Generalized Light Portals

SHINJI OGAKI

Fig. 1. Conference room rendered with environment map sampling and cosine-lobe sampling combined via MIS (left) and the three sampling techniques (environment, cosine-lobe, and our portal sampling) combined via MIS (right). The star-shaped windows are used as light portals. Each image is rendered in 100 seconds. Our portal sampling achieves significant noise reduction.

Light portals are useful for accelerating the convergence of Monte Carlo path tracing when rendering interiors. However, they are generally limited to flat polygonal shapes. In this paper, we introduce a new concept that allows existing polygon meshes with arbitrary shaders in a scene to be used as generalized light portals. We also present an efficient sampling method that takes into account the pixel values of the environment map and ray guiding two-dimensional textures that are typically opacity or transparency maps. This novel sampling strategy can be combined with other sampling techniques by using multiple importance sampling.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Ray tracing**.

Additional Key Words and Phrases: illumination, ray tracing, image based lighting, sampling

ACM Reference Format:
Shinji Ogaki. 2020. Generalized Light Portals. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2, Article 10 (August 2020), 19 pages. https://doi.org/10.1145/3406176

1 INTRODUCTION

Despite recent advances in rendering algorithms, rendering interiors is still time-consuming because light reflects multiple times before reaching the virtual camera. It is prohibitively expensive, especially when the outside light enters through a set of small openings. In such cases, light portals are useful to reduce noise. Major production renderers such as Arnold and RenderMan support this feature [Autodesk, Inc. 2020; PIXAR ANIMATION STUDIOS 2020]. However, the shape of a portal is generally limited to a flat rectangle, and locating such an object to cover (such as a window) is a laborious task. We generalize light portals to facilitate designers' use of arbitrary polygon meshes with shaders (possibly texture-mapped). A typical use case is a curved stained-glass window object that is selected and marked as a light portal.

Author's address: Shinji Ogaki, shinji.ogaki@gmail.com.

We apply an importance sampling technique designed for many lights [Estevez and Kulla 2018] to the portal objects to efficiently sample the areas covered by the light portals. The key difference is the use of a two-level hierarchical data structure. We use a surface-area-oriented heuristic (SAOH) optimized bounding volume hierarchy (BVH) [Estevez and Kulla 2018; Liu et al. 2019] as the top-level acceleration structure (TLAS). For each leaf of the TLAS, we build another N-ary tree data structure using the two-dimensional barycentric coordinates as the bottom level acceleration structure (BLAS). BLASes enable the efficient generation of samples in regions with high transparency controlled by two-dimensional texture maps. Additionally, we describe the advantages of using wide tree data structures for addressing many light sampling problems and introduce performance-improving techniques.

Simply sending rays to light portals leads to suboptimal convergence rates when portals connect indoor and outdoor spaces because the intensity of the environment map is ignored. We also address this issue by probabilistically selecting a node based on the total energy of the environment map pixels covered by the solid angle subtended by the node bounding box while traversing the two-level hierarchical data structure. Our sampling technique, however, is not restricted to environment map sampling. For instance, a lampshade can be used as a light portal to capture light passing through its small, complex holes.

Our goal is not the automatic creation of light portals [Nguyen 2014], but to allow the use of objects that already exist in a scene as light portals with minimum setup cost. We do not consider dynamic scenes, and our sampling technique does not take into account arbitrary BRDFs or visibility because these can be managed by combining with other algorithms [Liu et al. 2019; Moreau et al. 2019; Pantaleoni 2019]. The proposed method is unbiased and does not require a significant effort to implement if a renderer already supports hierarchical importance sampling for many lights and MIP mapping.

2 RELATED WORK

Hierarchical Importance Sampling of Many Lights. Many-light sampling can be efficiently performed using hierarchical importance sampling [Estevez and Kulla 2018; Keller et al. 2017]. The typical implementation is to build a BVH over light-emitting objects and stochastically traverse it from the root. At each level of the traversal, the relative contributions of the child nodes are estimated to determine which node to traverse next. Estevez and Kulla [2018] introduced the surface area orientation heuristic (SAOH), which allows the emitter power to be evenly distributed among the tree and cluster lights with a similar orientation. Moreau et al. [2019] showed that sampling using a SAOH-optimized BVH can be efficiently performed on a GPU, and dynamic light sources can be handled well with a two-level BVH hierarchy [Haines and Akenine-Möller 2019; Moreau et al. 2019]. Liu et al. [2019] extended the method to consider BRDFs and proposed an adaptive heuristic that dynamically determines the number of samples allocated for different sampling techniques. Stochastic Lightcuts [Lin and Yuksel 2020; Yuksel 2019a,b], being an extension of the Lightcuts algorithm that eliminates the sample correlation by replacing it with noise, is similar.

Textured Polygonal Lights. Sik and Krivanek [2013] proposed an algorithm to generate randomly distributed points on a triangular mesh with probability density by using a two-dimensional texture map. They demonstrated that the algorithm can compute the illumination from a triangle mesh light with emission defined by an HDR texture. Their approach is not optimal because the geometric terms are ignored during sampling. Linearly transformed cosines can approximate microfacet BRDFs of arbitrary roughness, enabling analytical integration over arbitrary spherical polygons [Heitz et al. 2016]. When computing the illumination from textured polygonal lights, some error is introduced due to the prefiltering of the area light texture. The lack of shadowing effects was addressed by stochastic ray tracing using a ratio estimator [Heitz et al. 2018]. Okuno and Iwasaki [2019]

stored precomputed visibility in a binary space partitioning tree associated with a polygon mesh vertex. The resolution of the visibility map bounds the quality, and it does not scale to production assets because of its high initialization cost. Belcour et al. [2018] proposed a numerical solution for computing the integral of spherical harmonic expansions clipped to polygonal domains. It can effectively approximate the reflected radiance on glossy surfaces from polygonal light sources. They also showed that the method can be combined with control variates to compensate for the true variation of the occluded geometry. A similar approach was concurrently developed by Wang and Ramamoorthi [2018]. In both methods, complex light sources with different patterns and colors are deconstructed and processed into constituent polygons. Wang and Ramamoorthi [2020] later derived analytic formulae for spherical harmonic gradients that enable scaling Precomputed Radiance Transfer to hundreds of polygonal lights.

Light Portals. Light portals are useful for accelerating the convergence of path tracing, especially for indoor and semi-outdoor scenes. Nguyen [2014] developed an automated approach to detect and construct portals for architectural scenes. The portal-rectified reparameterization of the environment allows samples to be drawn proportionally to the product of the environment map and its visibility through the portal [Bitterli et al. 2015]. While significant variance reduction is achieved, the shape is restricted to a rectangle, and a separate tabulated environment map has to be built for each portal. Alternatively, product sampling of the BRDF and environment through the portal can be achieved by integrating spherical harmonics expansions clipped to polygonal domains [Belcour et al. 2018]. Atanasov et al. [2018] introduced a two-step algorithm for efficient environmental light-sampling that takes into account visibility. In the first learning phase, the visibility information is cached in the camera space. The cache is used to adapt the environment sampling strategy in the second final rendering phase. Anderson et al. [2017] introduced tridirectional path tracing to render a challenging scene, for example, where the camera and the light are placed in separate rooms, and there is only a small aperture connecting the two. Their tridirectional path sampling handles this case by constructing two-vertex portal segments that pass through the small opening. Industry-standard renderers support light portals [Autodesk, Inc. 2020; PIXAR ANIMATION STUDIOS 2020]. However, the shape of a light portal is usually limited to a planar rectangle.

3 IMPORTANCE SAMPLING OF TEXTURED MESH LIGHTS

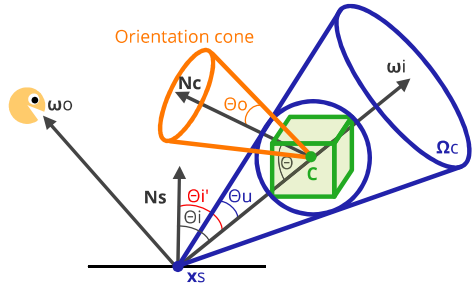


Fig. 2. Geometric setup to compute the contribution from a cluster. We denote $\boldsymbol{x_s}$ the shading point, $\boldsymbol{N_s}$ the normal vector at the shading point, $\boldsymbol{N_c}$ the orientation cone axis, $\boldsymbol{C}$ the center of the cluster, and Ω_c the solid angle of the bounding sphere seen from the shading point. The diameter of the bounding sphere is equivalent to the diagonal of the bounding box of the cluster.

Our portal sampling applies Estevez and Kulla's technique. In this section, we first extend it to support textured mesh lights and describe optimization techniques dedicated to wide BVHs. Following their approach, we build a BVH over light-emitting polygons by recursively partitioning polygons based on the surface area oriented heuristic (SAOH). The SAOH does not only take into account the surface area of a bounding box, but also the normals and intensities of polygons so that emitters with similar orientations are clustered together. During hierarchical sampling, a child node is selected based on the following measure:

$$E\frac{\max\{0,\cos\theta_i'\}}{d^2}\times\begin{cases}\cos\theta' & \text{if } \theta'<\theta_e\\ 0 & \text{otherwise,}\end{cases} \tag{1}$$

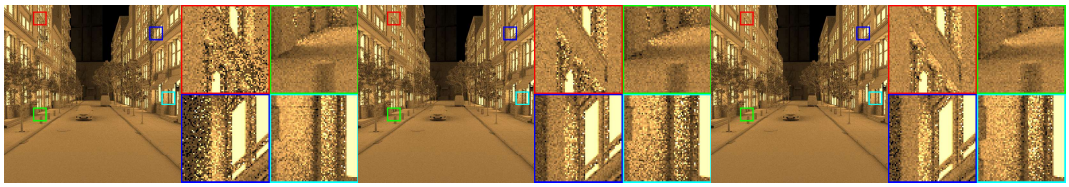
where E is the total energy of the emitters inside the node, d the distance between the shading point $\boldsymbol{x_s}$ and center of the bounding box $\boldsymbol{C}$, and θ_e is the emission profile. The angles θ_i' and θ' are given as

$$\begin{aligned}\theta_i' &= \max\{0,\theta_i-\theta_u\}\\ \theta' &= \max\{0,\theta-(\theta_u+\theta_o)\},\end{aligned}$$

where θ_i is the angle between $\boldsymbol{\omega_i}\left(=\frac{\boldsymbol{C}-\boldsymbol{x_s}}{||\boldsymbol{C}-\boldsymbol{x_s}||}\right)$ and the normal vector at the shading point $\boldsymbol{N_s}$, θ_u the angle of a cone that covers the entire bounding box as seen from $\boldsymbol{x_s}$, θ the angle between $-\boldsymbol{\omega_i}$ and the cluster's orientation $\boldsymbol{N_c}$, and θ_o is the angle of the cluster orientation cone. Fig. 2 illustrates the geometric setup used to compute the measure. We omit the term to represent a BRDF.

3.1 Hierarchical Importance Sampling using Wide BVHs

Utilizing wide vector units such as SSE, AVX, and AVX-512 is a common method to accelerate ray tracing on modern CPUs [Áfra 2013; Dammertz et al. 2008; Fuetterling et al. 2017]. When importance sampling many lights, wide BVHs accelerate computation and reduce noise. Estevez and Kulla [2018] showed that using 4-ary BVHs offers high performance. They noted that quality improvements are due to less stretch caused by sample warping. Noise reduction is achieved by removing nodes with large approximation errors. Collapsing a binary BVH to form a wide BVH can be regarded as creating multiple cuts in the original binary BVH. We observed that 16-ary BVHs yields better results than binary and 4-ary BVHs (Fig. 3). Therefore, we use 16-ary BVH throughout



(a) Binary BVH
32spp
18.44s

(b) 4-ary BVH
32spp
12.10s

(c) 16-ary BVH
32spp
11.48s

Fig. 3. Street scene rendered with (a) binary BVH, (b) 4-ary BVH, and (c) 16-ary BVH. Increasing the arity of BVH leads to better results in general. We used 32 samples per pixel for all images.

the paper because to date AVX-512 is the widest SIMD instruction set on the x86 platform, and we neither perform splitting nor construct a cut. The node data structure used for the TLAS is as follows:

```
template<int32_t N>
struct vec3
{
    float<N> x, y, z; // float<N> packs N floating-points
};

template<int32_t N> // Arity
struct TLASNode // Node of TLAS
{
    int32_t  indices[N];  // references to child nodes
    vec3<N>  aabb_min;    // bounding box min values
    vec3<N>  aabb_max;    // bounding box max values
    vec3<N>  axes;        // cluster orientations
    float<N> cos_theta_o; // cos(theta_o)
    float<N> sin_theta_o; // sin(theta_o)
    float<N> values;      // total cluster energies
};
```

We store one triangle per leaf node, and use the sign bit of each child node reference to specify whether it is a leaf or not. In the sampling process, we build a cumulative distribution from the contribution of the child nodes. With the AVX-512 instruction set, importance sampling can be performed by counting the number of trailing zero bits:

```
__m512 cdf; // cumulative distribution function
__m512 rnd; // random number stored with _mm512_set1_ps()
auto mask  = _mm512_cmp_ps_mask(rnd, cdf, _CMP_LT_OQ);
auto index = _tzcnt_u32(mask);
```

Listing 1. Importance Sampling using AVX-512.

To improve the sampling performance further, we avoid calling inverse trigonometric functions by reformulating Eq. 1 using trigonometric addition formulas. There is no need to compute θ_i' and θ' because we only need $\cos\theta_i'$ and $\cos\theta'$ in the end. We can compute $\cos\theta_i'$ as

$$\cos\theta_i' = \begin{cases} \cos\theta_i\cos\theta_u + \sin\theta_i\sin\theta_u & \text{if } \cos\theta_i < \cos\theta_u \\ 1 & \text{otherwise,} \end{cases}$$

where $\sin\theta_i = \sqrt{1-\cos^2\theta_i}$ and $\sin\theta_u = \sqrt{1-\cos^2\theta_u}$. We can similarly obtain $\cos\theta'$. It is helpful to avoid the use of inverse trigonometric functions, especially when using wide BVHs because vectorized trigonometric functions are not available in the AVX-512 instruction set. Note that the vectorized version of the squared root function `_mm512_sqrt_ps()` is available. (Intel® Short Vector Math Library provides `__mm512_acos_ps()`, which computes the inverse cosine of packed

single-precision floating-point elements. However, this type of function is not provided in other compilers.)

Since we only consider polygon meshes, the emission profile θ_e is always $\frac{\pi}{2}$. Therefore, we do not store θ_e in a BVH node and store $\cos(\theta_o)$ and $\sin(\theta_o)$ instead of θ_o.

3.2 Textured Mesh Lights

Here, we elaborate how to support textured mesh lights. A straightforward approach is to pre-tessellate and pre-shade all the light-emitting polygon meshes similar to Manuka's shade before hit architecture [Fascione et al. 2018], and then perform hierarchical importance sampling. Instead, we adopt a different approach to avoid storing vertex data including positions and texture coordinates of pre-tessellated polygons. Our method uses a two-level hierarchical data structure. The TLAS is a BVH built over the input polygon meshes, as described in the previous subsection. At each leaf node, we create another N-ary tree data structure for each triangle in the barycentric coordinate. This tree structure in the two-dimensional space also helps to reduce the number of bounding boxes and orientation bounds. Constructing BLASes can be trivially parallelized. The node data structure of the BLAS is given as follows:

```
template<int32_t N> // Arity
struct BLASNode // Node of BLAS
{
    uint32_t indices[N]; // references to child nodes
    float<N> values;     // average intensities of texels
};
```

At each node, we only store N indices to its child nodes and N average intensities of texels covered by the sub-triangles in the child nodes. The i-th element is a leaf if its index is out of the range of the node array.

To tessellate triangles adaptively, we leverage Sik and Krivanek's method with a few modifications. Their approach first subdivides each triangle until the sub-triangle size matches the texture resolution, and then merges tessellated triangles if they have the same value. Although efficient sampling can be achieved, merging tessellated triangles is redundant. We instead subdivide triangles only where necessary, with the help of min-max MIP maps (Fig. 4). During the triangle subdivision, we scan MIP maps from coarse to fine levels. If the current sub-triangle falls in a single pixel of a MIP map, and the minimum and maximum values of the pixel are equal, no further subdivision is necessary (Fig. 5). Our method runs roughly two times faster than Sik and Krivanek's method. The speed increased from 268ms to 123ms for the lantern model of Fig. 13. The improvement is subtle compared to the total rendering time; however, it is crucial to reduce the initialization cost to achieve better interactive feedback.

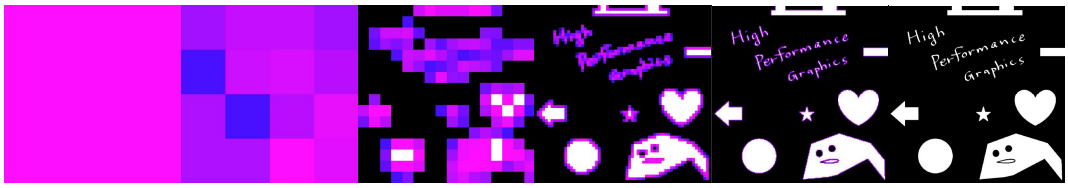


Fig. 4. Example of MIP maps. Minimum values are stored in red, maximum in green, and average in blue. When a triangle is divided into $N = 16$, each edge length of a sub-triangle will be one fourth. Thus, the resolution is quadrupled as the level increases.

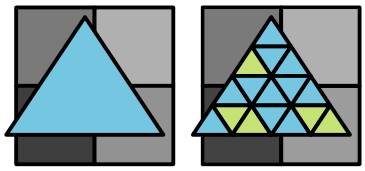
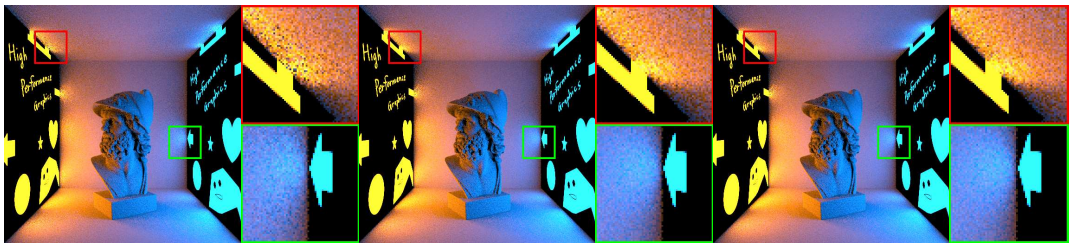


Fig. 5. Avoiding redundant tessellation. Before subdivision a triangle overlaps multiple pixels of the MIP map at the current level. After subdivision five sub-triangles colored in green are completely inside a single pixel. If the minimum and maximum values of the pixel are equivalent, we can stop further subdivision for those triangles and avoid redundant splitting and merging sub-triangles.



(a)$N = 4, \epsilon = 0$ 16spp 1.08s 2.22MB | (b) $N = 16, \epsilon = 0$ 16spp 1.05s 2.66MB | (c) $N = 16, \epsilon = 0.05$ 16spp 1.05s 2.62MB

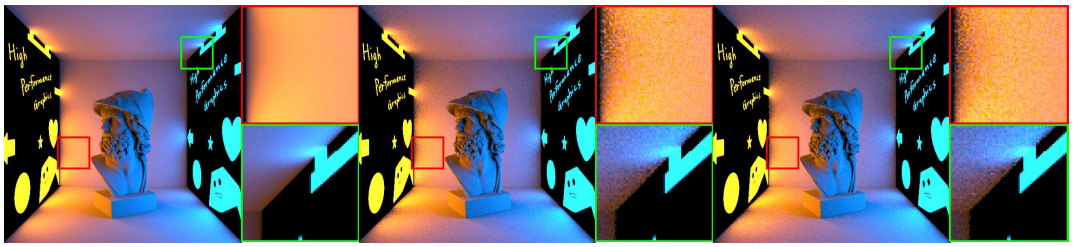
Fig. 6. Each sub-triangle is subdivided into (a) 4 and (b) 16 (c) 16 but with $\epsilon = 0.05$. Using a 16-ary tree for the bottom-level acceleration structure reduces noise. The model is courtesy of Torolf Sauermann.

Although our approach introduces overhead by creating MIP maps, it can be executed very quickly (our non-optimized code took 54ms for a 1024×1024 image), and they can be reused for later renderings and are useful for other purposes. To reduce the number of sub-triangles further, we stop subdividing triangles if the difference between the minimum and maximum values is under a certain threshold ϵ. In our paper we set $\epsilon = 0.05$ to reduce memory consumption (Fig. 6). Note that there is no need to store the minimum and maximum values for practical implementation. We use the sign bit of each pixel value of the MIP maps to specify if the minimum and maximum values are almost identical.

Accurate illumination from a textured polygonal light source can be computed using linearly transformed cosines and the ratio estimator [Heitz et al. 2016, 2018]. However because we aim to develop a technique that can also be used for light portal sampling that takes into account the intensity of the environment map, we use hierarchical importance sampling at the leaf node as well (see Sec. 4). During traversing the BLAS, one sub-triangle is chosen based on its irradiance at each node. Here, we assume that a triangle is textured, and thus computing the exact irradiance from each sub-triangle is expensive. Alternatively, we calculate an approximated irradiance value by using the average color of the texels that are covered by the sub-triangle.

With boundary integration [Heitz 2017], the irradiance from polygonal lights can be obtained as

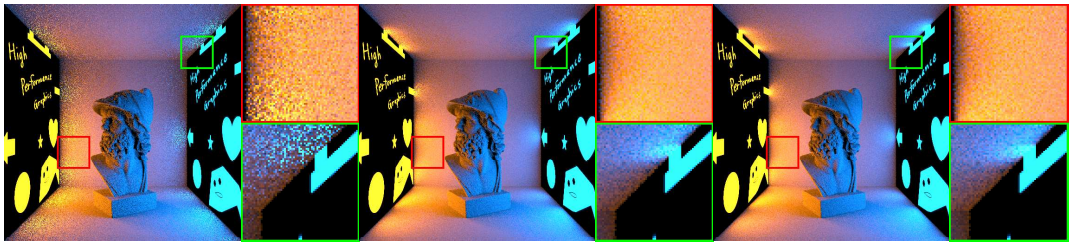
$$I = \frac{L}{2\pi} \sum_{i=0}^{n-1} \gamma_i \cos \delta_i, \tag{2}$$



(a) Reference
-
-

(b) Ours (BI)
16spp
2.37s

(c) Ours (BI) + Cosine-lobe
16spp
3.32s



(d) Sik and Krivanek
16spp
1.01s

(e) Ours (cons.)
16spp
1.06s

(f) Ours (cons.) + Cosine-lobe
16spp
1.54s

Fig. 7. Bust model lit using textured polygonal lights. Only the direct illumination is computed. (a) Reference. (b) Our importance sampling with boundary integration (Eq. 2). (c) Same as (b) but combined with cosine-lobe sampling via MIS. (d) Sik and Krivanek's method. (e) Our importance sampling with the conservative geometric term (Eq. 3). (f) Same as (e) but combined with cosine-lobe sampling via MIS. The images except the reference are rendered using 16 samples per pixel.

where n is the number of vertices, L the radiance, $\gamma_i = \cos^{-1}(\boldsymbol{v}_i \cdot \boldsymbol{v}_j)$, $\delta_i = \boldsymbol{N}_s \cdot \frac{\boldsymbol{v}_i \times \boldsymbol{v}_j}{||\boldsymbol{v}_i \times \boldsymbol{v}_j||}$, $\boldsymbol{v}_i$ is a vertex of a spherical polygon, and the suffix j is $(i+1) \mod n$. We assume that the polygon is clipped to the hemisphere around $\boldsymbol{N}_s$ and $||\boldsymbol{v}_i|| = 1$. When a triangle is small, this computation suffers from numerical error due to the catastrophic cancellation in the cross product. Small triangles are not only created by tessellation but also by clipping a triangle, which is necessary when one or two vertices are beneath the surface. When the solid angle is small, the irradiance can be computed as

$$I \simeq LA \frac{\cos\theta_s \cos\theta_l}{\pi ||\boldsymbol{x}_s - \boldsymbol{x}_l||^2},$$

where A is the surface area of a polygon, and $\boldsymbol{x}_l$ is the barycenter of the polygonal light. By letting $\boldsymbol{N}_l$ be the normal of the polygonal light, $\cos\theta_s = \boldsymbol{N}_s \cdot \frac{\boldsymbol{x}_l - \boldsymbol{x}_s}{||\boldsymbol{x}_s - \boldsymbol{x}_l||}$ and $\cos\theta_l = \boldsymbol{N}_l \cdot \frac{\boldsymbol{x}_s - \boldsymbol{x}_l}{||\boldsymbol{x}_s - \boldsymbol{x}_l||}$. Further, we make the following approximation:

$$I \approx LA \frac{(1+\cos\theta_s)(1+\cos\theta_l)}{4\pi ||\boldsymbol{x}_s - \boldsymbol{x}_l||^2} \tag{3}$$

This extremely conservative term is always non-negative and can be easily vectorized because it involves neither inverse trigonometric functions nor polygon clipping. In bright regions, this approximation generates more noise but is much cheaper to evaluate (Fig. 7 (b) and (e)). Note also that using boundary integration (Eq. 2) does not necessarily provide better results than our approximation (Fig. 7 (b) and (e), the close-ups) because we use the average color of the texels to compute the approximated irradiance. During sampling, we choose a child node by assigning a

probability

$$P_i = \frac{I_i}{\sum_{j=0}^{N-1} I_j} \tag{4}$$

to the i-th child node. When we reach a leaf node of the BLAS, we pick a random point in the selected sub-triangle using the low distortion map between the triangle and square proposed by Heitz [2019].

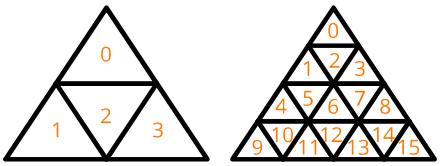


Fig. 8. Our indexing scheme.

To combine our sampling with other sampling strategies such as BRDF sampling, we need to be able to compute the probability density of a ray hitting the given intersection point. When a ray hits a triangle, we deterministically traverse the tree from the root to the leaf node containing the intersection point. This is possible by computing the sub-triangle index i from the barycentric coordinate $\boldsymbol{B} = (b_0, b_1, b_2)$ of the intersection point as

$$i = \beta_0^2 + (\beta_1 - \beta_2) + \beta_0$$

with

$$\beta_j = \min\{\sqrt{N} - 1, \lfloor (1 - b_j)\sqrt{N} \rfloor\}.$$

The range of b_i is $[0, 1]$. Our indexing scheme (Fig. 8) is slightly different from the one proposed by Sik and Krivanek [2013] and works for arbitrary non-zero square numbers ($N = 1^2, 2^2, 3^2, 4^2, \ldots$). Once the index is obtained, we can evaluate the probability with Eq. 4.

4 GENERALIZING LIGHT PORTALS

The previously described textured mesh light sampling can be used to sample light portals by replacing emission values with ray guiding weights, such as transparency. Simply replacing emission by transparency works when the background color is constant. Unfortunately, this leads to a suboptimal convergence rate, for example, when rendering a scene with an environment map where light enters through a small hole. The mapping introduced by Bitterli et al. [2015] can handle this situation very well. However, our goal is to enable the use of arbitrary polygon meshes as light portals, and thus their method cannot be used. To take the environment map into account, we substitute the product of the average transparency and pixel values of the environment map covered by the solid angle of a node into E in Eq. 1 and L in Eq. 3.

We store the environment map using Clarberg's equal-area parameterization, and approximate the average pixel intensity covered by a solid angle with a MIP map (Fig.9). After creating MIP maps, we apply a 3×3 box filter to each level image to avoid an abrupt transition between pixels. (Fig. 9 (rightmost) shows how pixels are connected at the boundary. In the supplemental material, we provide the code that maps a pixel coordinate outside the image to the appropriate one.)

The solid angle Ω covered by each pixel is given as

$$\Omega = \min\left\{4\pi, \frac{(3 \times 3) \times 4\pi}{(2^{level})^2}\right\}. \tag{5}$$

Fig. 9. Example of filtered MIP maps. The size of MIP maps of the environment map (https://hdrihaven.com/) is doubled as the level increases. The rightmost image shows how pixels are connected at the boundary.

The MIP map level can be obtained by inverting this equation. In the actual implementation, it can be effectively computed by counting leading zero bits:

```
auto level = (32 - _lzcnt_u32((uint32_t)(36 * pi / omega))) / 2;
```

Listing 2. MIP map level.

Our sampling method starts by selecting a triangle within the TLAS (Alg. 1). At every traversal step, we compute each child node's illumination contribution and build a CDF (lines 12-25), and select the next node to traverse (lines 28-32). We then update the probability (line 33) and rescale the random value (lines 34-37). We generate a sample point when arriving at the TLAS leaf node by traversing the BLAS corresponding to the selected triangle (Alg. 2). Alg. 2 is almost identical to Alg. 1, except that we keep subdividing the solid angle covered by a sub-triangle even when reaching the BLAS leaf node (lines 38-45). Otherwise, we may fail to capture the bright pixels of the environment map.

At the initialization step, we also generate min-max MIP maps of the environment map and use them to avoid unnecessary traversal steps (Alg. 2, lines 19, 42-43). A 3×3 filter is applied to the min-max MIP maps as well, so that each pixel stores the minimum and maximum values of the 3×3 neighboring pixels. As done in Sec. 3, in the actual implementation, we do not explicitly store minimum and maximum values. Instead, we set the sign bit of each pixel value if the difference between its minimum and maximum values is below ϵ. After reaching the BLAS leaf node, we can immediately terminate the subdivision if the sign bit of the pixel value of the selected MIP map is *true*.

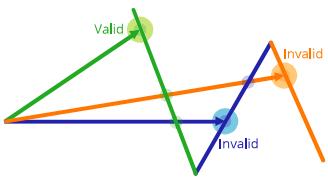


Fig. 10. If a ray hits a triangle to which the sampled point does not belong, the sample should be discarded to compute the probability of hitting the sample accurately.

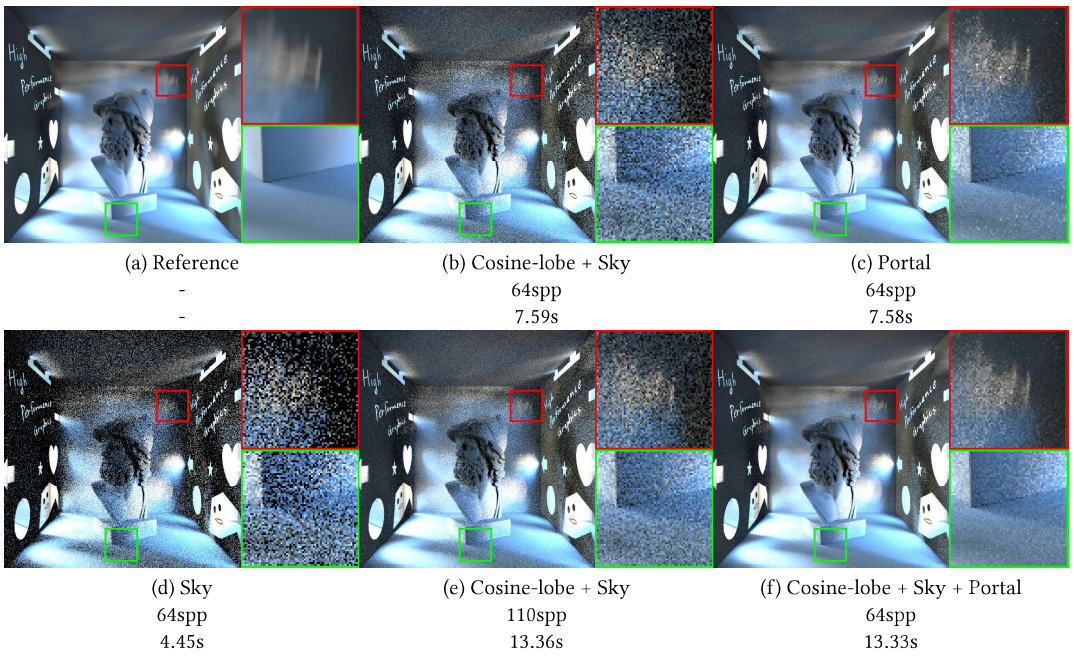
Note that only samples directly visible from the shading point are valid, and those that cannot be seen from the shading point must be discarded (Fig. 10). Otherwise, we cannot compute a proper PDF. To capture the light coming through the portal, we cast a ray from the shading point to the sampled point. The ray traversal is not terminated at the sampled point and continues until it hits an opaque object. When the ray hits a transparent object, the ray throughput is multiplied by its

transparency. On the opaque object, we evaluate the shader and recursively trace the generated rays. If there is no object beyond the sampled point, the ray returns the color of the environment map. This seems inefficient if many portal objects are nested. However, thanks to the measures we apply during hierarchical importance sampling, surfaces close to the shading point are more frequently sampled, and our method works well in practice.

5 RESULTS

We integrated the previously described method in our research renderer. We compiled our code with Clang 9.0.0 and executed tests on Intel® Core ™i7-1065G7. Through the experiment, we used the balance heuristic [Veach 1998] when combining multiple sampling strategies via multiple importance sampling, and we drew the same number of samples from each strategy. We also used blue noise dithered sampling [Georgiev and Fajardo 2016; Peters 2020] to distribute errors in screen space.

Fig. 11 shows a comparison of different sampling strategies. For this scene, our portal sampling (Fig. 11 (c)) achieves notable noise reduction; however, owing to the approximation we used for the measure, spike noise appears. It is suppressed by using our method with cosine-lobe sampling ($PDF(\boldsymbol{\omega}) = \frac{1}{\pi}\langle \boldsymbol{N_s}, \boldsymbol{\omega} \rangle$) and environment map sampling (Fig. 11 (f)). The combination of the three techniques achieved the best overall image quality with a given number of samples. Fig. 1 is another example where our technique achieves significant noise reduction. We also replicated one of Aether's examples [Anderson et al. 2017]. In our experiment shown in Fig. 12, the pinhole is created by a texture map. The scene is challenging for a traditional path tracer. On the other hand,



(a) Reference
-
-

(b) Cosine-lobe + Sky
64spp
7.59s

(c) Portal
64spp
7.58s

(d) Sky
64spp
4.45s

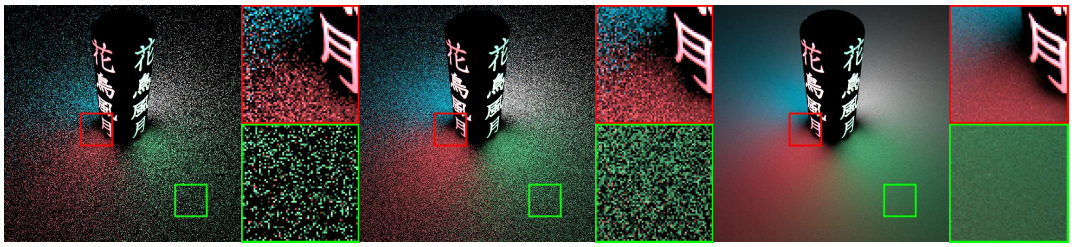
(e) Cosine-lobe + Sky
110spp
13.36s

(f) Cosine-lobe + Sky + Portal
64spp
13.33s

Fig. 11. Bust in box. The left and right walls are trimmed, and light comes from the environment map through the holes. Only the direct illumination is calculated. (a) Reference. (b) Cosine-lobe and environment map sampling combined via MIS. (c) Our portal sampling. (d) Environment map sampling. (e) Same as (b) but with 110 samples per pixel. (f) Cosine-lobe, environment map, and our portal sampling combined via MIS.

(a) Traditional path tracing 128spp (b) Our portal sampling 128spp

Fig. 12. Replication of Aether's example scene [Anderson et al. 2017]. The Sponza Palace atrium [McGuire 2017] is lit by a point light source and projected into a box through a pinhole. The area of the pinhole is 0.01 percent of the face of the pinhole camera. This scene is notoriously difficult for traditional Monte Carlo methods (a). Our method allows rendering this scene in the path tracing framework (b). The images (a) and (b) are rendered using 128 samples per pixel.



(a) Cosine-lobe 32spp 1.84s (c) Cosine-lobe 99spp 5.68s (b) Cosine-lobe + Portal 32spp 5.65s

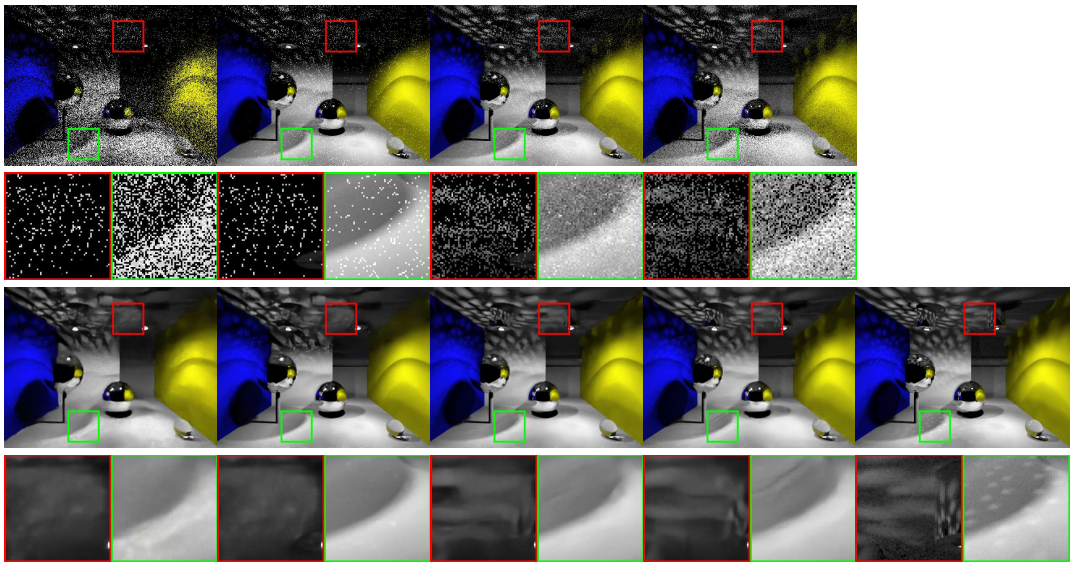
Fig. 13. Lantern with diffuse transmittance material. From left to right: (a) cosine-lobe sampling with 32 samples per pixel, (b) cosine-lobe sampling with 99 samples per pixel, and (c) our portal sampling method applied to the lantern.

the developed sampling technique enables generating portal segments that pass through the small opening.

Our technique allows portal objects to have arbitrary shaders. In the example shown in Fig. 13, a diffuse transmittance material is assigned to the lantern mesh. We placed a point light source inside. More rays are sent to the bright regions of the lantern, and we obtained less noisy results compared to cosine-lobe sampling alone.

Fig. 14 shows a situation where tweaking the guide weights may be useful. In this example, the caustics from the tiny light emitters were rendered by path tracing. This kind of scene is difficult to render by path tracing alone because each emitter is encapsulated in a metallic housing with a glass lens. When only defining the lamp housing lenses as portals, the light paths contributing to the caustics on the ceiling are not well captured (Fig. 14 (b)). Thus, the caustic pattern is not well recovered after de-noising. Defining also the metallic balls as portals and increasing their guiding weights helps to recover better caustic patterns (Fig. 14 (c) and (d)).

The proposed algorithm can also be used to generate paths starting from a light source. This is helpful to render caustics, for example, generated by metallic flakes or carved letters on a rusted iron plate as shown in Fig. 15. When casting photons, we removed the term $\max\{0, \cos\theta_i'\}$ from



(a) Cosine-lobe	(b) Cosine-lobe + Portal	(c) Cosine-lobe + Portal	(d) Cosine-lobe + Portal	(e) Reference
1024spp	1024spp	1024spp	1024spp	-
-	1 : 0	10 : 1	1 : 1	-

Fig. 14. WinOSi scene (http://www.winosi.onlinehome.de). From left to right: (a) cosine-lobe sampling, (b, c, d) our portal sampling and cosine-lobe sampling combined via MIS, and (e) reference. The images in the second row are de-noised with Intel® Open Image Denoise. The bottom row shows the ratio between the guiding weight assigned to the lenses of the ceiling lights and that assigned to the metallic balls. By increasing the guiding weights of the metallic balls, better caustic patterns are recovered on the ceiling. The images, except the reference, are rendered with 1024 samples per pixel.

Eq. 1 and $\frac{1+\cos\theta_s}{2}$ from Eq. 3. Note that the stratification of generated samples are degraded due to sample warping.

Our sampling and PDF evaluations have non-negligible costs. However, users can benefit from our sampling when complex shader networks need to be evaluated. The main bottleneck is the construction of the cumulative distribution table during sampling. Especially when the environment map is taken into account, we fetch multiple values from the MIP maps created for the environment map, and this process cannot be easily vectorized.

6 LIMITATIONS AND FUTURE WORK

Portal Creation and Guiding Weights. When there is an object that can be used as a portal in a scene, the setup cost is almost negligible. We can even automate the setup if the names of shaders have a prefix such as *'window'*. However, users must manually locate a portal object to render, for example, a scene where light passes through a gap in the door. If there are only window objects with transparency maps, the manual adjustment of guiding weights is not necessary. However, when using lampshades as portals, setting guiding weights proportional to emission intensity results in higher-quality images. We did not automate this process.

Tiling and Procedural Textures. Throughout the paper, we assumed that $(u, v) \in [0, 1]^2$, and we did not consider tiling or procedural textures. Tiling would be relatively easy to support; however, it may increase memory consumption. Supporting patterns defined by implicit functions or resolution

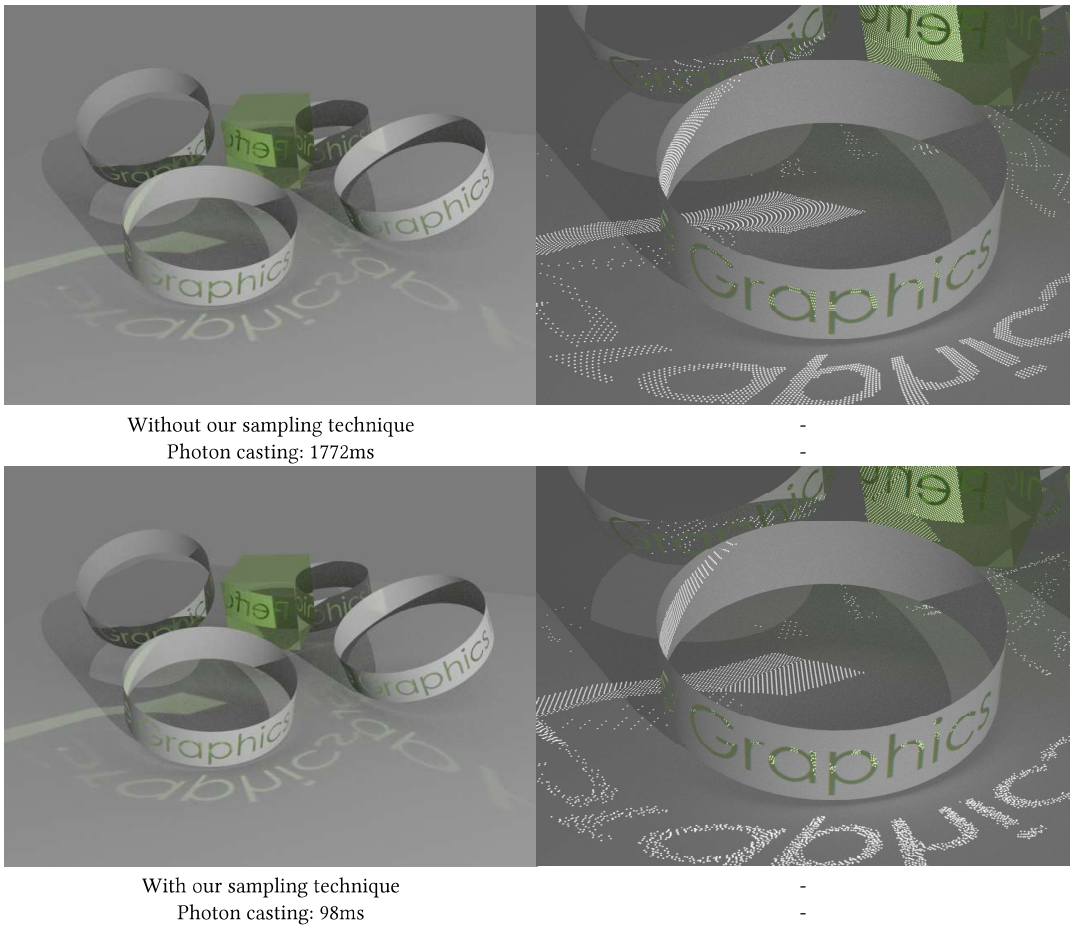
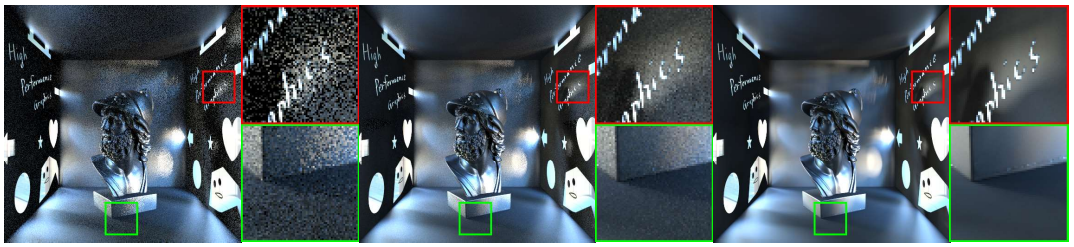


Fig. 15. Caustics generated by carved letters. The scene is lit by a point light. The images on the left are rendered using photon mapping with 100k photons. Our sampling method can generate photons an order of magnitude faster (bottom). The images on the right show how photons are distributed. Samples generated with our method are less stratified (bottom).

free vector graphics is an interesting future avenue to consider. Some noise functions are filterable, and it would be interesting to make use of such properties during hierarchical importance sampling.

Nested Light Portals. Our method works when the portals are nested. However, since each sample is drawn stochastically based on its solid angle, transparency, and the intensity of the environment map, its performance can degenerate. For instance, if a wall with a small portal is entirely covered by another wall that has a large portal, many samples would be generated on the large portal object, and be wasted. Although we believe this scene setup is rare in practice, we would like to develop a better algorithm that can handle such a situation.

Product Importance Sampling. Although our method can be easily combined with other sampling strategies via multiple importance sampling, it will be more efficient to take into account other factors such as BRDFs. Fig. 16 shows an experiment to render glossy surfaces. Since computing the averaged BRDF over the solid angle Ω_c of a cluster's bounding sphere is expensive [Liu et al. 2019], we evaluated a BRDF using the vector $\boldsymbol{C} - \boldsymbol{x_s}$ and multiplied it to the illumination contribution



(a) BRDF(GGX) + Sky	(b) BRDF(GGX) + Sky + Portal	(c) Reference
125spp	64spp	-
13.74s	13.71s	-

Fig. 16. Our sampling applied to glossy surfaces (GGX [Walter et al. 2007] with $\alpha = 0.2$). Only the direct illumination is calculated. From left to right: (a) GGX and environment map sampling combined via MIS, (b) GGX, environment map, and our portal sampling combined via MIS, and (c) reference.

(Fig. 16, (b)). This naive approach worked unexpectedly well for this simple scene; however, we would like to find a more accurate and effective way to approximate the averaged BRDF.

Multiple Importance Sampling. Our new sampling technique should be combined with other samplers to handle various types of scenes robustly. We observed that our textured mesh light sampling combined with cosine-lobe sampling gives slightly worse results than our sampling alone (Fig. 7 (b) and (c), (e) and (f)). It would be necessary to modify MIS weights or PDFs by recently proposed techniques such as MIS compensation [Karlík et al. 2019], optimal multiple importance sampling [Kondapaneni et al. 2019], and variance-aware multiple importance sampling [Grittmann et al. 2019] not to ruin our sampling budget by defensive sampling.

Path Guiding. Path guiding [Vorba et al. 2019] is an adaptive variance reduction technique that learns an approximate representation of the spatiodirectional radiance field of the scene. Our method does not involve any learning process. However, our method could be useful for accelerating learning by generating good initial paths.

BVH Optimization. Agglomerative clustering might improve tree quality, as mentioned by Estevez et al. [Estevez and Kulla 2018]. Recently, re-braiding has been used to support dynamic scenes [Moreau et al. 2019]. We did not use a cut at all in the paper because we used wide BVHs. It could be interesting to combine adaptive cuts [Pantaleoni 2019] with our method. Other optimization techniques designed to accelerate ray tracing apply to improving hierarchical importance sampling. In particular, preferentially contracting nodes that cause significant errors may improve quality. The amount of error could be estimated after casting representative rays as done in the ray specialized contraction method by Gu et al. [2015]. In addition, contracting frequently visited nodes may reduce the costs of sampling and PDF evaluation. There is also a variety of BVH optimization techniques for static and dynamic scenes, including rotation and restructuring. Applying them could improve SAOH-optimized BVHs.

7 CONCLUSION

We introduced the concept of generalized light portals and presented an efficient sampling technique. Our portal sampling, which is based on the computation of illumination from textured mesh lights, can capture light passing through small apertures or transmissive objects in the path tracing framework. The use of generalized light portals is as simple as ticking a binary flag attached to a mesh or shader.

ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their insightful comments and helpful suggestions.

REFERENCES

Attila T. Áfra. 2013. *Faster Incoherent Ray Traversal Using 8-Wide AVX Instructions.* Technical Report. Babeş-Bolyai University, Cluj-Napoca, Romania. http://www.cs.ubbcluj.ro/~afra/publications/afra2013tr_mbvh8.pdf

Luke Anderson, Tzu-Mao Li, Jaakko Lehtinen, and Frédo Durand. 2017. Aether: An Embedded Domain Specific Sampling Language for Monte Carlo Rendering. *ACM Trans. Graph.* 36, 4, Article 99 (July 2017), 16 pages. https://doi.org/10.1145/3072959.3073704

Asen Atanasov, Vladimir Koylazov, Blagovest Taskov, Alexander Soklev, Vassillen Chizhov, and Jaroslav Křivánek. 2018. Adaptive Environment Sampling on CPU and GPU. In *ACM SIGGRAPH 2018 Talks* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. Association for Computing Machinery, New York, NY, USA, Article 68, 2 pages. https://doi.org/10.1145/3214745.3214808

Autodesk, Inc. 2018 (accessed March 5, 2020). *light_portal.* Autodesk, Inc. https://docs.arnoldrenderer.com/display/A5NodeRef/light_portal

Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer, Wojciech Jarosz, and Derek Nowrouzezahrai. 2018. Integrating Clipped Spherical Harmonics Expansions. *ACM Trans. Graph.* 37, 2, Article 19 (March 2018), 12 pages. https://doi.org/10.1145/3015459

Benedikt Bitterli, Jan Novák, and Wojciech Jarosz. 2015. Portal-Masked Environment Map Sampling. *Comput. Graph. Forum* 34, 4 (July 2015), 13–19. http://dl.acm.org/citation.cfm?id=2858834.2858837

Petrik Clarberg. 2008. Fast Equal-Area Mapping of the (Hemi)Sphere using SIMD. *Journal of Graphics Tools* 13, 3 (2008), 53–68.

Holger Dammertz, Johannes Hanika, and Alexander Keller. 2008. Shallow Bounding Volume Hierarchies for Fast SIMD Ray Tracing of Incoherent Rays. In *Proceedings of the Nineteenth Eurographics Conference on Rendering* (Sarajevo, Bosnia and Herzegovina) *(EGSR '08)*. Eurographics Association, Goslar, DEU, 1225–1233. https://doi.org/10.1111/j.1467-8659.2008.01261.x

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 25 (Aug. 2018), 17 pages. https://doi.org/10.1145/3233305

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Trans. Graph.* 37, 3, Article 31 (Aug. 2018), 18 pages. https://doi.org/10.1145/3182161

Valentin Fuetterling, Carsten Lojewski, Franz-Josef Pfreundt, Bernd Hamann, and Achim Ebert. 2017. Accelerated Single Ray Tracing for Wide Vector Units. In *Proceedings of High Performance Graphics* (Los Angeles, California) *(HPG '17)*. Association for Computing Machinery, New York, NY, USA, Article 6, 9 pages. https://doi.org/10.1145/3105762.3105785

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-Noise Dithered Sampling. In *ACM SIGGRAPH 2016 Talks* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 35, 1 pages. https://doi.org/10.1145/2897839.2927430

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-Aware Multiple Importance Sampling. *ACM Trans. Graph.* 38, 6, Article 152 (Nov. 2019), 9 pages. https://doi.org/10.1145/3355089.3356515

Yan Gu, Yong He, and Guy E. Blelloch. 2015. Ray Specialized Contraction on Bounding Volume Hierarchies. *Comput. Graph. Forum* 34, 7 (Oct. 2015), 309–318. https://doi.org/10.1111/cgf.12769

Eric Haines and Tomas Akenine-Möller (Eds.). 2019. *Importance Sampling of Many Lights on the GPU.* Apress, Berkeley, CA, 255–283. https://doi.org/10.1007/978-1-4842-4427-2_18

Eric Heitz. 2017. *Geometric Derivation of the Irradiance of Polygonal Lights.* Research Report. Unity Technologies. https://hal.archives-ouvertes.fr/hal-01458129

Eric Heitz. 2019. A Low-Distortion Map Between Triangle and Square. (June 2019). https://hal.archives-ouvertes.fr/hal-02073696 working paper or preprint.

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-Time Polygonal-Light Shading with Linearly Transformed Cosines. *ACM Trans. Graph.* 35, 4, Article 41 (July 2016), 8 pages. https://doi.org/10.1145/2897824.2925895

Eric Heitz, Stephen Hill, and Morgan McGuire. 2018. Combining Analytic Direct Illumination and Stochastic Shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Montreal, Quebec, Canada) *(I3D '18)*. Association for Computing Machinery, New York, NY, USA, Article 2, 11 pages. https://doi.org/10.1145/3190834.3190852

Ondřej Karlík, Martin Šik, Petr Vévoda, Tomáš Skřivan, and Jaroslav Křivánek. 2019. MIS Compensation: Optimizing Sampling Techniques in Multiple Importance Sampling. *ACM Trans. Graph.* 38, 6, Article 151 (Nov. 2019), 12 pages. https://doi.org/10.1145/3355089.3356565

Alexander Keller, Carsten Wächter, Matthias Raab, Daniel Seibert, Dietger van Antwerpen, Johann Korndörfer, and Lutz Kettner. 2017. The Iray Light Transport Simulation and Rendering System. *CoRR* abs/1705.01263 (2017). arXiv:1705.01263 http://arxiv.org/abs/1705.01263

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *ACM Trans. Graph.* 38, 4, Article 37 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323009

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of I3D 2020)* 3, 1 (2020), 18. https://doi.org/10.1145/3384543

Yifan Liu, Kun Xu, and Ling-Qi Yan. 2019. Adaptive BRDF-Oriented Multiple Importance Sampling of Many Lights. *Comput. Graph. Forum* 38, 4 (2019), 123–133. https://doi.org/10.1111/cgf.13776

Morgan McGuire. 2017. Computer Graphics Archive. https://casual-effects.com/data

Pierre Moreau, Matt Pharr, and Petrik Clarberg. 2019. Dynamic Many-Light Sampling for Real-Time Ray Tracing. In *High-Performance Graphics 2019 - Short Papers, Strasbourg, France, July 8-10, 2019*, Markus Steinberger and Tim Foley (Eds.). Eurographics Association, 21–26. https://doi.org/10.2312/hpg.20191191

Phi Hung Le Nguyen. 2014. Light Portals : Light Transport Variance Reduction. https://escholarship.org/uc/item/2z2051bb

Hiroki Okuno and Kei Iwasaki. 2019. Binary Space Partitioning Visibility Tree for Polygonal Light Rendering. In *SIGGRAPH Asia 2019 Technical Briefs* (Brisbane, QLD, Australia) *(SA '19)*. Association for Computing Machinery, New York, NY, USA, 79–82. https://doi.org/10.1145/3355088.3365153

Jacopo Pantaleoni. 2019. Importance Sampling of Many Lights with Reinforcement Lightcuts Learning. arXiv:1911.10217 [cs.GR]

Christoph Peters. 2016 (accessed March 25, 2020). Free blue noise textures. (2016 (accessed March 25, 2020)). http://momentsingraphics.de/BlueNoise.html

PIXAR ANIMATION STUDIOS 2019 (accessed March 5, 2020). *PxrPortalLight*. PIXAR ANIMATION STUDIOS. https://rmanwiki.pixar.com/display/REN/PxrPortalLight

Martin Sik and Jaroslav Krivanek. 2013. Fast Random Sampling of Triangular Meshes. In *Pacific Graphics Short Papers*, Bruno Levy, Xin Tong, and KangKang Yin (Eds.). The Eurographics Association. https://doi.org/10.2312/PE.PG.PG2013short.017-022

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J. AAI9837162.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses* (Los Angeles, California) *(SIGGRAPH '19)*. Association for Computing Machinery, New York, NY, USA, Article 18, 77 pages. https://doi.org/10.1145/3305366.3328091

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Grenoble, France) *(EGSR'07)*. Eurographics Association, Goslar, DEU, 195–206.

Jingwen Wang and Ravi Ramamoorthi. 2018. Analytic Spherical Harmonic Coefficients for Polygonal Area Lights. *ACM Trans. Graph.* 37, 4, Article 54 (July 2018), 11 pages. https://doi.org/10.1145/3197517.3201291

Jingwen Wang and Ravi Ramamoorthi. 2020. Analytic Spherical Harmonic Gradients for Real-Time Rendering With Many Polygonal Area Lights. *ACM Trans. Graph.* 39, 4 (July 2020), 14.

Cem Yuksel. 2019a. Stochastic Lightcuts. In *High-Performance Graphics - Short Papers*, Markus Steinberger and Tim Foley (Eds.). The Eurographics Association. https://doi.org/10.2312/hpg.20191192

Cem Yuksel. 2019b. *Stochastic Lightcuts for Sampling Many Lights*. Technical Report UUCS 19-006. School of Computing, University of Utah.

Algorithm 1: Select a sample point S for a given shading point $\boldsymbol{x_s}$ (TLAS).

Input: level Lv and direction $\boldsymbol{\omega_i}$
Result: pixel value I
Function `EvaluateMipmap`$(Lv, \boldsymbol{\omega_i})$:
  $u, v \leftarrow$ the pixel coordinate corresponding to $\boldsymbol{\omega_i}$ // Clarberg [2008] (see also supplemental material)
  $Lv \leftarrow \min(\textit{the maximum MIP map level}, Lv)$ // clamp
  I $\leftarrow$ the value at (u, v) of the level-Lv MIP map
  return (I)

Input: shading point $\boldsymbol{x_s}$
Result: sample point S and its probability p
Function `SampleTLAS`$(\boldsymbol{x_s})$:
  $N \leftarrow 16$ // arity
  $p \leftarrow 1$ // probability
  $n \leftarrow$ root node of the TLAS // node of TLAS
  $\xi \leftarrow$ a random value in $[0, 1)$
  while n *is not leaf* **do**
    // build CDF
    $W_{sum} \leftarrow 0$
    Let PDF$[0..N-1]$ and CDF$[0..N-1]$ be new arrays // to store PDF and CDF (both are unnormalized)
    for $i \leftarrow 0$ **to** $N-1$ **do**
      PDF$[i] \leftarrow$ illumination contribution // Eq.1
      if $0 <$ PDF$[i]$ **then**
        if *the environment map exists* **then**
          $\Omega_c \leftarrow$ the solid angle of the i-th child node // Fig.2
          $C \leftarrow$ the center of the i-th child node // Fig.2
          $\boldsymbol{\omega_i} \leftarrow \frac{C - \boldsymbol{x_s}}{\|C - \boldsymbol{x_s}\|}$
          $Lv \leftarrow$ the MIP map level selected by the solid angle Ω_c // Listing. 2
          I $\leftarrow$ `EvaluateMipmap`$(Lv, \boldsymbol{\omega_i})$
          PDF$[i] \leftarrow$ PDF$[i] \times |$I$|$
        $W_{sum} \leftarrow W_{sum} +$ PDF$[i]$
      CDF$[i] \leftarrow W_{sum}$
    // dead branch
    if $0 = W_{sum}$ **then**
      return $(null, p)$
    // select child (Listing 1)
    $j \leftarrow N - 1$
    for $i \leftarrow 0$ **to** $N-1$ **do**
      if $C[i] > \xi \times W_{sum}$ **then**
        $j \leftarrow i$
        break
    // update the probability
    $p \leftarrow p \times$ PDF$[j]/W_{sum}$
    // rescale the random value
    if $0 < j$ **then**
      $\xi \leftarrow (\xi \times W_{sum} -$ CDF$[j-1])/$PDF$[j]$
    else
      $\xi \leftarrow \xi \times W_{sum}/PDF[0]$
    // next node to traverse
    $n \leftarrow$ the j-th child node of n
  $t \leftarrow$ the triangle in the leaf node n
  $S, p_{sub} \leftarrow$ `SampleBLAS` $(t, \boldsymbol{x_s})$
  return $(S, p \times p_{sub})$

Algorithm 2: Select a sample point S for a given shading point $\boldsymbol{x_s}$($BLAS$).

Input: triangle t and shading point $\boldsymbol{x_s}$
Result: sample point S and its probability p

Function SampleBLAS($t, \boldsymbol{x_s}$):
  $N \leftarrow 16$ // arity
  $p \leftarrow 1$ // probability
  $n \leftarrow$ root node of the BLAS corresponding to t
  $\xi \leftarrow$ a random value in $[0, 1)$
  $Lv \leftarrow$ MIP map level selected by the solid angle of the triangle t // Listing. 2
  loop
    // increment MIP map level: +2 for arity $N = 16$, +1 for arity $N = 4$
    $Lv \leftarrow Lv + 2$
    // build CDF
    $W_{sum} \leftarrow 0$
    Let PDF$[0..N-1]$ and CDF$[0..N-1]$ be new arrays // to store PDF and CDF (both are unnormalized)
    Let SignBits$[0..N-1]$ be a new array to store sign bits
    for $i \leftarrow 0$ **to** $N-1$ **do**
      PDF$[i] \leftarrow$ illumination contribution // Eq.3
      if $0 <$ PDF$[i]$ **then**
        if *the environment map exists* **then**
          $C \leftarrow$ the center of the i-th subtriangle
          $\boldsymbol{\omega_i} \leftarrow \frac{C - \boldsymbol{x_s}}{\|C - \boldsymbol{x_s}\|}$
          I $\leftarrow$ EvaluateMipmap($Lv, \boldsymbol{\omega_i}$)
          SignBits$[i] \leftarrow$ the sign bit of I
          PDF$[i] \leftarrow$ PDF$[i] \times |$I$|$
        $W_{sum} \leftarrow W_{sum} +$ PDF$[i]$
      CDF$[i] \leftarrow W_{sum}$
    // dead branch
    if $0 = W_{sum}$ **then**
      return $(null, p)$
    // select child (Listing 1)
    $j \leftarrow N-1$
    for $i \leftarrow 0$ **to** $N-1$ **do**
      if $C[i] > \xi \times W_{sum}$ **then**
        $j \leftarrow i$
        break
    // update the probability
    $p \leftarrow p \times$ PDF$[j] / W_{sum}$
    // rescale the random value
    if $0 < j$ **then**
      $\xi \leftarrow (\xi \times W_{sum} -$ CDF$[j-1])/$PDF$[j]$
    else
      $\xi \leftarrow \xi \times W_{sum}/PDF[0]$
    // next node and subtriangle Fig.8 (see also supplemental material)
    if n *is not leaf* **then**
      $n \leftarrow$ the j-th child node of n
    $t \leftarrow$ the j-th subtriangle of t
    // exit condition
    if n *is leaf* **then**
      if *the environment map exists* **then**
        if *the maximum MIP map level* $\leq Lv$ **then**
          break
        if SignBits$[j]$ *is true* **then**
          break
      else
        break
  // generate a sample (Heitz [2019])
  $S \leftarrow$ randomly sampled point on the subtriangle t
  return (S, p)