High Performance Graphics (2014)
Jonathan Ragan-Kelley and Ingo Wald (Editors)

Layered Reflective Shadow Maps for Voxel-based Indirect Illumination

Masamichi Sugihara, Randall Rauwendaal, and Marco Salvi

Intel Corporation

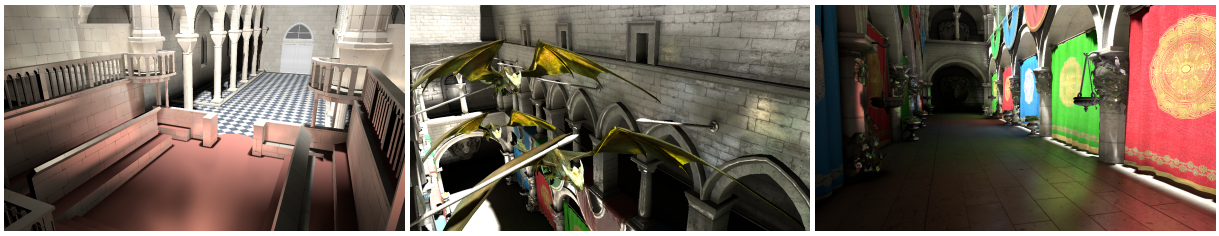


Figure 1: Voxel-based interactive indirect illumination supporting both dynamic lights and scenes with significantly reduced and bounded memory consumption.

Abstract

We introduce a novel voxel-based algorithm that interactively simulates both diffuse and glossy single-bounce indirect illumination. Our algorithm generates high quality images similar to the reference solution while using only a fraction of the memory of previous methods. The key idea in our work is to decouple occlusion data, stored in voxels, from lighting and geometric data, encoded in a new per-light data structure called layered reflective shadow maps (LRSMs). We use voxel cone tracing for visibility determination and integrate outgoing radiance by performing lookups in a pre-filtered LRSM. Finally we demonstrate that our simple data structures are easy to implement and can be rebuilt every frame to support both dynamic lights and scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Global illumination (GI) greatly increases visual realism by simulating light transport between each surface in the scene; as such, it is a highly desirable effect to enable in real-time applications. This prospect is particularly challenging due to the tremendous amount of computations and bandwidth required, which is why the use of GI is, to this date, mostly confined to off-line rendering. Recently, several interactive GI techniques have been developed, thanks to the ever increasing computational capabilities of modern GPUs, but with several limitations such as pre-computation requirements, poor performance, and image artifacts.

Among several interactive GI techniques, voxel cone tracing (VCT) [CNS*11] simulates at interactive rates both diffuse and glossy indirect illumination, and it has been evaluated for use in game engines [Mit12]. VCT does not exhibit many of the problems of other real-time GI algorithms, such as bright spots or temporal flickering, and rendering time is less dependent on scene complexity because cones interact with a regular and filterable data structure. Despite these benefits, voxel-based methods can be very memory intensive; each voxel must encode a large number of the attributes necessary to compute indirect lighting, including directionally dependent terms that require even more memory when pre-filtered (i.e. mipmapping). Sparse voxel data structures need to be constructed to reduce memory consumption, which makes an implementation more complex. Moreover the voxelization (i.e. the process of converting triangles into voxels) and lighting data encoding both require expensive atomic operations to avoid data races.

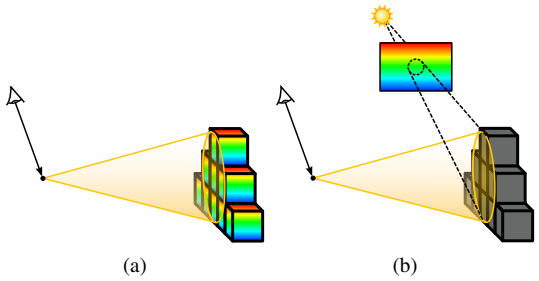


Figure 2: (a) Traditional voxel cone tracing relies on tracing cones into "fat" voxel volumes with many attributes per voxel. (b) In contrast, our "slim" voxels encode only binary visibility information and store the remaining attribute data in our pre-filtered LRSM data structure.

We present a new method for interactive indirect illumination using voxel cone tracing with layered reflective shadow maps (LRSMs). Figure 2 summarizes the difference between the traditional voxel cone tracing approach and our work. Our method uses voxels only for visibility determination, requiring to store only one byte per voxel. We also introduce a new pre-filterable data structure, layered reflective shadow maps (LRSMs), which supports radiance integration. Our contributions can be summarized as follows:

- A voxel cone tracing method to interactively simulate both diffuse and glossy indirect illumination with significantly reduced and bounded memory consumption.
- Simple data structures: occlusion voxels and layered reflective shadow maps, neither of which require sparse memory allocations or atomic operations.
- Efficient support for fully dynamic geometry and lighting.

2. Related Work

A great deal of effort has been expended on finding GI solutions; we restrict our focus to those methods that target interactive rendering of indirect illumination effects. For comprehensive surveys of the state of the art in this field see [RDGK12, DKH*14], and for non-interactive methods see [DBB06].

Virtual Point Lights Keller [Kel97] introduced the concept of *virtual point lights* (VPLs) to represent the indirect illumination in the scene. This is an effective approach, but can suffer from bright spots based on its sampling strategy. *Reflective shadow maps* (RSMs) [DS05] consider each pixel of the shadow map (extended with additional information) as an indirect light source, but neglect occlusion information. Dachsbacher and Stamminger [DS06] later converted the process of gathering illumination from the RSMs to a scatter operation with a bounded splatting approach using the rasterizer. Laine et al. [LSK*07] proposed an extension to VPLs that enabled the reuse of VPL shadow maps across several frames. Ritschel et al. [RGK*08] introduced *imperfect shadow maps* to rapidly create hundreds of VPL shadow maps via coarse point-rendering. However, due to holes in the coarse shadow maps, ISMs are not able to accurately evaluate near-field illumination. Several approaches cluster VPLs [DGR*09, PKD12] to create area light sources, or attempt to improve the VPL distribution [REH*11], with the aim of accelerating and improving the indirect illumination results. Tokuyoshi and Ogaki [TO12] use a real-time bidirectional sampling strategy to avoid VPL artifacts without resorting to clamping but it supports only rough glossy indirect illumination due to a limited number of samples.

Voxel-based Global Illumination Recently a series of approaches have relied increasingly on voxels in order to better utilize the available graphics hardware and achieve a degree of scene independence when it comes to computing secondary illumination effects. Kaplanyan et al. [KD10] stored a discretized distribution of light, initialized from RSMs, using low order spherical harmonics (SH) and developed an efficient light diffusion scheme. Thiedemann et al. [THGM11] used voxels to accelerate ray-intersections and generated RSMs to compute radiance for real-time near-field indirect illumination. Crassin et al. [CNS*11] traced not rays, but volumetric cones through a hierarchical voxel structure, and stored an anisotropic radiance function in a sparse voxel octree (SVO). Mittring [Mit12] demonstrated the feasibility of such approaches to video games. Rauwendaal [Rau13] used a similar approach but stored SH coefficients in the voxels. The benefit of voxel-based approaches is that given a sufficiently fast voxelization, the computation of indirect lighting becomes largely independent of scene complexity. The downside of the voxel-based approach is that as voxels become increasingly “fat”, that is, as the number of attributes increases linearly, the volume wastes cubically more space. Sparse voxel structures such as SVO in Crassin et al. [CNS*11], and hardware support for sparse texture resources alleviate this requirement somewhat, but still fail to put a tight bound on memory requirements.

In contrast, our approach stores only binary “slim” voxel occlusion data, and performs a lookup into our filtered (memory bounded) LRSM data structure to recover attribute information. This approach is similar to [THGM11] but is adapted to voxel cone tracing which can support a wider range of diffuse and glossy indirect illumination effects. Binary voxels are compact enough that they do not necessitate the construction of sparse data structures, and they can be built without the use of atomic operations.

3. Overview

Our work is inspired by voxel cone tracing, which we review briefly in Section 3.1, followed by an overview of our algorithm (Figure 3) in Section 3.2.

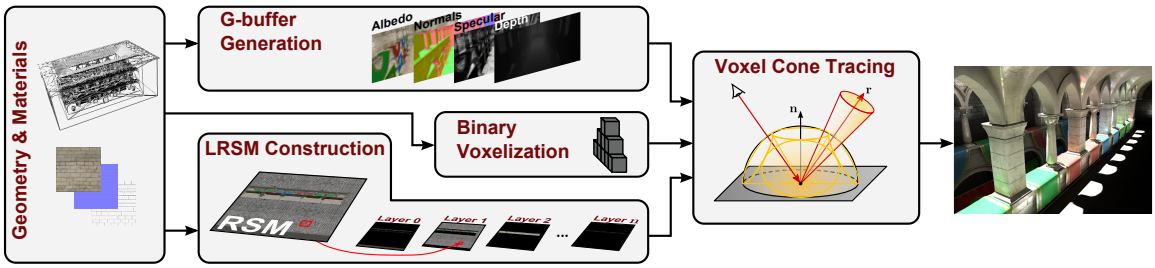


Figure 3: Pipeline. Traditional approaches are used to initialize both G-buffers and RSMs. The RSMs are split into the quarter resolution layers to avoid pre-filtering depth and normal discontinuities. Finally, a set of cones is traced for each pixel of the G-buffer into the binary voxels to determine visibility and look up the sample attributes within the pre-filtered LRSM structure to compute the indirect illumination contribution for the final image.

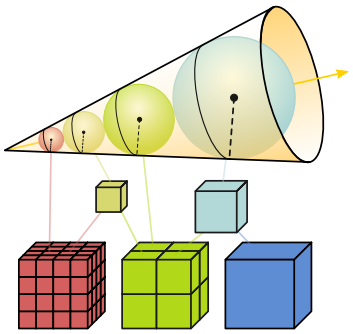


Figure 4: The diameter of each sample along the cone selects the quadrilinearly interpolated voxel sample point within the pre-filtered voxel mipmap levels. The sample size progressively increases as it steps further.

3.1. Voxel Cone Tracing for Indirect Illumination

Voxel cone tracing (VCT) estimates the incoming light intensity I by tracing over pre-filtered voxel-based representation of the scene. It approximates tracing a bundle of rays with one cone by exploiting their spatial coherence. A cone volume is approximated by a sequence of cone elements (we use sphere samples as shown in Figure 4), over which occlusion and outgoing radiance are integrated. The intensity I is then accumulated in front-to-back order:

$$I = \alpha_0 L_0 + \sum_{i=1} \alpha_i L_i \prod_{j=0}^{i-1} (1 - \alpha_j) \qquad (1)$$

where α_i and L_i are the integrated occlusion and outgoing radiance for the i^{th} sample. These terms can be instantly computed using the pre-filtered data stored in the voxel mipmaps chain. The diameter of sphere sample is used to select the voxel mip level: $mipLevel = \log_2(d_{sphere})$ where d_{sphere} is the sphere diameter in voxel units. The tracing process stops when a cone is considered fully occluded or when tracing steps reach a certain threshold. A full description of this method can be found in [HN12].

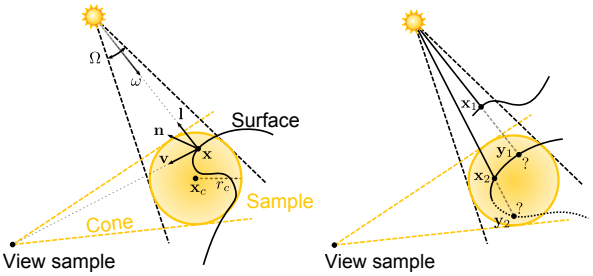
Crassin et al. [CNS*11] utilizes this mechanism to estimate indirect illumination by encoding both geometric and lighting data in voxels. Final gathering is approximated by shooting several cones from each image sample visible from the eye. Wide cones covering the hemisphere are used to capture the indirect diffuse component, while a single cone directed along the mirror reflection vector, whose aperture is controlled by glossiness, samples the indirect specular component. These cones can be adjusted according to the material BRDF.

3.2. Our Algorithm

Our method operates as shown in Figure 3. We first convert the scene into binary voxels (i.e. empty or full voxels) using a GPU based voxelization method [CG12]. This data is used to determine occlusion. This coarse scene representation is pre-filtered by generating mipmaps. Second, an LRSM is generated for each light by rendering the scene from the light's point-of-view into a reflective shadow map (RSM) [DS05], which is split into several pre-filtered layers. Splitting the RSM into layers is a key step to avoid large depth and normal discontinuities, which significantly increases the quality of our pre-filterable depth and normal representations.

Our software rendering pipeline utilizes deferred shading to avoid unnecessary calculations. We first render a G-buffer from the eye and then compute direct and indirect lighting at each visible sample. Indirect illumination is estimated with Equation 1 by performing cone-based final gathering over voxels and LRSMs. For each cone sample, we first evaluate occlusion α by sampling the voxelized scene, and then sample the LRSM to gather the associated outgoing radiance L. We determine which LRSM region to sample by projecting the sphere associated with each cone sample onto the LRSM, as shown in Figure 2b.

We essentially follow the strategy of Section 3.1 except computing the integrated outgoing radiance L from LRSMs, which allows us to avoid encoding geometric and lighting



(a) (b)

Figure 5: (a) Radiance integration is performed over the solid angle Ω from the light source. (b) However, the regions behind an occluding surface y_1 (i.e. in shadow) and the backface regions y_2 are omitted from the integration.

data within the voxels. The entire process (Figure 3) can be performed every frame, enabling fully dynamic scenes and lights.

4. Layered Reflective Shadow Maps

In this section, we first describe how to integrate outgoing radiance using a reflective shadow map (Section 4.1). Subsequently, we show pre-filtering (Section 4.2) and partitioning (Section 4.3) strategies to accelerate radiance integration. Note that we describe how our method works with a spot light, but we also support directional lights as in [DS05].

4.1. Radiance Integration

The outgoing radiance L associated with a cone sample can be computed as the integral over a surface S contained inside the sample volume [BN12,HN12]. This is the local illumination equation which is a function of incoming radiance and surface attributes toward the observer (view sample in our case). We define L as a function of a single light source and a filter w:

$$L = \frac{\int_S w'(\mathbf{x}) f_r(\mathbf{l},\mathbf{v}) L_d \langle \mathbf{n}\cdot\mathbf{l}\rangle V(\mathbf{x},\mathbf{l}) d\mathbf{x}}{\int_S w'(\mathbf{x}) d\mathbf{x}} \qquad (2)$$
$$w'(\mathbf{x}) = w(\mathbf{x}) V(\mathbf{x},\mathbf{v}) \langle \mathbf{n}\cdot\mathbf{v}\rangle$$

where $\mathbf{x}$ is a point on the surface S. f_r is the BRDF, L_d is the incoming radiance directly from the light source with direction $\mathbf{l}$, $\mathbf{n}$ is the surface normal, and $\mathbf{v}$ is the direction to the view sample, all of the terms evaluated at $\mathbf{x}$. $V(\mathbf{x},\mathbf{l})$ and $V(\mathbf{x},\mathbf{v})$ are binary visibility functions from $\mathbf{x}$ to the light and the view sample, while $\langle\cdot\rangle$ represents a dot product clamped to zero.

Since we compute this equation using an RSM where each sample corresponds to a differential solid angle $d\omega$ from a light source, we need to rewrite Equation 2 as the integral over the solid angle Ω covering a cone sample from the light source (Figure 5a). However, the surface not receiving light cannot be integrated with this change because we miss its information (Figure 5b). Therefore, we make two assumptions before defining the integral:

- An RSM can locally cover the entire surface S inside a cone sample; we ignore the backface-culled surface areas from a light source.
- If the ray along a differential solid angle from a light source does not hit the surface S inside a cone sample, we assume it corresponds to the shadowed surface area facing to the light direction (i.e. $\mathbf{n} = \mathbf{l}$).

These assumptions allow us to rewrite Equation 2 as follows:

$$L \approx \frac{\int_\Omega w'(\omega) f_r(\mathbf{l},\mathbf{v}) L_d \langle \mathbf{n}\cdot\mathbf{l}\rangle V(\mathbf{x},\mathbf{x}_c,r_c) d\omega}{\int_\Omega w'(\omega) d\omega} \qquad (3)$$

where $\mathbf{x}$ is the position on the surface hit by a ray from the light direction $\mathbf{l} = -\omega$. We replace $V(\mathbf{x},\mathbf{l})$ with $V(\mathbf{x},\mathbf{x}_c,r_c)$ which returns 1 if $\mathbf{x}$ is inside a cone sample, and 0 otherwise. $\mathbf{x}_c$ and r_c are the center point and the radius of the cone sample, and $V(\mathbf{x},\mathbf{x}_c,r_c)$ is defined as $H(r_c - \|\mathbf{x}-\mathbf{x}_c\|)$ where H is the Heaviside step function. This change also yields the Jacobian $d\mathbf{x} = (r^2/\langle\mathbf{n}\cdot\mathbf{l}\rangle)d\omega$ where r is the distance between $\mathbf{x}$ and the light source, and the weight w' is rewritten as follows:

$$w'(\omega) = \frac{w(\omega) V(\mathbf{x},\mathbf{v}) \langle \mathbf{n}\cdot\mathbf{v}\rangle}{\langle \mathbf{n}\cdot\mathbf{l}\rangle} \qquad (4)$$

where r^2 yielded from the Jacobian can be considered as a constant inside a cone sample, and is canceled by moving it outside the integrals of Equation 3. Note that $\mathbf{n} = \mathbf{l}$ if $V(\mathbf{x},\mathbf{x}_c,r_c) = 0$ from the second assumption.

4.2. Reflective Shadow Map Pre-filtering

Calculating the radiance associated with arbitrarily large regions of an RSM can be computational expensive and does not guarantee the stable and predictable performance required by real-time applications. To solve this problem, we pre-filter the RSM attributes by using the GPU to generate a mipmap chain. A similar method is adopted by translucent shadow maps [DS03] where pre-filtered attributes rendered from light space are stored in a mipmap chain to enable fast integration of subsurface scattering. Although the filter w is limited to a box filter, the attributes can be efficiently integrated using the hardware texture samplers.

However, this requires to reformulate Equation 3 so that it is linear with respect to the RSM attributes. To do so, we make assumptions similar to the ones often followed by other pre-filtering methods [BN12]:

- All of the RSM attributes are uncorrelated; the equation can be decomposed into several simple terms.
- The Jacobian and view-dependent terms are locally constant; they can be moved outside the integrals and canceled.

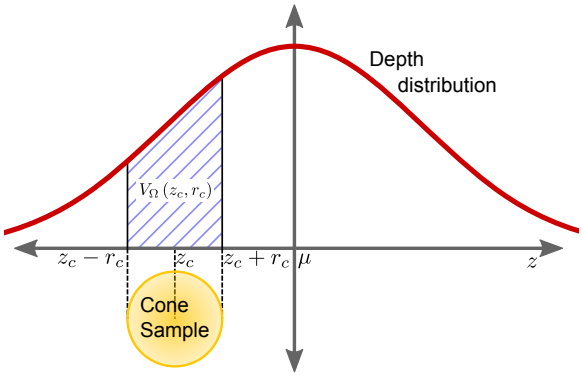


Figure 6: We use a Gaussian distribution to approximate the depth distribution of RSM samples. The shaded area represents the percentage of RSM samples inside the cone sample.

These assumptions allow us to approximate radiance integration as follows:

$$L(\mathbf{x}_c, r_c, \mathbf{v}_c) \approx R_\Omega V_\Omega(\mathbf{x}_c, r_c) N_\Omega(\mathbf{v}_c) \tag{5}$$

$$R_\Omega = \int_\Omega \hat{w}(\omega) f_r(\mathbf{l}, \mathbf{v}) L_d \langle \mathbf{n} \cdot \mathbf{l} \rangle d\omega \tag{6}$$

$$V_\Omega(\mathbf{x}_c, r_c) = \int_\Omega \hat{w}(\omega) H(r_c - \|\mathbf{x} - \mathbf{x}_c\|) d\omega \tag{7}$$

$$N_\Omega(\mathbf{v}_c) = \langle \bar{\mathbf{n}} \cdot \mathbf{v}_c \rangle, \ \bar{\mathbf{n}} = \int_\Omega \hat{w}(\omega) \mathbf{n} d\omega \tag{8}$$

$$\hat{w}(\omega) = \frac{w(\omega)}{\int_\Omega w(\omega) d\omega} \tag{9}$$

where $\mathbf{x}_c$, r_c, and $\mathbf{v}_c$ are the variables dependent on a cone sample: center position, radius, and cone direction from $\mathbf{x}_c$ to the view sample. We restrict the indirect bounce to diffuse only, so $f_r(\mathbf{l}, \mathbf{v})$ is replaced with the diffuse BRDF ρ/π where ρ is the diffuse albedo at $\mathbf{x}$. Also, since we lose the view dependency due to the second assumption, we introduce an additional term $N_\Omega(\mathbf{v}_c)$ to approximate backface culling. With this change, all of the RSM attributes are pre-filterable except $V_\Omega(\mathbf{x}_c, r_c)$ due to non-linearity introduced by the Heaviside step function (see Equation 7).

Depth pre-filtering Essentially, the $V_\Omega(\mathbf{x}_c, r_c)$ term estimates the percentage of the rays from the light, or RSM samples, hitting the surface inside a cone sample. This is a similar problem to shadow map filtering techniques. Variance shadow maps [DL06] estimate the percentage of light reaching a surface by using a pre-filterable depth representation based on computing the first two moments of the depth distribution. Thus, we adopt a similar solution and use the depth distribution of the RSM samples to estimate Equation 7.

We first render the depth z and the square of the depth z^2 into an RSM instead of the position $\mathbf{x}$. The depth mean μ and variance σ^2 over the solid angle Ω can be then computed as follows:

$$\mu = \int_\Omega \hat{w}(\omega) z d\omega \tag{10}$$

$$\sigma^2 = \int_\Omega \hat{w}(\omega) z^2 d\omega - \mu^2 \tag{11}$$

We approximate the depth distribution with the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ because its integral over the depth range of a cone sample estimates the percentage of the RSM samples inside the cone sample (Figure 6), which can be directly used to approximate Equation 7. By replacing $\mathbf{x}_c$ with z_c, the depth of the cone sample center from the light, it can be reformulated as follows:

$$V_\Omega(z_c, r_c) \approx \int_{z_c - r_c}^{z_c + r_c} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} dz \tag{12}$$

This formulation allows us to approximate Equation 7 using the pre-filtered depths and squared depths.

4.3. Reflective Shadow Map Partitioning

If a filtered region contains depth or normal discontinuities, the radiance approximation (Equation 5) is no more valid due to violating the assumptions made in Section 4.2. To avoid pre-filtering such regions, we partition an RSM into several layers and pre-filter each layer separately. Our strategy is to partition an RSM according to depth and normal values. However, since a normal is a 3D vector, we instead use the dot product between the normal and the incoming light direction, $\langle \mathbf{n} \cdot \mathbf{l} \rangle$, as a partitioning criterion to simplify the process. It does not isolate normal discontinuities completely but minimizes the artifacts from normal discontinuities in practice. We place static partitions along depth and normal. If we define a partitions in depth and b partitions in normal, the number of layers n is $(a+1)(b+1)$.

When performing radiance integration at a cone sample, we compute Equation 5 at each layer with a small weight modification. Since the RSM attributes are scattered over the layers, we add an opacity term α to the weight $\hat{w}$ (Equation 9):

$$\hat{w}_l(\omega) = \frac{w(\omega)\alpha_l}{\int_\Omega w(\omega)\alpha_l d\omega} \tag{13}$$

where α_l returns 1 if the RSM attributes at a solid angle ω are valid in the layer l and 0 otherwise. We then compose all of the integration results L_l:

$$L = \sum_{l=1}^{n} A_l L_l, \ A_l = \frac{\int_\Omega w(\omega)\alpha_l d\omega}{\int_\Omega w(\omega) d\omega} \tag{14}$$

Theoretically, we need to compute L_l at every layer. By following the sampling strategy of layered variance shadow maps [LM08], however, we instead sample only the selected layers according to the depth of the cone sample center z_c (i.e. only sampling the layers containing z_c in their depth range). This reduces the amount of layer sampling, making it scalable with the number of the depth partitions. This

Attribute	Value	Format
Reflected radiance	$\rho L_d \langle \mathbf{n} \cdot \mathbf{l} \rangle / \pi$	R16G16B16A16 FLOAT
Normal	$\mathbf{n}$	R16G16B16A16 FLOAT
Depth and Depth^2	z, z^2	R16G16B16A16 UNORM

Table 1: LRSM memory layout. We also store the opacity value α in both the normal and depth textures. This value is used to cancel empty texel contributions from the mipmaps during layer sampling.

also reduces the light bleeding exhibited by many shadow map filtering techniques. To ensure a smooth transition between layers, we add some overlap between layers during depth partitioning, and when an overlap region is sampled, we smoothly decay the sample influence. We define such a decay function $f(z_c)$ in the layer depth range $[zMin_l, zMax_l]$ as follows:

$$f(z_c) = \begin{cases} k(\frac{z_c - zMin_l}{\delta}) & z_c < zMin_l + \delta \\ 1 & zMin_l + \delta \leq z_c \leq zMax_l - \delta \\ k(\frac{zMax_l - z_c}{\delta}) & zMax_l - \delta < z_c \end{cases} \quad (15)$$

where δ is the overlap region and k is a smooth falloff function clamped between 0 and 1. Unless an overlap region is sampled, the number of the layers to sample is $b+1$ which depends only on the number of normal partitions.

The additional memory requirement of having layers is relatively small. Since the purpose of partitioning is to avoid pre-filtering depth and normal discontinuities, we need layers only in mipmap levels, increasing memory usage by only $n/3$, where n is number of layers; see Table 2 for examples of memory consumption in typical configurations.

5. Implementation

We have implemented our algorithm in Direct3D 11. We generate occlusion voxels using a GPU voxelization method [CG12] and write the results into a 3D texture via UAVs. Since occlusion voxels are binary before mipmapping, atomic operations are not required to update the 3D texture. Because of this, the ideal memory allocation is 1 bit per voxel in the base voxel level and 1 byte in the mipmap levels. However, since 3D APIs do not expose 1 bit formats, we allocate 1 byte per voxel.

The LRSM is constructed in two passes and stored in a 2D texture array. The LRSM attributes and memory layout are shown in Table 1. We first render an RSM from the light's point-of-view via rasterization and store reflected radiance, depth, normal, and $\langle \mathbf{n} \cdot \mathbf{l} \rangle$ in 2D textures. These attributes can be packed in two 16-bit per component textures. In the second pass, each RSM texel is evaluated according to depth and $\langle \mathbf{n} \cdot \mathbf{l} \rangle$ (Section 4.3) and assigned to the appropriate layer using a Compute Shader (CS). At the same time, we also warp the depth values between 0 and 1 according to the depth range of the assigned layer as in [LM08] and compute their squared depth values. Since the LRSM is a filtered representation of the RSM, the layers begin at one quarter the resolution of the RSM. Therefore, we launch one CS thread per 2x2 RSM texel region, and texels are merged if they are assigned to the same layer. Once all of the data structures are constructed, we generate mipmaps of occlusion voxels and LRSMs to pre-filter the encoded attributes.

For lighting computation, we perform final gathering by shooting cones for every G-buffer pixel. Occlusion voxels and LRSMs are accessed by cones to fetch pre-filtered attributes and compute Equation 1. To compute the integrated outgoing radiance at a cone sample by evaluating an LRSM, we first project the cone sample onto the RSM space (Figure 2b) to determine the RSM coordinates and mip level. Similar to voxel mip level selection, the diameter of the projected circle region determines the RSM mip level: $mipLevel = \log_2(d_{circle})$ where d_{circle} is the diameter of the projected circle in texel units. We then select the layers to sample according to the cone sample depth z_c as described in Section 4.3 and fetch the pre-filtered RSM attributes at each selected layer. Once the layers are sampled, the integrated outgoing radiance can be evaluated by computing Equation 14.

The pseudo code for sampling an LRSM is shown in Algorithm 1. We compute Equation 12 by looking up the standard normal distribution table. When normal $\mathbf{n}_{mip}$, depth z_{mip}, and squared depth z^2_{mip} are sampled, they need to be normalized by the relative opacity value stored in their textures in order to remove the contribution of empty texels. However, this is not necessary for reflected radiance $\mathbf{r}_{mip}$, as unnormalized $\mathbf{r}_{mip}$ is equivalent to being weighted by A in Equation 14.

Algorithm 1 Sample LRSM

```
L = 0                                              ▷ outgoing radiance
for all layers do
  if z_c in depth range of layer l [zMin_l, zMax_l] then
    Sample normal n_mip
    N = ⟨n_mip · v_c⟩                                  ▷ Eq. 8
    if N > 0 then                                  ▷ backface culling
      Sample depth z_mip and squared depth z²_mip
      Depth mean μ = z_mip                             ▷ Eq. 10
      Depth variance σ² = z²_mip − μ²                  ▷ Eq. 11
      Compute V from N(μ, σ²)                          ▷ Eq. 12
      if V > 0 then
        Sample reflected radiance r_mip (= AR)
        Compute decay function f(z_c)                  ▷ Eq. 15
        L+ = f(z_c)ARVN                                ▷ Eq. 14
      end if
    end if
  end if
end for
```

	SIBENIK		SPONZA		SPONZA DRAGON	
Data Structure	Resolution	Memory	Resolution	Memory	Resolution	Memory
Occlusion Voxels	256^3	19.2 MB	512^3	156.4 MB	512^3	156.4 MB
RSM	1024^2	22.4 MB	1024^2	22.4 MB	1024^2	22.4 MB
LRSM	512^2	33.6 MB	512^2	33.6 MB	512^2	50.4 MB
Number of Layers (D × N)	4 layers (2 × 2)		4 layers (2 × 2)		6 layers (3 × 2)	
Total Memory		75.2 MB		212.4 MB		229.2 MB
Performance	GTX770	Iris Pro	GTX770	Iris Pro	GTX770	Iris Pro
	58.9 ms	258.3 ms	58.5 ms	197.4 ms	63.8 ms	244.2 ms

Table 2: The summary of data structure and performance for each scene. Note that LRSM resolution is always RSM resolution/4, and that memory consumption is independent of scene complexity. D × N represents depth and normal layer numbers.

	VCT GTX770	LRSM GTX770	LRSM Iris Pro
Voxelization		5.3 ms	29.3 ms
LRSM Construction		3.1 ms	18.1 ms
Shadow Cone	9.8 ms	7.0 ms	35.7 ms
Diffuse Cone (6 cones)	4.5 ms	22.1 ms	68.3 ms
Specular Cone (1 cone)	20.5 ms	16.6 ms	43.2 ms

Table 3: Timings of individual stages to render SPONZA. Voxelization time includes voxel clear, voxelization, and voxel mipmap generation. LRSM construction time includes RSM clear, RSM rendering, LRSM construction, and LRSM mipmap generation. Note that the performance difference of shadow cone tracing between VCT and LRSM can be attributed to different data structures: anisotropic voxels (VCT) vs. isotropic voxels (LRSM).

6. Results

We tested our algorithm using three scenes: SIBENIK, SPONZA, and SPONZA DRAGON (Figure 1 and Table 2). A spot light is placed in SIBENIK, while a directional light is used in SPONZA and SPONZA DRAGON. SPONZA DRAGON is a dynamic version of the SPONZA scene containing three textured animated dragons composed of 74k triangles each. All of the scenes shown in this paper were rendered at 1280×720. We reuse the occlusion voxels to render soft shadows by tracing shadow cones [Cra11]. For comparison, we also implemented Crassin et al.'s voxel cone tracing [CNS*11] which we denote VCT. To simulate single-bounce indirect illumination, we pre-computed their data structure using a dense voxel grid: we encoded radiance and occlusion (R16G16B16A16), and normal (R16G16B16A16) in the base voxels and generated anisotropic mips from them by storing filtered radiance and occlusion in each direction.

Memory The main contribution of our algorithm is significantly reduced and bounded memory consumption. The memory used in each scene is shown in Table 2. The most memory intensive scene is SPONZA DRAGON. However, it still uses only 229.2 MB while it requires 3 GB in VCT without sparse voxel octree (SVO) construction. An SVO could reduce memory consumption but its construction is not a trivial process and its memory usage is not tightly bounded. In SPONZA and SPONZA DRAGON, memory usage is mostly dominated by the 512^3 occlusion voxels. With 256^3 occlusion voxels in SIBENIK, it is down to 75.2 MB.

Performance We have measured our implementation on a Geforce GTX 770 discrete graphics card (230 W TDP) and an Intel Iris Pro 5200 integrated GPU (47 W TDP shared with CPU). We achieved interactive rates on all scenes (Table 2) with data structure construction time included. The cost of individual stages for SPONZA are also shown in Table 3. The most time consuming part of our algorithm is tracing the 6 diffuse cones, but interestingly, tracing 1 specular cone takes more time in VCT. This is likely because wider cones are used in diffuse cone tracing and most of sampling happens in lower level mips, while specular cones access higher level mips and sometimes even the base voxels. In our algorithm, tracing 6 diffuse cones is slower than specular tracing due to the computational cost of evaluating Equation 12.

Image Quality We generated reference images using the Embree path tracer [WWB*14] for the image comparison shown in Figure 7. Despite reduced memory consumption and dynamic data structure construction, our results closely resemble the VCT and reference results. In SPONZA, our algorithm captures diffuse color bleeding from the curtains, indirect shadows behind the round curtains, and glossy reflections. In SIBENIK, there is light leaking through thin walls in our algorithm. This does not happen in VCT because anisotropic mips are used for occlusion integration. However, they are sometimes over conservative and miss capturing the color bleeding. We have also tested a dynamic scene in SPONZA DRAGON (Figure 1 and the accompanying video) thanks to our dynamic data structure construction. All of the illumination effects are rendered smoothly without any temporal artifacts.

7. Limitations

We have demonstrated that our algorithm can simulate single-bounce indirect illumination at interactive rates. To enable multiple bounces, however, we would need to store

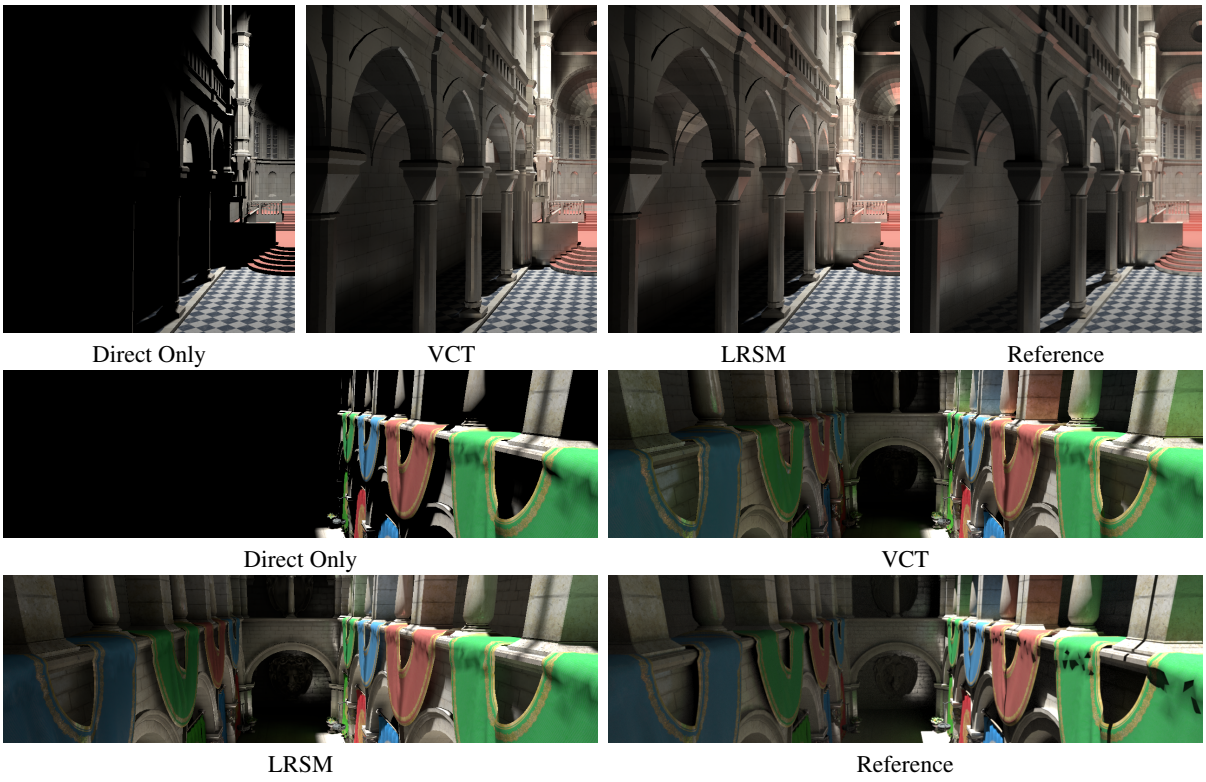


Figure 7: The comparison of VCT, LRSM (our algorithm), and Reference (1024 samples per pixel) in SIBENIK (Top) and SPONZA (Bottom). VCT and LRSM images were rendered by performing cone-based final gathering from every pixel. 6 diffuse cones were shot over the hemisphere with 60° cone aperture, and 1 specular cone was shot toward the mirror reflection with 10° cone aperture if the pixel contains specular components.

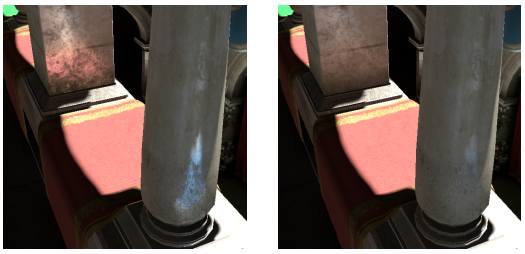


Figure 8: The trade-off between preserving sharp features and avoiding temporal artifacts. The glossy reflection on the right image is clamped for temporal coherence. See the accompany video for more details of this trade-off.

additional geometry not visible from the light sources. Our algorithm also shares some limitations with common shadow map techniques. Our algorithm does not scale well with the number of light sources as we must evaluate one LRSM for each light source. Additionally, temporal coherence with a dynamic light is resolution-dependent. Although our algorithm has few temporal artifacts, it is difficult to completely remove them. Clamping the mip level for LRSM sampling to a lower resolution level can minimize the artifacts with minimal image impact for diffuse indirect illumination because of its low frequency. However, there is a trade-off between temporal coherence and sharp features in glossy indirect illumination. This trade-off is shown in Figure 8 and the accompanying video.

8. Conclusion and Future Work

We have introduced a novel global illumination algorithm which supports both diffuse and glossy indirect illumination. Our simple, memory-friendly, data structures enable fast, temporally coherent results for scenes with fully dynamic geometry and lighting.

However, there is still room for further performance optimizations and quality improvements. We use a static partitioning approach to generate layers, although a more adaptive and automated partitioning approach would be desirable. Our system performs cone-based final gathering for every pixel. Since diffuse indirect illumination is low frequency, this process can be accelerated using interleaved

sampling [SIMP06]. Additionally, the indirect bounce could be extended to support multiple BRDFs with BRDF filtering methods [BN12]. The memory consumption in our algorithm is dominated by occlusion voxels, so we could further reduce memory consumption using a cascaded voxel approach [KD10]. This could also support large, open scenes.

Acknowledgements

We thank the reviewers for their insightful comments, and thank the Advanced Rendering Technology team and Andrew Lauritzen at Intel for their valuable feedback and discussions. We also thank David Blythe, Tom Piazza and Chuck Lingle for supporting this research.

References

[BN12] BRUNETON E., NEYRET F.: A Survey of Nonlinear Prefiltering Methods for Efficient and Accurate Surface Shading. *IEEE Transactions on Visualization and Computer Graphics 18*, 2 (2012), 242–260. 4, 9

[CG12] CRASSIN C., GREEN S.: Octree-Based Sparse Voxelization Using The GPU Hardware Rasterizer. In *OpenGL Insights*, Cozzi P., Riccio C., (Eds.). CRC Press, July 2012, pp. 303–319. 3, 6

[CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive Indirect Illumination Using Voxel Cone Tracing. *Computer Graphics Forum 30*, 7 (2011), 1921–1930. 1, 2, 3, 7

[Cra11] CRASSIN C.: *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes*. PhD thesis, UNIVERSITE DE GRENOBLE, July 2011. 7

[DBB06] DUTRÉ P., BALA K., BEKAERT P.: *Advanced Global Illumination*, 2nd ed. A K Peters, Ltd., 2006. 2

[DGR*09] DONG Z., GROSCH T., RITSCHEL T., KAUTZ J., SEIDEL H.-P.: Real-time Indirect Illumination with Clustered Visibility. In *Vision, Modeling, and Visualization Workshop* (2009), pp. 187–196. 2

[DKH*14] DACHSBACHER C., KŘIVÁNEK J., HAŠAN M., ARBREE A., WALTER B., NOVÁK J.: Scalable Realistic Rendering with Many-Light Methods. *Computer Graphics Forum 33*, 1 (2014), 88–104. 2

[DL06] DONNELLY W., LAURITZEN A.: Variance Shadow Maps. In *Symposium on Interactive 3D Graphics and Games* (2006), pp. 161–165. 5

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent Shadow Maps. In *Eurographics Symposium on Rendering* (2003), pp. 197–201. 4

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective Shadow Maps. In *Symposium on Interactive 3D Graphics and Games* (2005), pp. 203–208. 2, 3, 4

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting Indirect Illumination. In *Symposium on Interactive 3D Graphics and Games* (2006), pp. 93–100. 2

[HN12] HEITZ E., NEYRET F.: Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees. In *High Performance Graphics* (2012), pp. 125–134. 3, 4

[KD10] KAPLANYAN A., DACHSBACHER C.: Cascaded Light Propagation Volumes for Real-Time Indirect Illumination. In *Symposium on Interactive 3D Graphics and Games* (2010), pp. 99–107. 2, 9

[Kel97] KELLER A.: Instant Radiosity. In *Proceedings of SIGGRAPH 97* (1997), ACM, pp. 49–56. 2

[LM08] LAURITZEN A., MCCOOL M.: Layered Variance Shadow Maps. In *Graphics Interface* (2008), pp. 139–146. 5, 6

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental Instant Radiosity for Real-Time Indirect Illumination. In *Eurographics Symposium on Rendering* (2007), pp. 277–286. 2

[Mit12] MITTRING M.: The Technology Behind the Unreal Engine 4 Elemental demo. SIGGRAPH 2012 Advances in Real-Time Rendering in 3D Graphics and Games Course, 2012. 1, 2

[PKD12] PRUTKIN R., KAPLANYAN A., DACHSBACHER C.: Reflective Shadow Map Clustering for Real-Time Global Illumination. In *Eurographics (Short Papers)* (2012), pp. 9–12. 2

[Rau13] RAUWENDAAL R.: *Voxel Based Indirect Illumination using Spherical Harmonics*. PhD thesis, Oregon State University, 08 2013. 2

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The State of the Art in Interactive Global Illumination. *Computer Graphics Forum 31*, 1 (2012), 160–188. 2

[REH*11] RITSCHEL T., EISEMANN E., HA I., KIM J. D. K., SEIDEL H.-P.: Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes. *Computer Graphics Forum 30*, 8 (2011), 2258–2269. 2

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Transactions on Graphics 27*, 5 (2008), 129:1–129:8. 2

[SIMP06] SEGOVIA B., IEHL J. C., MITANCHEY R., PÉROCHE B.: Non-interleaved Deferred Shading of Interleaved Sample Patterns. In *Graphics Hardware* (2006), pp. 53–60. 9

[THGM11] THIEDEMANN S., HENRICH N., GROSCH T., MÜLLER S.: Voxel-based Global Illumination. In *Symposium on Interactive 3D Graphics and Games* (2011), pp. 103–110. 2

[TO12] TOKUYOSHI Y., OGAKI S.: Real-Time Bidirectional Path Tracing via Rasterization. In *Symposium on Interactive 3D Graphics and Games* (2012), pp. 183–190. 2

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree–A Kernel Framework for Efficient CPU Ray Tracing. *ACM Transactions on Graphics* (2014). to appear. 7