Eurographics Symposium on Rendering (DL-only Track) (2019)
T. Boubekeur and P. Sen (Editors)

Foveated Real-Time Path Tracing in Visual-Polar Space

M. Koskela[1],A. Lotvonen[1], M. Mäkitalo[1], P. Kivi[1], T. Viitanen[1,2], and P. Jääskeläinen[1]

[1]Tampere University, Tampere, Finland
[2]Now with Nvidia, Helsinki, Finland

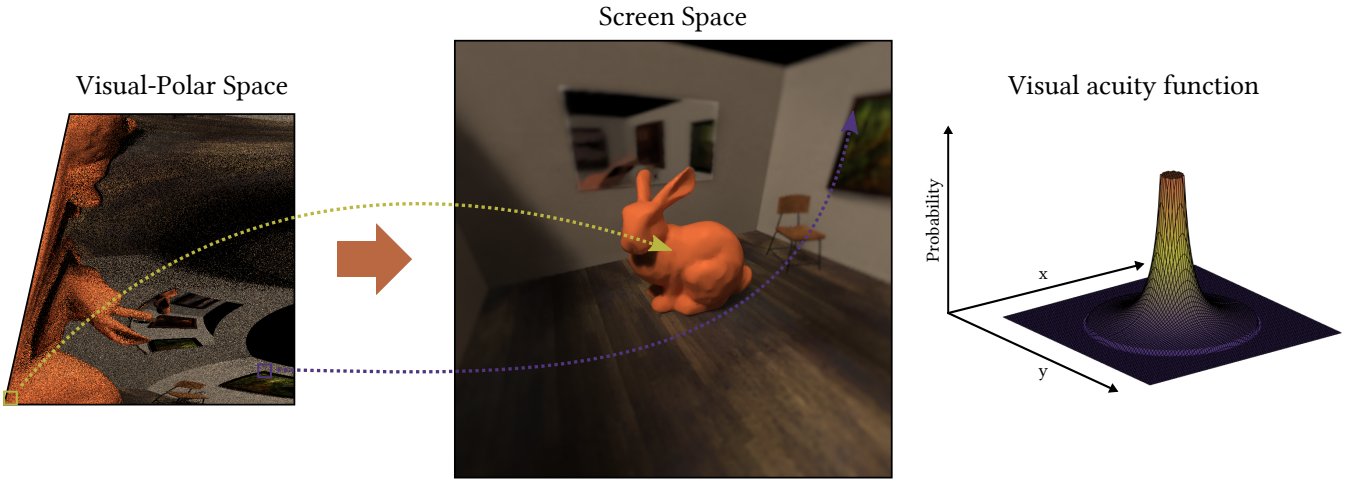


Figure 1: *Illustration of a path traced frame in Visual-Polar space, the denoised result transformed into Cartesian screen space, and the distribution of the path tracing samples in screen space. Path tracing and denoising in Visual-Polar space makes both* $2.5\times$ *faster.*

Abstract

Computing power is still the limiting factor in photorealistic real-time rendering. Foveated rendering improves perceived quality by focusing the rendering effort on where the user is looking at. Applying foveated rendering to real-time path tracing where we must work on a very small number of samples per pixel introduces additional challenges; the rendering result is thoroughly noisy and sparse in the periphery. In this paper we demonstrate foveated real-time path tracing system and propose a novel Visual-Polar space in which both real-time path tracing and denoising is done before mapping to screen space. When path tracing a regular grid of samples in Visual-Polar space, the screen space sample distribution follows the human visual acuity model, making both the rendering and denoising $2.5\times$ *faster with similar perceived quality. In addition, when using Visual-Polar space, primary rays stay more coherent, leading to improved utilization of the GPU resources and, therefore, making ray traversal* $1.3-1.5\times$ *faster. Moreover, Visual-Polar space improves 1 sample per pixel denoising quality in the fovea. We show that Visual-Polar based path tracing enables real-time rendering for contemporary virtual reality devices even without dedicated ray tracing hardware acceleration.*

CCS Concepts

• ***Computing methodologies*** → ***Perception;*** *Ray tracing; Virtual reality;*

1. Introduction

In order to produce an immersive and comfortable *virtual reality* (VR) or gaming experience with the evolving *head mounted displays* (HMD), the ability to generate high resolution content with a very high frame rate is essential. Computing power remains to be the limiting factor in generating realistic content for these devices. Some approaches to overcome this problem include having a spatially varying shading rate [VST*14; HGF14] and temporally varying shading locations with reprojection based on camera movement and animations [HEMS10; XLV18]. Another idea is to optimize rendering based on the fact that the human visual system recog-

DOI: 10.2312/sr.20191219

nizes details accurately only on a small area around the gaze point. Using this information in rendering optimization is typically called foveated rendering [WSR*17].

Path tracing is one of the most interesting options for generating realistic content. It simulates how photons interact with the scene and is thus able to naturally generate real life effects such as soft shadows, global illumination and reflections. Even though there is now dedicated acceleration hardware in consumer desktop GPUs for ray tracing [KMSB18], the achievable real-time path tracing rendering budget is still below 1 *sample per pixel* (spp) [Bar18]. The need for high resolution and refresh rate further reduces the available sample budget. In addition, higher spp counts are needed for rendering more complex materials and producing effects such as depth of field and motion blur.

In this article, we optimize path tracing rendering using foveation based methods. Our system utilizes a novel human visual system inspired sample coordinate space we call the Visual-Polar space. The main idea of the proposed method can be seen in Figure 1. Our rendering demonstrator uses only 0.4 spp and is still able to generate visually pleasing fully denoised results. To the best of our knowledge this is the first time foveated denoised path tracing is demonstrated in real-time with full resolution of contemporary VR device. The main contributions of this article are:

1. We propose a novel Visual-Polar space, which saves 61% of the rendering work compared to Cartesian screen space and allows coherent primary rays with improved SIMD/SIMT utilization. The only additional overhead is mapping back to the Cartesian screen space which with our test setup takes only 1.6 ms for a contemporary 1280×1440 VR HMD resolution.
2. We show that state-of-the-art real-time path tracing denoisers such as A-SVGF [SPD18], BMFR [KIM*19], and SVGF [SKW*17] can all operate in the proposed Visual-Polar space, which saves 61% of the denoising work and requires only minor changes to the denoiser.
3. We show that in the fovea the denoised output quality improves beyond conventional screen space quality, because when the denoiser is applied in the Visual-Polar space it automatically adapts to higher than 1 spp in the fovea.

2. Related Work

Path tracing rendering evaluates the rendering equation via Monte Carlo integration; therefore, it converges to the correct result when more and more noisy samples are averaged [Kaj86]. Even on recent GPUs with hardware acceleration for ray traversal, in real time we can only path trace approximately 1 spp [Bar18], which results in a highly noisy image. Two basic approaches to improving image quality are to apply denoising filters, and to reuse and accumulate samples from previous frames, resulting in a higher effective sample count. Recent real-time ray tracing methods combine both approaches to cope with 1 spp inputs.

One option for the real-time filtering is a wavelet-based filter called *Spatiotemporal Variance-Guided Filtering* (SVGF) [SKW*17]. To achieve real-time denoising, SVGF uses temporal accumulation to have an increased effective sample count and spatiotemporal luminance variance estimations for wavelet-based spatial filtering. SVGF's advanced version (A-SVGF) [SPD18] derives adaptive temporal accumulation factors to add support for temporal effects such as moving lights. It also improves the quality of materials, such as mirrors, where the first bounce motion vectors produce blurred results. On the other hand, regression-based methods have previously shown good denoising results in offline rendering [BRM*16], and a recent real-time work, called *Blockwise Multi-order Feature Regression* (BMFR), achieved even faster performance than wavelets [KIM*19]. The idea behind BMFR is to do fitting of the feature data to noisy input in relatively big blocks instead of deciding every pixel's color individually.

Path traced frames are typically viewed by a human visual system and an interesting characteristic of the system is that it can only resolve details accurately in a very small area around the gaze point. The number of cycles per eccentricity degree a human eye can resolve is described in the so-called visual acuity function as

$$V(e) = \begin{cases} 60.0 & 0 \leq e \leq 5.79 \\ \frac{449.4}{(0.3e+1)^2} & e > 5.79 \end{cases}, \quad (1)$$

where e is the eccentricity angle, and the result tells how many times per degree the image can change from completely white to completely black [Red97]. The function has been determined in user studies. If the change is not from completely white to completely black the resolvable cycles per degree is even less. The figure for showing different resolvable cycles per degree as a function of contrast is called the *Contrast Sensitivity Function* (CSF) [SRJ11].

Interestingly, it follows from the visual acuity function that if we had a rendering system capable of showing 60 cycles per degree, 95% of the rendered detail would be excessive [KVJT16]. On contemporary HMD devices this figure is around 75% depending on the resolution and the *field of view* (FOV). However, simply reducing sampling according to the visual acuity function causes both spatial and temporal aliasing artifacts in peripheral parts of the vision. Peripheral parts of the vision are sensitive especially to temporal artifacts [WSR*17] and, therefore, overly simple periphery quality reduction methods without temporal filtering are easily detectable by the user.

Foveated rendering utilizes these known features of the human visual system to reduce computational costs with a minimal noticeable quality decrease. The literature review [WSR*17] gives a comprehensive summary on previous foveated rendering research. The basic idea is to approximate the visual acuity of the human visual system in the distribution of samples. Foveated sample distribution can also be combined with other sample importance metrics, e.g., it is more important to shade the pixels around the object silhouettes [SGEM16].

With a rasterization type rendering, a coarse approximation of foveated sampling can be achieved by rendering multiple views of the scene at different resolutions [PSK*16; GFD*12; LW90; WWHW97]. The viewport is rendered fully only with a low resolution and a smaller image with greater pixel density is rendered and overlaid at the gaze region. Typically, there is some overlap in the

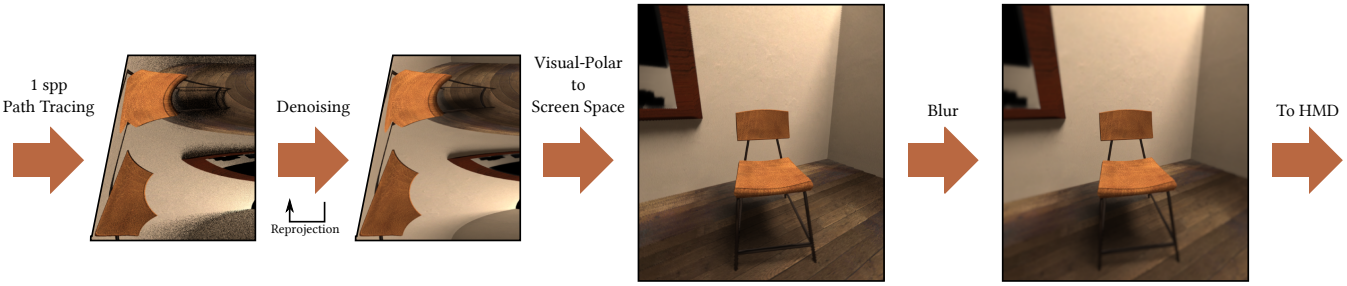


Figure 2: *The foveated Visual-Polar path tracing rendering pipeline is described in Sec. 3.*

rendered areas and interpolation is used to make the transition between resolutions smooth. For more accurate approximations there can also be a third intermediate resolution image [GFD*12].

Ray tracing is a good option for foveated rendering since it allows flexible sampling in screen space [KVJT16]. With rasterization, flexible sampling would require modifying the rasterization algorithm, which is typically done with dedicated hardware accelerators. The new Nvidia Turing architecture supports variable rate shading in screen space. However, the shading rate needs to be decided for a square block of pixels and completely free shading locations are not allowed [KMSB18]. However, due to how different work-items are scheduled to the processing elements of a programmable GPU, even with ray tracing it is difficult to do flexible sampling so that the whole computing capacity of the hardware is utilized.

For example, a common sample distribution of "linear falloff" uses a rendering probability for the peripheral part and does linear blending to full rendering in the fovea [WRK*16; WRHS18a]. Randomly killing some of the rays in the periphery causes idling lanes with both SIMD and SIMT hardware and is therefore suboptimal in terms of hardware utilization.

Sampling can also follow the visual acuity function more closely like in [KIV*18]. In this case, the GPU utilization is high, but the ray distribution in lanes is completely random, which makes primary rays completely incoherent and thus reduces cache locality. Another approach is to have a predefined sampling map [SCMP19]. They achieve fast interpolation from sparse ray tracing locations to full screen resolution, by using a precomputed triangulated mesh. In addition, one way to sample is to use spatially varying pixel density based on the distance to the shifting foveation point like in [RFS18].

An interesting option is to use log-polar space for the rendering. One downside of polar spaces is that they have discontinuity. However, artifacts can be avoided if discontinuity is taken into account, e.g., by using wrap around accesses. if ray tracing was done directly in log-polar space, the primary rays are coherent, and the rendered pixels are not sparse. One option is to first rasterize the G-buffer in Cartesian space, then map the result to log-polar space for deferred shading, and finally map the shading back to Cartesian screen space [MDZV18]. However, in the previous work log-polar distribution was not compared to the human visual system and even with the introduced kernel function all the tested parameters pack more samples to the gaze point than required.

In this paper we apply foveation-based rendering to path tracing and introduce the Visual-Polar space, which distributes the samples according to the visual acuity function. It also has coherent primary rays and all the lanes of SIMD/SIMT hardware are used, resulting in full utilization of all hardware resources. We use 1 spp real-time path tracing to generate the frames and denoise them with A-SVGF, BMFR, and SVGF. In contrast to previous work, we denoise in the Visual-Polar space before mapping the image to screen space, which means that the denoiser only needs to handle the lower resolution, and any possible denoising artifacts get circularly bent around the fovea.

3. Visual-Polar Space

The pipeline of path tracing rendering in the proposed Visual-Polar space is shown in Figure 2. The pipeline stages are described in the following subsections.

3.1. Path Tracing Setup

In this paper the 1 spp path tracing is done similarly as described in BMFR [KIM*19] and SVGF [SKW*17]. That is, we have one primary ray from every pixel and from the closest intersection point in the 3D space we trace one secondary indirect ray. From the hit points of the both rays we trace one shadow ray towards a random point in a random light.

The ray traversal, in other words, finding the closest intersection of each ray typically takes around 20% of the execution time in the path tracer used in our experiments. The coherence of the rays is quite high, with the path tracing setup used by this paper. Two out of four rays are either highly coherent primary rays or shadow rays traced from the first intersection towards the lights. With just one light these shadow rays are also highly coherent. Ray traversal of incoherent rays is significantly slower than ray traversal of coherent rays [Bar18]. Also, shader execution in the hit surface is faster with the coherent rays, because in a typical scene nearby areas have the same material. The best hardware utilization is achieved if all the work-items in a wavefront execute the same material code and, therefore, execute the same branches and load the same data. For these reasons, it is important that our foveated path tracing has similarly coherent primary rays as the Cartesian path tracing. Otherwise some of the gain from the foveation is lost in the inefficient ray traversal.

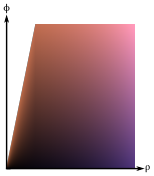
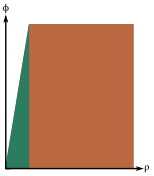
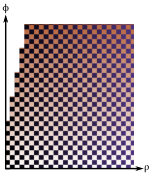


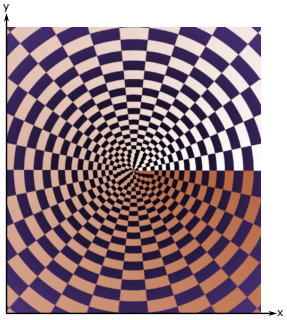
Figure 3: *Visualization of the axes ranges. Distance coordinate* ρ *runs from zero (black) to maximum field of view (purple) on horizontal axis. Angle coordinate* ϕ *runs from zero degrees (black) to 360 degrees (orange) on vertical axis. Notice how in the fovea area even with the triangular clipping the whole range of degrees is still present. When a uniform grid of samples is path traced in this space, the distribution shown in Figure 1 is achieved.*



(a) *Visual-Polar space of SVGF*



(b) *Visual-Polar space of BMFR*



(c) *Screen space showing BMFR block distribution*

Figure 4: *Visual-Polar spaces for both of the denoisers.* ρ *is on the horizontal axis and* ϕ *is on the vertical axis. BMFR's blocks are shown in the screen space image. In the BMFR figures lighter orange means smaller* ϕ *values and darker purple means smaller* ρ *values.*

3.2. Rendering

There were two main requirements we placed for the designed ray tracing sample distribution function. First, the ray traced samples should be picked in such a way that the primary rays (and also the first shadow rays) are coherent, which means that the rays in a single wavefront have approximately the same origin and traverse to approximately the same direction and, therefore, access the same *bounding volume hierarchy* (BVH) nodes as often as possible. Second, it should be feasible to perform denoising locally in the designed space that has less pixels to process than in the screen space.

An interesting option that fulfills both of the requirements is to ray trace in a polar coordinate space, so that the gaze point is always at the zero of the radius coordinate and the angle coordinate rotates around the gaze point. That is, if ray tracing a uniform grid in polar space, its sample distribution would be $\frac{1}{e}$, where e is the eccentricity angle. The problem is that this distribution does not accurately model the visual acuity function of the human eye. There are at least two simple ways to improve the distribution: Either adjust the number of samples on the angle coordinate ϕ, or change the scaling of the radius coordinate ρ.

Adjusting the number of samples on the angle coordinate requires a varying resolution on the ϕ-axis. For example, a constant sample distribution could be achieved by clipping the polar space along the $\phi = 2\pi\rho$ line, and with a more complicated clipping pattern we can match sampling with the visual acuity sample distribution. However, single sample coverage in the peripheral parts becomes stretched, which produces significant artifacts.

In the other option of scaling the radius coordinate ρ to follow the visual acuity distribution, the cumulative distribution function of the desired distribution and its inverse are needed [Dev86]. The inverse of the cumulative distribution function of the visual acuity function is too complex for a real-time implementation to be computed online [KIV*18]. However, we found that a fitted polynomial can approximate it efficiently enough for our use case. The downside of scaling the radius coordinate is that if the polar coordinate space has a constant resolution on the ϕ-axis, the first columns of samples are mapped to cover to whole fovea area and there are major stretching artifacts.

In summary, just varying the resolution of the ϕ-axis results in artifacts in peripheral vision, and just scaling the ρ-axis results in artifacts at the gaze point. Therefore, we chose a distribution where we combine the best parts of both approaches: In the fovea area we vary the resolution in the ϕ-axis and in the peripheral area we scale the ρ-axis.

In order to have a constant distribution of samples in the fovea area, we use linear mapping on the ρ-axis and clip a triangular area off from the ϕ-axis. The fovea area is marked with green color in Figure 4a. The white clipped area above the green triangle reduces 12% of the original space size. We conservatively chose the clipping boundary so that the fovea has more than 1 spp in screen space.

In the peripheral part we use scaling of the ρ-axis to achieve the visual acuity falloff. The cumulative distribution function of the visual acuity is

$$\left(\frac{1}{0.3d+1} + \ln(0.3d+1)\right) \times 166.4\pi - 612.3, \qquad (2)$$

where d is the distance to the gaze point [KIV*18]. Outside the fovea the ρ-axis is scaled with Eq. 2 when mapping from screen space to the Visual-Polar space. In the actual implementation we avoid the logarithm and the division by using a least squares fitted polynomial as an approximation of this function.

The fitted inverse of the cumulative distribution function which is used when mapping from the Visual-Polar space to screen space is

$$25.09\rho^4 + 0.1680\rho^3 + 27.61\rho^2 + 23.87\rho + 3.232. \qquad (3)$$

This scaling is also visualized in the vertical axis of Figure 3.

On top of the visual acuity based scaling in the ρ-axis, we also scale the ρ-axis so that we keep the number of rendered paths constant no matter where the user is looking at on the screen. This is done by finding the greatest distance from the gaze point to the screen corners. The distance is used to scale the axis so that the maximum ρ is equal to it. This scaling dynamically changes the size of the fovea to be larger when the user looks at the edges of the vision, which compensates for the typically worse eye tracking accuracy in these areas [RWH*17].

For the path tracing itself we render a regular grid of pixels in the Visual-Polar (ρ, ϕ) space. When we compute the origins and directions for the primary rays, we do the mapping from the Visual-Polar space to screen space and then compute the origin and direction as is typically done in conventional screen space path tracing.

3.3. Denoising

An interesting feature of real-time path tracing denoisers like SVGF and BMFR is their use of temporal data that also reduces temporal flickering caused by the changing sample locations in the peripheral parts of the foveated rendering. We found that adapting these state-of-the-art real-time denoisers to the Visual-Polar space requires only minor modifications to them, which we describe in the following.

Out-of-bounds access handling: All the sampling in Visual-Polar space needs to be wrap around on the ϕ-axis. On the ρ-axis we used clamp-to-edge edge handling, but one could also use proper edge handling which rotates the sample to its correct position on the other side of the gaze point.

Temporal accumulation: Accessing temporal data needs to take the Visual-Polar space into account. Conventionally, the temporally-aware denoisers calculate from the 3D world positions of the current frames which screen space location they need to access in the previous frame. We simply apply the screen space to Visual-Polar space transformation to these locations and access data with bilinear sampling. When performing bilinear interpolation from four samples in the Visual-Polar space frame it needs to be taken into account that the height of the ϕ-axis can be different on both of the accessed columns. As in the original implementations of the denoisers, we decide separately for each sample whether to discard it due to, e.g., disocclusions.

Denoise filter sampling: With SVGF bilateral sampling and A-Trous sampling is done so that we consider the possibly smaller resolution in the ϕ-axis. With BMFR we do not scale the blocks based on the smaller resolution, but instead we reshape the Visual-Polar space so that the height of the ϕ-axis is always the same within one block column area. Synching the clipping with the BMFR blocks produces a staircase-like clipping boundary, which can be seen in Fig 4b. In addition, we keep the location of the clipping boundary synced with BMFR's constantly pseudo-randomly changing block locations.

Since the Visual-Polar space maps more pixels to the gaze point, it naturally scales the screen space A-Trous blur radius of SVGF and the screen space block size of BMFR. BMFR block size in screen space can be seen in Figure 4c. A smaller screen space area with the same number of path tracing samples in the Visual-Polar space enables the denoisers to produce better quality results with more difficult cases like reflections, which can be seen in Figure 5. The case is challenging for the denoisers, because the data of the world seen in the reflections is not in any way present in the feature buffers (G-buffer). For example, if the phenomenon BMFR tries to reconstruct is not present in the feature buffers, it reconstructs the result from gradient-like data available, e.g., in the world position buffer. Reconstructing a detailed sharp reflection from gradients yields a blurred result. In our experiments, SVGF performs better with sharp reflections, because it also uses color data in guiding the blurring. However, with the natural size reduction caused by the Visual-Polar space both of the denoisers have good quality in fovea. In the Visual-Polar space the size stays the same and, therefore, there is always the same number of path tracing samples affecting the denoising of a single pixel.

3.4. Mapping to Screen Space

The Visual-Polar space image can be mapped to screen space in one pass without synchronization. The mapping can be done with backwards projection, i.e., every screen space pixel samples their color from the Visual-Polar space pixels. Also in this case we handle out-of-bounds accesses with wrap around on the ϕ-axis and clamp-to-edge edge on the ρ-axis. We found that the wrap around on the ϕ-axis hides the discontinuity of the Visual-Polar space efficiently.

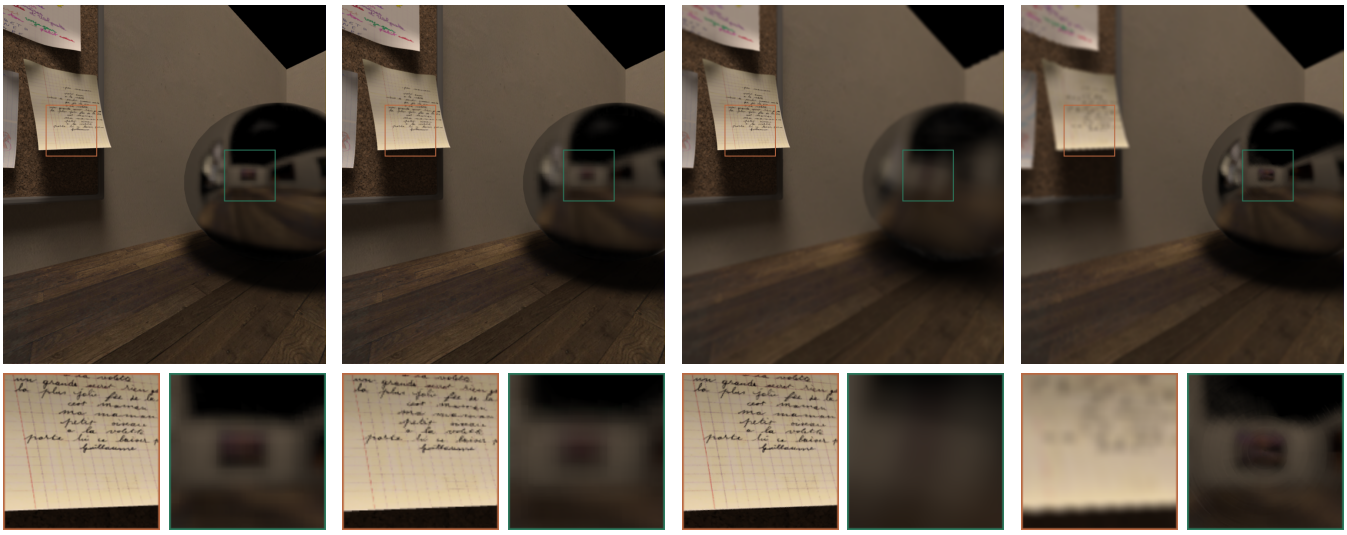
On ρ-axis the mapping is always either one-to-one or magnification. Therefore, bilinear sampling is enough for sufficient quality. In contrast, on ϕ-axis one screen space pixel can map to many Visual-Polar space pixels. Unlike in previous work, for the trilinear mapping from Visual-Polar space to screen space it is enough to generate just mipmaps on one axis of the smaller resolution image. Therefore, we use custom mipmap generation and sampling code, which generates mipmaps only on the ϕ-axis. Our unoptimized version of the code with the test setup takes less than 0.5 ms. This code could be highly optimized since the mipmaps are only required in the fovea area and the sampling positions can be precomputed.

3.5. Blur

After mapping to the screen space, we apply a moderate Gaussian blur. The idea is to blur just enough to remove most of the spatial aliasing problems but not too much to cause tunnel vision. We used zero blurring on the fovea area, and after the edge of the fovea we linearly increase the amount of blurring until the eccentricity angle of 30 degrees where the amount of blur is $\sigma = 6.8$ pixels (resolution 1280×1440). For performance reasons the blur was implemented in two separated passes. In our experiments, we decided to use linear falloff over visual acuity function with the parameters, because it gave more intuitive control over the parameters.

4. Experiments

This section describes the different user tests we ran with the Visual-Polar rendering. The results of each experiment are listed at the end of each subsection. The purpose of the experiments was to test different parameters for the foveation methods and to compare them with each other.

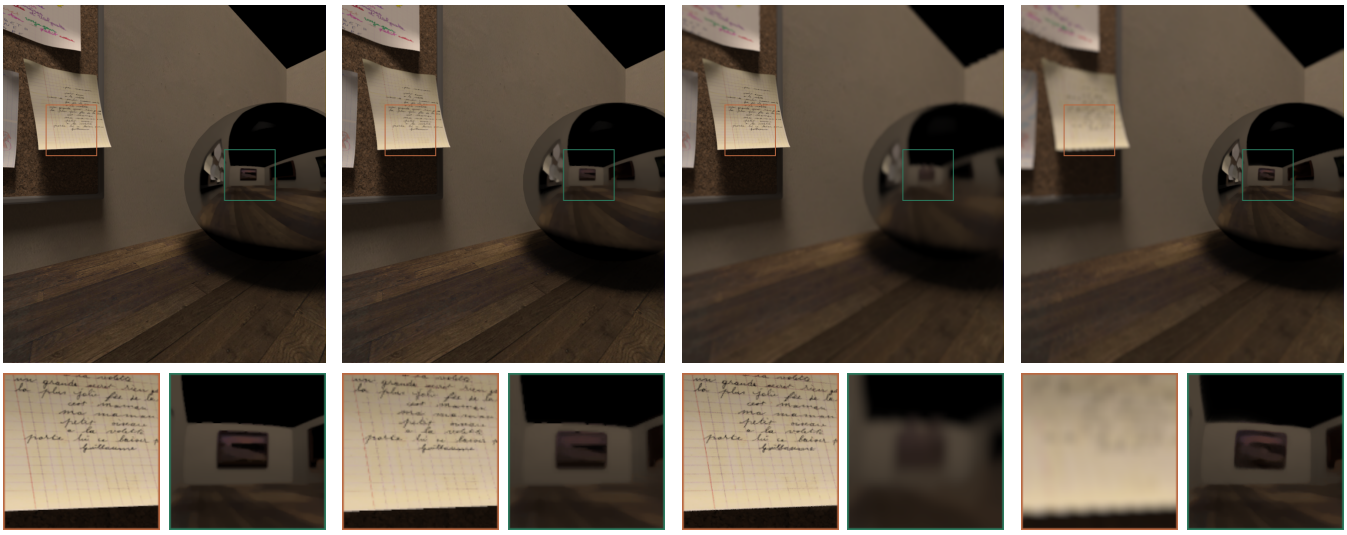


(a) *Screen space BMFR with full resolution*

(b) *Screen space BMFR with the same number of samples as in Visual-Polar*

(c) *Visual-Polar BMFR, with the gaze point in the middle of the left square*

(d) *Visual-Polar BMFR, with the gaze point in the middle of the right square*



(e) *Screen space SVGF with full resolution*

(f) *Screen space SVGF with the same number of samples as in Visual-Polar*

(g) *Visual-Polar SVGF, with the gaze point in the middle of the left square*

(h) *Visual-Polar SVGF, with the gaze point in the middle of the right square*

Figure 5: *Example on how the denoisers can preserve reflections and details better when they are applied in the Visual-Polar space.*

4.1. Test Set-Up

In the experiments we used the FOVE 0 HMD which is equipped with an eye tracker. The computer used in the experiments has a single AMD Vega Frontier Edition GPU, an Intel Core i7-6700K CPU, and 32GB of memory.

Even though the display of the FOVE 0 has only 1280×1440 pixels per eye, the driver requires a 1792×2016 frame per eye because it performs warping automatically. Our test system renders the actual resolution of the display for one eye and shows the same image for both eyes. In other words, we do not have binocular disparity. In addition, upsampling and warping our 1280×1440 frame to the internal representation of the driver introduces some minor sampling artifacts to the results. We decided to render the actual resolution for just one viewpoint because it reduces the required path tracing and denoising by 74%, which allowed us to use more complicated scenes even without dedicated ray tracing hardware acceleration.

The eye tracking accuracy of the system is roughly ± 1 degrees [WRHS18b]. The minimum latency of the system is at least 38 ms from eye movement to rendered pixels [KIV*18]. The users of a foveated rendering system can tolerate up to 70 ms end to end latency [APLK17].

(a) *Music room* **(b)** *Gray white room*

(c) *Dining room* **(d)** *Breakfast room*

(e) *Staircase* **(f)** *Pabellon*

Figure 6: *Example views of typical directions the participants decided to look at in the sampling experiment.*

4.2. Sampling Experiment

The purpose of this test was to compare different parameter sets of the proposed method to a full screen space resolution and this way find good parameters to use in the latter path tracing experiment. As the used hardware cannot path trace and denoise the full screen space resolution of 1280×1440 with a reasonable frame rate, the test was done by sampling prerendered converged omnidirectional images. Prerendering was done by path tracing equirectangular images. The images were fully converged so only the sample location changes might cause temporal artifacts. These artifacts are similar to the artifacts that the denoisers are generating, since they discard temporal data projected from other objects. For this experiment we removed the translation of the camera, because it would have required either many equirectangular images or reprojection and filling with depth information. For the full resolution screen space rendering, the equirectangular images were sampled in real-time with the full Cartesian resolution of the HMD. For the proposed method we sampled the prerendered images from the same positions from which path tracing with that distribution would have sampled the world.

4.2.1. Procedure

In the user study we showed the participants a pair of two renderings of the same scene and their task was to compare the quality of them. In case different quality was experienced, the users were asked to choose which one of the renderings was better. In every pair one of the renderings was a full resolution Cartesian rendering. The other one was the result of a randomly picked parameter set of the proposed method. The order of the two renderings in the pair was randomly selected.

The parameter sets in the study where different sample reductions and different blurring parameters. These sets where found out by testing the system with the authors first during the development. After the trial runs, some participants noticed that one of the renderings in the pairs was always the full resolution Cartesian rendering. To address this, we added random pairs of two foveated renderings of the same method. These random pairs were not used in the results. The idea was to get the users to focus on the quality and not to find ways to “cheat the test” by finding out if the rendering reacted to eye-tracking by quickly moving their eyes back and forth multiple times.

We used six different scenes, which are shown in Figure 6. As can be seen in the images, we included many hard cases where the scene contains patterns that are intentionally almost checkerboard patterns. The test lasted for approximately 40 minutes per test subject. In the experiment we had eight participants all with normal or corrected to normal vision.

4.2.2. Results

The answer distribution of the questionnaire of the sampling experiment can be seen in Figure 7. The names of the methods tell how much path tracing and denoising work was reduced. For example, 61% is achieved by reducing both the the width and the height of the resolution by 37%. The answers show that we can reduce the rendered pixels by approximately 60-70%, after which the users start to see too many artifacts. A possible cause for the lack of "Better than reference" answers in the 61% reduction option is that different blur settings work the best with different reductions. According to paired t-tests, the 88% reduction showed significant difference in users’ responses compared with both of the smaller reductions (p-value < 0.05). Between 61% and 71% reductions, no significant difference was found. Based on these results, in the second test described below, we decided to use 61% reduction since we wanted to be conservative with our parameter choices.

It is also important to note that this test did not utilize temporal reprojection of samples. In other words, there were different temporal artifacts, caused by the moving gaze point and moving HMD, compared to results where a temporal-aware denoiser is used. Therefore, the results should not be treated as absolute numbers but only compared to each other. However, the results are very

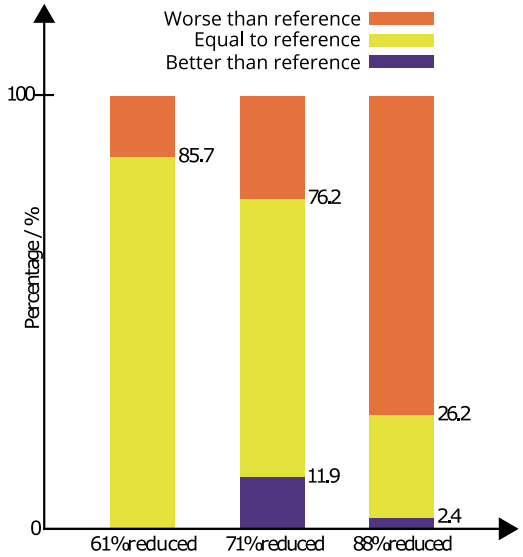
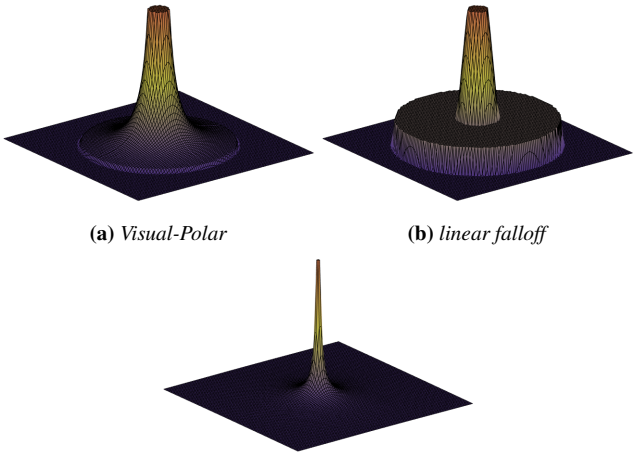


Figure 7: *Answer distribution of the sampling experiment with the best performing blurring settings for every reduction.*



(a) *Visual-Polar* **(b)** *linear falloff*

(c) *log-polar*

Figure 8: *Sample distribution around the gaze point with different methods in the path tracing experiment. Linear falloff requires a lot more samples in the periphery and, therefore, we cannot afford to give it as many samples in the fovea with our real-time budget. In reality the spike of the log-polar distribution is orders of magnitude higher than the highest values in other distributions.*

similar and this is the closest imposter of 1 spp denoised full Cartesian resolution we can run in real time. The main idea of this experiment was to get blurring and sample count reduction parameters for the real-time path tracing where we have temporal reprojection in the denoisers.

One of the participants marked every single time the foveated rendering as worse than Cartesian rendering which did not help with relative comparisons of the tested methods. In other words, this participant thought that the renderings are the same 0% of the time, and in contrast, the other participants thought 86% of the time that the renderings are the same. Since the behavior was a clear outlier we concluded that the eye tracking did not work properly for this one user and removed him/her from the results. Different eye-tracking behavior could be caused by astigmatism this participant had in his/her other eye.

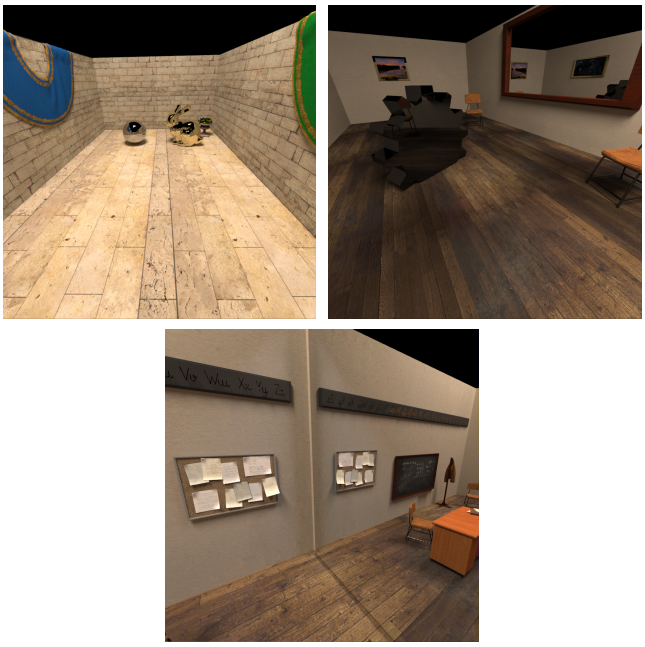
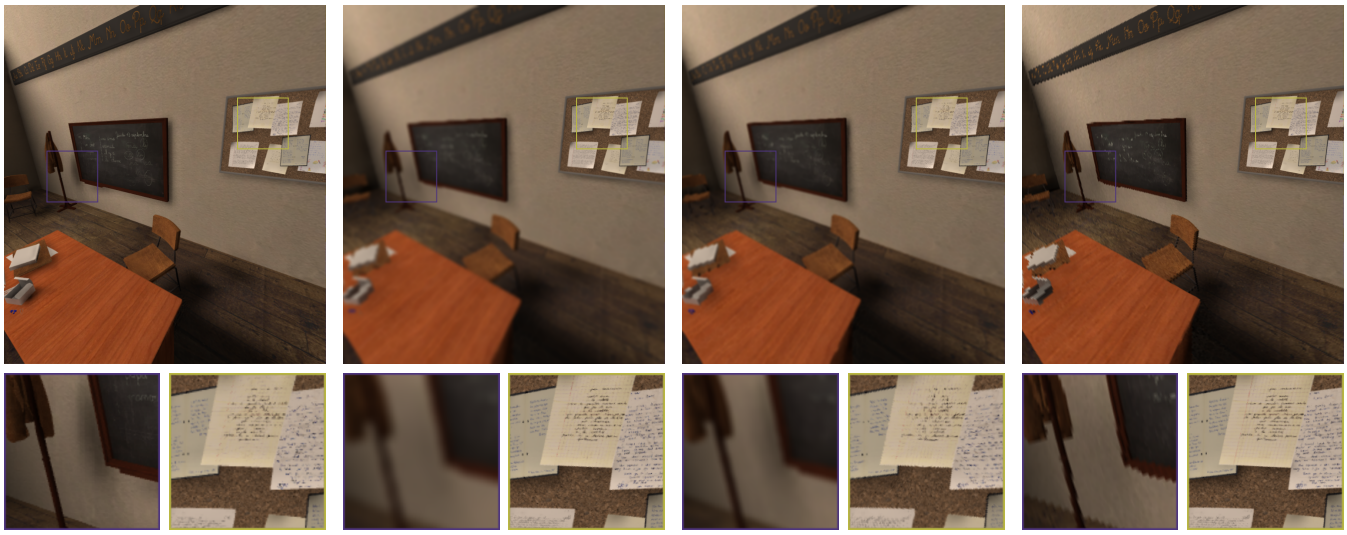


Figure 9: *Three scenes used in the path tracing experiment.*

4.3. Real-Time Path Tracing Experiment

The purpose of the second experiment was to compare different sample distributions in a path tracing scenario with with the same full resolution 1280×1440 as in the previous user study, but with a number of samples per pixel that can be rendered in real-time. In other words, we had to leave full resolution screen space rendering out of this test. The compared sample distributions were 1) uniform screen space, 2) linear falloff as described in [WRK*16], 3) log-polar [MDZV18], and 4) the proposed Visual-Polar distribution. The sample distribution around the gaze point of each of the foveated methods can be seen in Figure 8.

In the uniform distribution case, we used the same number of rendered pixels as in the proposed method because it started to reach the limit of how many path tracing samples we can produce in real-time on the test machine. In the linear falloff we used a sampling probability of 20% in the peripheral parts, fovea radius of 10 eccentricity degrees and periphery starting at 20 eccentricity degrees. This means it has a larger fovea and a higher sampling probability in the periphery. For this reason, we gave it $1.5\times$ more samples compared to the Visual-Polar method. To make the comparison fairer we also added blurring to the linear falloff method. In log-polar we rendered directly in log-polar space and used the parameter setup ($\sigma = 1.8$ $\alpha = 4.0$) described by the original paper [MDZV18]. To make the comparison fair we rendered the same



(a) *Screen space* **(b)** *Visual-Polar* **(c)** *Linear falloff* **(d)** *Log-polar*

Figure 10: *Different methods tested in the path tracing experiment.*

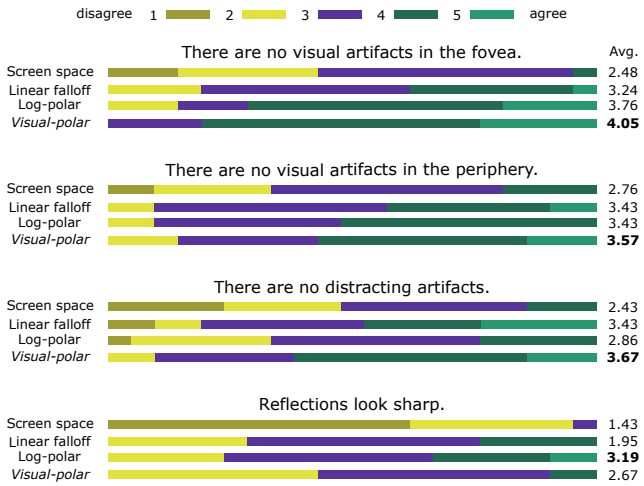


Figure 11: *Answer distribution of the path tracing user study.*

number of pixels with the proposed method as in the log-polar method. In this experiment we used BMFR with all the methods because it is faster than SVGF. Using just one denoiser reduced the number of parameter permutations in the test making it shorter.

4.3.1. Procedure

In the user study, we rendered a scene with a random rendering method and asked the participants to rate on a five-point Likert scale how much they agree with the following statements.

1. There are no visual artifacts in the fovea.
2. There are no visual artifacts in the periphery.
3. There are no distracting artifacts.
4. Reflections look sharp.

Table 1: *P-values from Mann-Whitney U test for every question in the real-time path tracing experiment. The comparison is against the proposed method on every row.*

Comparison	1	2	3	4
Cartesian	**8.2e-07**	**0.0093**	**3.8e-04**	**3.6e-06**
Linear falloff	**0.0023**	0.53	0.59	**0.0049**
Log-polar	0.41	0.58	**0.0059**	**0.060**

In question (1) the users were asked to assess the quality at their gaze point. The idea of question (2) was to measure both temporal artifacts and tunnel vision. In question (3) both the fovea and the periphery were considered. Finally we added question (4), because we wanted to measure the better quality in the reflections visualized in Figure 5.

In total we had three scenes which are seen in Figure 9 and four different rendering methods shown in Figure 10. The scenes were chosen so that they contain different kinds of reflective objects and not too much geometry, just to make sure that without dedicated ray tracing hardware the ray traversal is not the bottleneck. The test took around 35 minutes to complete depending on how quickly the participant decided their rating. In the experiment we had seven participants all with normal or corrected to normal vision.

4.3.2. Results

The answer distribution of the path tracing experiment can be seen in Figure 11. The proposed method has the best average answer in all other questions than the reflection sharpness related question (4). In that question log-polar is better because it packs so many samples to the center of the gaze point as can be seen in Figure 8. Therefore, it is able to generate very sharp reflections on small area around the gaze point. Sampling the same number of samples with

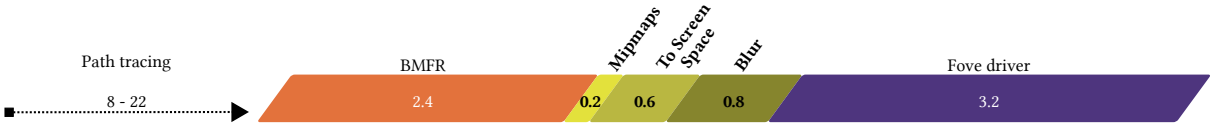


Figure 12: *The latency of different pipeline stages (ms) for screen space resolution of* 1280×1440 *and Visual-Polar space resolution of* 853×960*. Both path tracing and denoising (BMFR) are* $2.5\times$ *faster due to reduced resolution in the Visual-Polar space. Yellow parts are the ones added by the proposed method.*

the regular grid in screen space is clearly the worst of all the benchmarked methods. It also got bad ratings on the question (2) about the peripheral quality even though it has more samples in the periphery than other methods. The poor peripheral rating is likely caused by the poor resolution in the fovea, which made the users think that the rendering is overall bad.

We also performed paired Mann-Whitney U tests for the results between the proposed method and the other tested methods. The p-values for the statistical significance tests are shown in Table 1. Significant difference was found for each question between the proposed method and the same resolution Cartesian. Compared with Linear falloff, significant difference was found in questions (1) and (4). Significant difference with Log-polar was found in questions (3) and (4).

5. Execution Time

The execution time breakdown of our Visual-Polar rendering pipeline with the test machine can be seen in Figure 12. On our single consumer GPU computer without dedicated ray tracing hardware it takes 8-22 ms to path trace the scene at 1 spp in the Visual-Polar space resolution of 853×960, which maps to a 1280×1440 screen space resolution. The path tracing execution time varies a lot because it is heavily dependent on the scene. The Visual-Polar space saves 61% of the path tracing execution time. In other words, path tracing full screen space resolution of 1280×1440 would take around 20-56 ms, which is too much for the real-time frame budget.

The denoising takes only 2.4 ms in the Visual-Polar space, and the space also saves 61% of the denoising work. The extra steps added by the proposed method on top of path tracing are generating mipmaps (0.2 ms), mapping to the screen space (0.6 ms) and blurring (0.8 ms). These execution times are measured with our unoptimized OpenCL code. All in all, the Visual-Polar space saves 16-38 ms on path tracing and denoising while adding only about 1.6 ms.

6. Ray Coherence

In addition to the sample reduction, the Visual-Polar space also speeds up ray traversal because the primary rays are coherent and the SIMT/SIMD lanes have high utilization. On our test hardware if we ray trace coherent rays by completely randomizing the starting points as in [KIV*18] the ray traversal is approximately $1.5\times$ slower. This affects the ray traversal, which is typically 20% of the path tracing execution time on our test machine. In addition, if collisions are not prevented, there would be an additional cost of summing the samples using atomic operations and handling the unsampled locations.

On the other hand, a typical way to do linear falloff defined by sampling probability is to launch path tracing with full resolution and kill paths randomly in periphery. With random killing the ray traversal for the same number of samples is approximately $1.3\times$ slower compared to the Visual-Polar space.

7. Limitations and Future Work

We evaluated using contrast enhancement [PSK*16] both after the blur pipeline stage and as a replacement of it. However, we were not able to find parameters that would not make the users see less artifacts in the periphery. This was likely caused by the fact that we perform TAA [Kar14] in the Visual-Polar space and not after the contrast enhancer. We also tested applying TAA after mapping to screen space, but without modifications, it jitters the sample locations too much in the fovea and too little in the periphery. As a future work, it would be interesting to design a dedicated TAA method for Visual-Polar space rendering, which is applied after mapping to the screen space and therefore can be run after the contrast enhancer.

One drawback of Visual-Polar space is that adding fine-grained screen space sampling strategies, as in [SGEM16], would require sparse sampling with increased resolution. Sparse sampling was the thing we wanted to get rid of in the first place to fully utilize the SIMD/SIMT hardware.

Distortion correction (the counter operation of the HMD lenses) of the FOVE 0 is a black box to us, which we cannot disable. We provide the driver with an image rendered to a plane. In the future, it would be interesting to generate primary rays in the already warped Visual-Polar space, potentially further reducing the sampling requirement and eliminating the need for a separate distortion correction step. This would require an HMD driver which allows directly displaying the rendered pixels on the screen of the device.

The Visual-Polar space execution time could be further optimized by not rendering and denoising areas on the far edges of the vision, which are clipped by the viewport of the HMD. For this implementation the denoisers should handle these areas so that the outside data does not bleed into the viewport area.

8. Conclusions

We proposed the Visual-Polar space, which produces visually pleasing foveated real-time path tracing. The Visual-Polar space coordinates are a modification of the polar coordinates, so that the

sample distribution follows the visual acuity model of the human visual system. The wanted distribution is achieved by scaling and cutting the polar space. Compared to the previous work on polar foveated rendering our sample distribution follows human eye resolution more closely. This leads to getting more out from the very limited real-time path tracing sample budget.

Another key benefit is the ability to do both rendering and denoising in the Visual-Polar space making both of the stages $2.5\times$ faster. In addition, the primary rays are coherent, which contributes to improved SIMD/SIMT hardware utilization, ending up with a $1.3-1.5\times$ faster ray traversal. We proposed the first system that directly path traces in a modified polar space and demonstrates that also denoising can be done directly in the space making both faster.

The Visual-Polar foveated path tracing was evaluated in two user studies. In the first study we compared different parameters for the proposed sampling pattern. In the second user study we compared the proposed method with other foveated path tracing sampling patterns. The proposed method had the best average answer in all artifact related questions.

To the best of our knowledge, our system is the first one that can do path traced and denoised foveated rendering in real-time for contemporary VR headset resolutions.

Acknowledgements

The authors would like to thank anonymous reviewers and Julius Ikkala for fruitful comments. In addition, the authors are thankful to user test participants and model makers: Christophe Seux for Class room (CC0), Stanford 3D scanning repository for Stanford Bunny, Frank Meinl for Crytek Sponza (CC-BY), Wig42 for Music room (CC-BY), Gray white room (CC-BY), Dining room (CC-BY), Breakfast room (CC-BY), Staircase (CC-BY), and Hamza Cheggour for Pabellon (CC-BY).

The work was financially supported by the Tampere University (of Technology) Graduate School, Emil Aaltonen Foundation, Finnish Foundation for Technology Promotion, Nokia Foundation, Business Finland (funding decision 40142/14, FiDiPro-StreamPro), Academy of Finland (funding decisions 297548, 310411) and ECSEL JU project FitOptiVis (project number 783162).

References

[APLK17] ALBERT, RACHEL, PATNEY, ANJUL, LUEBKE, DAVID, and KIM, JOOHWAN. "Latency requirements for foveated rendering in virtual reality". *Transactions on Applied Perception* 14.4 (2017) 6.

[Bar18] BARRÉ-BRISEBOIS, COLIN. *Game Ray Tracing: State-of-the-Art and Open Problems*. High Performance Graphics Keynote. 2018 2, 3.

[BRM*16] BITTERLI, BENEDIKT, ROUSSELLE, FABRICE, MOON, BOCHANG, et al. "Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings". *Computer Graphics Forum*. Vol. 35. 4. 2016 2.

[Dev86] DEVROYE, LUC. *Non-Uniform Random Variate Generation*. Springer, 1986 4.

[GFD*12] GUENTER, BRIAN, FINCH, MARK, DRUCKER, STEVEN, et al. "Foveated 3D graphics". *Transactions on Graphics* 31.6 (2012) 2, 3.

[HEMS10] HERZOG, ROBERT, EISEMANN, ELMAR, MYSZKOWSKI, KAROL, and SEIDEL, H-P. "Spatio-temporal upsampling on the GPU". *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 2010 1.

[HGF14] HE, YONG, GU, YAN, and FATAHALIAN, KAYVON. "Extending the graphics pipeline with adaptive, multi-rate shading". *Transactions on Graphics* 33.4 (2014) 1.

[Kaj86] KAJIYA, JAMES. "The rendering equation". *ACM Siggraph Computer Graphics*. Vol. 20. 4. 1986 2.

[Kar14] KARIS, BRIAN. "High-quality Temporal Supersampling". *ACM SIGGRAPH 2014, Advances in Real-Time Rendering in Games*. 2014 10.

[KIM*19] KOSKELA, MATIAS, IMMONEN, KALLE, MÄKITALO, MARKKU, et al. "Blockwise Multi-Order Feature Regression for Real-Time Path Tracing Reconstruction". *accepted to Transactions on Graphics* (2019) 2, 3.

[KIV*18] KOSKELA, MATIAS, IMMONEN, KALLE, VIITANEN, TIMO, et al. "Instantaneous foveated preview for progressive Monte Carlo rendering". *Computational Visual Media* (2018) 3, 4, 6, 10.

[KMSB18] KILGARIFF, EMMETT, MORETON, HENRY, STAM, NICK, and BELL, BRANDON. *NVIDIA Turing Architecture In-Depth*. https://devblogs.nvidia.com/nvidia-turing-architecture-in-depth/ accessed 8th of January 2019. 2018 2, 3.

[KVJT16] KOSKELA, MATIAS, VIITANEN, TIMO, JÄÄSKELÄINEN, PEKKA, and TAKALA, JARMO. "Foveated path tracing". *Proceedings of International Symposium on Visual Computing*. 2016 2, 3.

[LW90] LEVOY, MARC and WHITAKER, ROSS. "Gaze-directed volume rendering". *ACM SIGGRAPH Computer Graphics*. Vol. 24. 2. 1990 2.

[MDZV18] MENG, XIAOXU, DU, RUOFEI, ZWICKER, MATTHIAS, and VARSHNEY, AMITABH. "Kernel Foveated Rendering". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018) 3, 8.

[PSK*16] PATNEY, ANJUL, SALVI, MARCO, KIM, JOOHWAN, et al. "Towards foveated rendering for gaze-tracked virtual reality". *Transactions on Graphics* 35.6 (2016) 2, 10.

[Red97] REDDY, MARTIN. "Perceptually modulated level of detail for virtual environments". PhD thesis. College of Science and Engineering, University of Edinburgh, 1997 2.

[RFS18] RITSCHEL, TOBIAS, FRISTON, SEBASTIAN, and STEED, ANTHONY. "Perceptual Rasterization for Head-mounted Display Image Synthesis". *arXiv preprint arXiv:1806.05385* (2018) 3.

[RWH*17] ROTH, THORSTEN, WEIER, MARTIN, HINKENJANN, ANDRÉ, et al. "A Quality-Centered Analysis of Eye Tracking Data in Foveated Rendering". *Journal of Eye Movement Research* 10.5 (2017) 5.

[SCMP19] SIEKAWA, ADAM, CHWESIUK, MICHAŁ, MANTIUK, RADOSŁAW, and PIÓRKOWSKI, RAFAŁ. "Foveated Ray Tracing for VR Headsets". *International Conference on Multimedia Modeling*. 2019 3.

[SGEM16] STENGEL, MICHAEL, GROGORICK, STEVE, EISEMANN, MARTIN, and MAGNOR, MARCUS. "Adaptive image-space sampling for gaze-contingent real-time rendering". *Computer Graphics Forum*. Vol. 35. 4. 2016 2, 10.

[SKW*17] SCHIED, CHRISTOPH, KAPLANYAN, ANTON, WYMAN, CHRIS, et al. "Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination". *Proceedings of High Performance Graphics*. 2017 2, 3.

[SPD18] SCHIED, CHRISTOPH, PETERS, CHRISTOPH, and DACHSBACHER, CARSTEN. "Gradient Estimation for Real-Time Adaptive Temporal Filtering". *Proceedings of the Computer Graphics and Interactive Techniques* 1.2 (2018) 2.

[SRJ11] STRASBURGER, HANS, RENTSCHLER, INGO, and JÜTTNER, MARTIN. "Peripheral vision and pattern recognition: A review". *Journal of vision* 11.5 (2011) 2.

[VST*14] VAIDYANATHAN, KARTHIK, SALVI, MARCO, TOTH, ROBERT, et al. "Coarse pixel shading". *Proceedings of High Performance Graphics*. 2014 1.

[WRHS18a] WEIER, MARTIN, ROTH, THORSTEN, HINKENJANN, ANDRÉ, and SLUSALLEK, PHILIPP. "Foveated depth-of-field filtering in head-mounted displays". *Transactions on Applied Perception* 15.4 (2018) 3.

[WRHS18b] WEIER, MARTIN, ROTH, THORSTEN, HINKENJANN, ANDRÉ, and SLUSALLEK, PHILIPP. "Predicting the gaze depth in head-mounted displays using multiple feature regression". *Proceedings of the Symposium on Eye Tracking Research & Applications*. 2018 6.

[WRK*16] WEIER, MARTIN, ROTH, THORSTEN, KRUIJFF, ERNST, et al. "Foveated Real-Time Ray Tracing for Head-Mounted Displays". *Computer Graphics Forum* 35.7 (2016) 3, 8.

[WSR*17] WEIER, MARTIN, STENGEL, MICHAEL, ROTH, THORSTEN, et al. "Perception-driven Accelerated Rendering". *Computer Graphics Forum*. Vol. 36. 2. 2017 2.

[WWHW97] WATSON, BENJAMIN, WALKER, NEFF, HODGES, LARRY F, and WORDEN, AILEEN. "Managing level of detail through peripheral degradation: Effects on search performance with a head-mounted display". *Transactions on Computer-Human Interaction* 4.4 (1997) 2.

[XLV18] XIAO, KAI, LIKTOR, GABOR, and VAIDYANATHAN, KARTHIK. "Coarse pixel shading with temporal supersampling". *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 2018 1.