compared with the ground truth. We measure MSE, MSE_d and MSE_g in the foveal and peripheral regions separately.

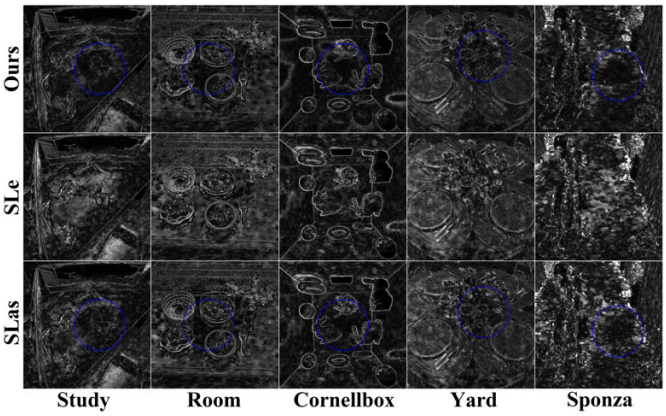

Fig. 5: Visualization of MSE of our method, SL_e and SL_{as} in different scenes.

Table 1 shows the comparison of MSE, MSE_d, and MSE_g of our method, SL_e and SL_{as} for the images in Figure 4. Benefiting from our spatiotemporal-luminance based lightcuts generation method, our method achieves smaller MSE, MSE_d, and MSE_g in both the fovea and periphery than those of SL_e and SL_{as}. Compared with SL_e, (MSE, MSE_d, MSE_g) of our method is (1.9-4.2, 1.9-5.7, 1.4-3.8)$\times$ smaller in fovea, and (1.2-1.6, 1.2-3.7, 1.2-1.9)$\times$ smaller in periphery. Compared with SL_{as}, (MSE, MSE_d, MSE_g) of our method is (1.1-1.7, 1.2-1.7, 1.1-1.4)$\times$ smaller in fovea, and (1.2-1.4, 1.1-1.7, 1.0-1.6)$\times$ smaller in periphery. This is because the FPSM constructed in our method is different from the AFSM constructed in SL_{as}, and the results show that the FPSM prefers to guide high-quality rendering for the regions with large (MSE, MSE_d, MSE_g).

Table 2 shows the comparison of the average samples per pixel $\overline{spp}$ of our method, SL_e and SL_{as} in different regions for the images in Figure 4. Compared with SL_e, our method achieves larger $\overline{spp}$ for the whole region of the current view in all the five scenes, thanks to our multi-resolution light samples selection method. $\overline{spp}$ of our method is 2.7-4.5$\times$ higher than that of SL_e in the foveal region. In the peripheral region, our method achieves higher rendering quality with lower $\overline{spp}$ than that of SL_e. This is because spp of our method is not uniform in the peripheral region, and it distributes more spp on the surface of high perception-sensitive objects in the peripheral region. Compared with SL_{as}, our method achieves higher rendering quality with lower $\overline{spp}$ in the peripheral region, and $\overline{spp}$ of our method is 1.4-1.7$\times$ smaller than that of SL_{as}.

The images of Figure 5 visualize MSE for the images in Figure 4. The whiter pixels represent the larger error. The error visualization shows that MSE of our method is always smaller than that of SL_e in both the fovea and periphery. MSE of our method is close to that of SL_{as} in the foveal region, but is smaller than that of SL_{as} in the peripheral region although $\overline{spp}$ of our method is smaller than that of SL_{as}.

The images of Figure 6 visualize the FPSM of our method (row 1) and the AFSM of SL_{as} (row 2) for the images in Figure 4. As can be seen from Figure 6, the FPSM is different from the AFSM in SL_{as}. The overall brightness of the FPSM is lower than that of the AFSM. AFSM is more inclined to the edge regions of objects, while the distribution of

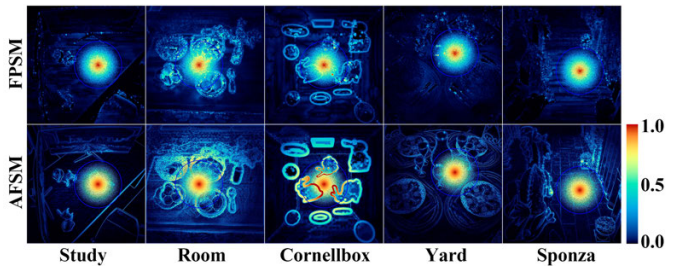

Fig. 6: Visualization of the FPSM (row 1) and the AFSM (row 2) in different scenes.

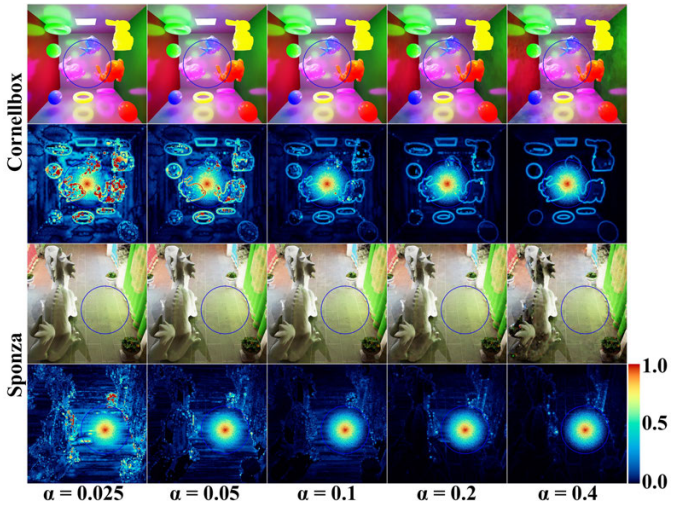

Fig. 7: Rendering results and the FPSM visualization of our method with different values of the maximum imperceptibility threshold α in *Cornellbox* and *Sponza*.

the FPSM is more uniform and concentrates on the entire surface of obvious objects.

The maximum imperceptibility threshold α of our method affects the image quality. Figure 7 shows the rendering results of our method in *Cornellbox* and *Sponza* with various α. Images in row 1 and row 3 show the rendering result, and images in row 2 and row 4 show the visualization of the FPSM when α is gradually increased from the left to the right: 0.025, 0.05, 0.1, 0.2, 0.4. In the rendering results of *Cornellbox* and *Sponza*, there is no obvious visual difference among the images in the foveal region, while the noise in the peripheral region increases when α increases. When α increases from 0.025 to 0.1, the rendering quality of the peripheral region does not decrease significantly, but when α continues to increase, the rendering quality of the peripheral region drops sharply. In the visualization of the FPSM, the brightness of the map in the peripheral region decreases when α increases. *MSE* and MSE_d in the foveal and peripheral regions of these two scenes are shown in Figure 8.

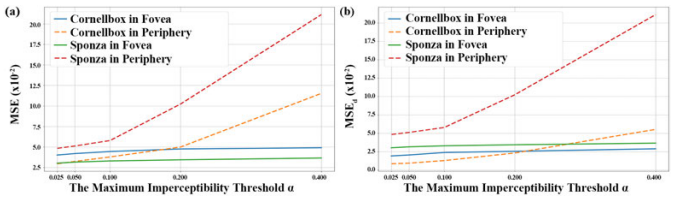

Fig. 8: *MSE* (a) and MSE_d (b) in *Cornellbox* and *Sponza* as a function of the maximum imperceptibility threshold α.

4.3 Performance

Table 3 shows the frame rendering time of our method and stochastic lightcuts with different rendering qualities. Columns 2 and 3 show the time cost of our method with (*Ours*) and without the multi-resolution light samples selection (*Ours*$'$). Our method with the multi-resolution light samples selection method gets 1.3-1.4× speedup. Compared with SL_{as} (column 4), our method achieves 1.3-2.0× speedup, and our method achieves higher rendering quality in both the foveal and peripheral regions. Compared with the stochastic lightcuts method SL_f (column 5) that has the comparable rendering quality in the foveal region as our method, our method achieves 2.5-3.9× speedup. Compared with the stochastic lightcuts method SL_p (column 6) that has the comparable rendering quality in the peripheral region as our method, our method achieves 1.6-2.4× speedup.

Table 3: Performance (ms) of our method compared with stochastic lightcuts.

Scene	*Ours*	*Ours*$'$	SL_{as}	SL_f	SL_p
Study	**9.1**	12.0	15.1	25.9	17.8
Room	**13.3**	17.8	26.5	52.5	32.3
Cornellbox	**12.6**	16.6	16.9	33.3	27.4
Yard	**18.9**	24.6	28.2	46.5	29.4
Sponza	**13.2**	18.9	23.7	49.6	29.1

Figure 9 shows the time cost on each step using our method (a) and SL_f (b) for rendering each frame of *Study*. Our method has five steps: 1) light tree construction, 2) spatiotemporal-luminance based lightcuts generation, 3) multi-resolution light samples selection, 4) illumination shading using light samples, 5) foveated path tracing for glossy effects. SL_f also has five steps: 1) light tree construction, 2) lightcuts generation, 3) light samples selection, 4) illumination shading using light samples, 5) one-bounced path tracing for glossy effects. Light tree construction of our method is almost the same as that of SL_f. Our spatiotemporal-luminance based lightcuts generation only achieves 1.1× speedup compared with lightcuts generation of SL_f. This is because we need to construct the FPSM in spatiotemporal-luminance based lightcuts generation, which incurs additional time overhead. Benefiting from our spatiotemporal-luminance based lightcuts generation, illumination shading using light samples and foveated path tracing for glossy effects of our method achieve 4.0× speedup compared with those of SL_f. This is because lightcuts generated by spatiotemporal-luminance based lightcuts generation have fewer nodes than those generated by lightcuts generation of SL_f, and using lightcuts with fewer nodes can speed up illumination shading using light samples. And the FPSM constructed in spatiotemporal-luminance based lightcuts generation can guide the allocation of *rspp* in foveated path tracing for glossy effects, allocating less *rspp* to low perception-sensitive regions to speed up foveated path tracing. Multi-resolution light samples selection of our method is 4.4× faster than light samples selection of SL_f.

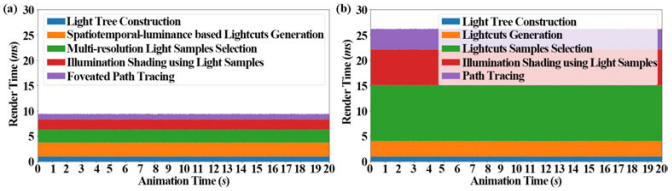

Fig. 9: Time cost in each step of our method (a) compared with that of SL_f (b) in *Study*.

Table 4 shows the performance in five scenes with the different maximum imperceptibility threshold α. When α increases from 0.025 to 0.1, the time cost of each frame is shortened by 13.6-30.2*ms*, but when α goes from 0.1 to 0.4, the time cost of each frame is shortened by only 4.2-8.9*ms*. This is because with the increase of α, the time cost of light tree construction and spatiotemporal-luminance based lightcuts generation does not decrease significantly, requiring a constant time of about 2.5*ms*. Only multi-resolution light samples selection, illumination shading using light sample, and foveated path tracing for glossy effect decrease linearly with the increase of α, but as α increases from 0.1 to 0.4, this makes the constant time consumption of light tree construction and spatiotemporal-luminance based lightcuts generation become one of the bottlenecks.

5 User Study

We design the within-subject study [9] to evaluate the perceived rendering quality between our method and stochastic lightcuts.

Conditions. We use our method as an experimental condition (*EC*), and *EC* uses our method to render five scenes {*Study*, *Room*, *Cornellbox*, *Yard*, *Sponza*}. The first control condition (*CC*1) uses *GT* to render five scenes. The second control condition (*CC*2) uses

Table 4: Performance (ms) of our method with different α in different scenes.

Scene	The Maximum Imperceptibility Threshold α				
	0.025	0.05	0.1	0.2	0.4
Study	22.7	16.4	9.1	6.5	**4.9**
Room	37.3	25.3	13.3	9.0	**6.4**
Cornellbox	31.5	25.1	12.6	9.1	**6.5**
Yard	49.1	36.6	18.9	14.0	**10.0**
Sponza	36.9	25.3	13.2	9.5	**7.0**

SL_e to render five scenes. The third control condition (*CC*3) uses SL_{as} to render five scenes. The forth control condition (*CC*4) uses SL_f to render five scenes.

Task 1. In Task 1, we test different scenes sequentially from *Study* to *Sponza*. For each scene, we first show the animation sequence rendered by *CC*1, then tell the participant that this is the ground truth, and then present the participant with the short animated sequences that rendered by *EC*, *CC*1, *CC*2, *CC*3 and *CC*4 in randomized order. Each short animated sequence is 8.0s long and separated by a short interval (0.5s) of black, which is the same as Guenter et al. [12]. In the process of the task, participants are asked to press one of two buttons (yes or no) to answer the question 'is this sequence the ground truth ?' after being presented each animated sequence. After this, the next sequence comes in.

Task 2. In Task 2, each participant explores five scenes rendered by *EC*, *CC*2, *CC*3 and *CC*4 in randomized order. Each exploration is 16.0s long and separated by a short interval (1.0s) of black. In the process of the task, participants are asked to press one of two buttons (acceptable or unacceptable) to answer the question 'is the quality of this sequence acceptable?' after each exploration. After this, the next exploration begins.

Participants. We recruit 32 participants in this user study, including 20 males and 12 females, who are between 18 and 52 years old. Each participant completes 50 trials in Task 1, and 40 trials in Task 2. Each item in all conditions is presented twice in Task 1 and 2. Since *CC*1 can not enable the interactive frame rates, it is not added to the comparison of Task 2. In each task, we record the participant's positive answer with a score of 1 and negative answer with a score of 0, i.e., 1 for 'yes' to the question of Task 1 and 'acceptable' to the question of Task 2.

Table 5: The scores of Task 1 for the perceived rendering quality of each item in *EC*, *CC*1, *CC*2, *CC*3 and *CC*4.

con.	avg. $\pm$ std. dev.	p value	Cohen's d	effect size
EC	0.79 ± 0.40	/	/	/
CC1	$\mathbf{0.96 \pm 0.19}$	<0.001	0.53	medium
CC2	0.17 ± 0.37	<0.001	1.62	very large
CC3	0.38 ± 0.49	<0.001	0.92	large
CC4	0.85 ± 0.36	0.063	0.15	very small

Results and Discussion. Table 5 shows the scores of Task 1 for *EC*, *CC*1, *CC*2, *CC*3 and *CC*4 in all scenes. We calculate the average scores (avg.) of all conditions, and use the p value [36] and Cohen's d [4] to estimate the difference between *EC* and other conditions. Avg. of *EC* and *CC*4 are closer to *CC*1 than those of *CC*2 and *CC*3. And avg. of *EC* is slightly lower than that of *CC*4. This is because *CC*4 has the high-quality illumination effects in both the foveal and peripheral regions, while the rendering quality of *EC* in some low perception-sensitive parts in the periphery is low. But the p value and Cohen's d both show that the perceived rendering quality of *EC* is similar to *CC*4, the effect size of Cohen's d is very small. Avg. of *EC* is significantly higher than that of *CC*2 and *CC*3. This is because the rendering quality of *CC*2 is low, and some high perception-sensitive parts in the periphery of *CC*3 can not be rendered with high quality. The p value and Cohen's d both show that the perceived rendering quality of *EC* is significant different from that of *CC*2 and *CC*3, effect size of Cohen's d in *CC*2 and *CC*3 is very large and large.

Table 6 shows the scores of Task 2 for *EC*, *CC*2, *CC*3 and *CC*4 in all scenes. Avg. of *EC* is significant higher than that of *CC*2, *CC*3 and *CC*4. Avg. of *CC*4 is not in line with the trend of Task 1. This is because the frame rates of *CC*4 can not support participants to explore virtual scenes immersively, although the rendering quality of *CC*4 is high enough. The perceived rendering quality of *EC* is significant different from that of *CC*2, *CC*3 and *CC*4 in the results of p value and Cohen's d, effect size of Cohen's d in *CC*2, *CC*3 and *CC*4 is very large, large and large separately.

Table 6: The scores of Task 2 for the perceived rendering quality of each item in *EC*, *CC*2, *CC*3 and *CC*4.

con.	avg. $\pm$ std. dev.	p value	Cohen's d	effect size
EC	$\mathbf{0.86 \pm 0.35}$	/	/	/
CC2	0.18 ± 0.39	<0.001	1.83	very large
CC3	0.50 ± 0.50	<0.001	0.80	large
CC4	0.47 ± 0.50	<0.001	0.90	large

6 Conclusion, Limitations and Future Work

Foveated rendering is a rendering framework for improving the rendering performance, which is particularly useful for VR HMD rendering. Many researchers adapted the existing rendering methods into the framework of foveated rendering, such as rasterization based foveated rendering [17], foveated ray tracing [37], and foveated instant radiosity [34], etc. We have proposed a foveated stochastic lightcuts method to adapt stochastic lightcuts into the framework of foveated rendering, so as to better support the many-lights illumination rendering in VR HMDs. Compared with the previous work, the novelty of our method is to provide an efficient dynamic many-lights rendering algorithm in VR HMDs, which uses the spatiotemporal and luminance sensitivity of the HVS to guide the lightcuts generation, and accelerates light samples selection for further improving the rendering performance. The foveated stochastic lightcuts renders high-quality perceived many-lights illumination effects of the virtual scenes for the HVS at high frame rates. The perceived quality of the foveated stochastic lightcuts has a high visual similarity with the results of the ground truth rendered by using the stochastic lightcuts with 2048 *lspp* and the one-bounced path tracing with 2048 *rspp*. The foveated stochastic lightcuts provides a new foveated rendering method for VR HMDs, which supports full-dynamic scenes containing over 250k dynamic light sources and diffuse/specular/glossy dynamic objects with the frame rates up to 110fps. Compared with the state-of-the-art stochastic lightcuts method with the comparable rendering quality in the fovea, our method achieves 2.5-3.9$\times$ speedup.

The virtual scene rendered by our method contains not only diffuse objects, but also specular and glossy objects. This is because we add a foveated path tracing step according to the FPSM after the shading of the lightcuts method. One limitation is that our method can not be extended to render transparent and translucent objects. The bidirectional lightcuts (BLC) [33] extends the lightcuts method to support the transparent material, but the complex bounding function of nodes in the light tree is time-consuming. So one possible future work is to accelerate BLC and to adapt it into the framework of foveated rendering. Another limitation of our method is that the efficiency of our multi-resolution light samples selection method may drop when there are a large number of light sources with complex shapes in a small space. Because in this case, when generating the high-resolution rendering result, the light samples for each pixel are quite different, and the light samples stored in the step of the low-resolution rendering cannot provide enough reusable light samples for high-resolution rendering. Therefore, the future work is to perform adaptive multi-resolution rendering according to the features of the scenes, to generate more low-resolution lighting samples for regions that require a large number of lighting samples, and to improve the reuse rate of low-resolution lighting samples, thereby improving rendering performance. In addition, using deep learning models to construct the map to guide the lightcuts generation is an interesting idea for accelerating the lightcuts method. However, the visual latency tolerance threshold of HVS is around 13ms, and complex deep learning models may be difficult to meet the performance requirements of VR applications. In future work, we consider creating an efficient generative deep learning model to construct the map for guiding the lightcuts generation.

Acknowledgments

This work is supported by National Key R&D plan 2019YFC1521102, by the National Natural Science Foundation of China through Projects 61932003 and 61772051.

REFERENCES

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[2] B. Bastani, B. Funt, S. Vignaud, and H. Jiang. Smoothly varying foveated rendering, Jan. 28 2020. US Patent 10,546,364.

[3] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pp. 671–679. Elsevier, 1987.

[4] J. Cohen. *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[5] A. Conty Estevez and C. Kulla. Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):1–17, 2018.

[6] T. Davidovič, I. Georgiev, and P. Slusallek. Progressive lightcuts for gpu. In *ACM SIGGRAPH 2012 Talks*, pp. 1–1. 2012.

[7] Z. Dengwen. An edge-directed bicubic interpolation algorithm. In *2010 3rd international congress on image and signal processing*, vol. 3, pp. 1186–1189. IEEE, 2010.

[8] A. T. Duchowski, D. Bate, P. Stringfellow, K. Thakur, B. J. Melloy, and A. K. Gramopadhye. On spatiochromatic visual sensitivity and peripheral color lod management. *ACM Transactions on Applied Perception (TAP)*, 6(2):1–18, 2009.

[9] A. Field and G. Hole. *How to design and report experiments*. Sage, 2002.

[10] L. Franke, L. Fink, J. Martschinke, K. Selgrad, and M. Stamminger. Time-warped foveated rendering for virtual reality headsets. In *Computer Graphics Forum*, vol. 40, pp. 110–123. Wiley Online Library, 2021.

[11] S. Friston, T. Ritschel, and A. Steed. Perceptual rasterization for head-mounted display image synthesis. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[12] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012.

[13] E. P. Lafortune and Y. D. Willems. Bi-directional path tracing. 1993.

[14] D. Lin and C. Yuksel. Real-time stochastic lightcuts. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1):1–18, 2020.

[15] T. Lindeberg. Concealing rendering simplifications using gazecontingent depth of field, 2016.

[16] X. Meng, R. Du, and A. Varshney. Eye-dominance-guided foveated rendering. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1972–1980, 2020.

[17] X. Meng, R. Du, M. Zwicker, and A. Varshney. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018.

[18] P. Moreau, M. Pharr, and P. Clarberg. Dynamic many-light sampling for real-time ray tracing. In *High Performance Graphics (Short Papers)*, pp. 21–26, 2019.

[19] H. Moreton and N. Stam. Turing texture space shading, 2018. http://https://devblogs.nvidia.com/texture-space-shading.

[20] D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, and J. R. Isidoro. Accelerating real-time shading with reverse reprojection caching. In *Graphics hardware*, vol. 41, pp. 61–62, 2007.

[21] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):179, 2016.

[22] E. Peli. Contrast in complex images. *JOSA A*, 7(10):2032–2040, 1990.

[23] M. C. Potter, B. Wyble, C. E. Hagmann, and E. S. McCourt. Detecting meaning in rsvp at 13 ms per picture. *Attention, Perception, & Psychophysics*, 76(2):270–279, 2014.

[24] X. Shi, L. Wang, X. Wei, and L.-Q. Yan. Foveated photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4183–4193, 2021.

[25] J. R. Stafford and A. Young. Selective peripheral vision filtering in a foveated rendering system, Jan. 1 2019. US Patent 10,169,846.

[26] M. Stengel, S. Grogorick, M. Eisemann, and M. Magnor. Adaptive image-space sampling for gaze-contingent real-time rendering. In *Computer Graphics Forum*, vol. 35, pp. 129–139. Wiley Online Library, 2016.

[27] N. T. Swafford, J. A. Iglesias-Guitian, C. Koniaris, B. Moon, D. Cosker, and K. Mitchell. User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception*, pp. 7–14, 2016.

[28] E. Turner, H. Jiang, D. Saint-Macary, and B. Bastani. Phase-aligned foveated rendering for virtual reality headsets. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1–2. IEEE, 2018.

[29] O. T. Tursun, E. Arabadzhiyska-Koleva, M. Wernikowski, R. Mantiuk, H.-P. Seidel, K. Myszkowski, and P. Didyk. Luminance-contrast-aware foveated rendering. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[30] O. R. Vincent, O. Folorunso, et al. A descriptive algorithm for sobel image edge detection. In *Proceedings of informing science & IT education conference (InSITE)*, vol. 40, pp. 97–107, 2009.

[31] B. Walter, A. Arbree, K. Bala, and D. P. Greenberg. Multidimensional lightcuts. In *ACM SIGGRAPH 2006 Papers*, pp. 1081–1088. 2006.

[32] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: a scalable approach to illumination. In *ACM SIGGRAPH 2005 Papers*, pp. 1098–1107. 2005.

[33] B. Walter, P. Khungurn, and K. Bala. Bidirectional lightcuts. *ACM Trans. Graph.*, 31(4), jul 2012. doi: 10.1145/2185520.2185555

[34] L. Wang, R. Li, X. Shi, L.-Q. Yan, and Z. Li. Foveated instant radiosity. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 1–11. IEEE, 2020.

[35] R. Wang, Y. Huo, Y. Yuan, K. Zhou, W. Hua, and H. Bao. Gpu-based out-of-core many-lights rendering. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.

[36] R. L. Wasserstein and N. A. Lazar. The asa statement on p-values: context, process, and purpose, 2016.

[37] M. Weier, T. Roth, E. Kruijff, A. Hinkenjann, A. Pérard-Gayot, P. Slusallek, and Y. Li. Foveated real-time ray tracing for head-mounted displays. *Computer Graphics Forum*, 35:289–298, 10 2016. doi: 10.1111/cgf.13026

[38] Q. Yang, Z. Chen, Y. Liu, G. Xing, and Y. Zhang. Foveated light culling. *Computers & Graphics*, 97:200–207, 2021. doi: 10.1016/j.cag.2021.04.021

[39] A. Young and J. R. Stafford. Real-time user adaptive foveated rendering, Jan. 29 2019. US Patent 10,192,528.

[40] C. Yuksel. Stochastic lightcuts. In *Proceedings of the Conference on High-Performance Graphics*, pp. 27–32, 2019.

[41] C. Yuksel. Stochastic lightcuts for sampling many lights. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[42] C. Yuksel and C. Yuksel. Lighting grid hierarchy for self-illuminating explosions. *ACM Trans. Graph.*, 36(4):110–1, 2017.

[43] Z. Zheng, Z. Yang, Y. Zhan, Y. Li, and W. Yu. Perceptual model optimized efficient foveated rendering. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–2, 2018.