Accepted at Pacific Graphics 2011
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin
Authors' version (The definitive version is available at http://diglib.eg.org/ and http://onlinelibrary.wiley.com/)

Volume 30 (2011), Number 7

Interactive Indirect Illumination Using Voxel Cone Tracing

Cyril Crassin[1,2] Fabrice Neyret[1,3] Miguel Sainz[4] Simon Green[4] Elmar Eisemann[5]

[1] INRIA Rhone-Alpes & LJK [2] Grenoble University [3] CNRS [4] NVIDIA Corporation [4] Telecom ParisTech

Figure 1: *Real-time indirect illumination (25-70 fps on a GTX480): We rely on a voxel-based cone tracing to ensure efficient integration of 2-bounce illumination and support diffuse and glossy materials on complex scenes.* *(Right scene courtesy of G. M. Leal Llaguno)*

Abstract

Indirect illumination is an important element for realistic image synthesis, but its computation is expensive and highly dependent on the complexity of the scene and of the BRDF of the involved surfaces. While off-line computation and pre-baking can be acceptable for some cases, many applications (games, simulators, etc.) require real-time or interactive approaches to evaluate indirect illumination. We present a novel algorithm to compute indirect lighting in real-time that avoids costly precomputation steps and is not restricted to low-frequency illumination. It is based on a hierarchical voxel octree representation generated and updated on the fly from a regular scene mesh coupled with an approximate voxel cone tracing that allows for a fast estimation of the visibility and incoming energy. Our approach can manage two light bounces for both Lambertian and glossy materials at interactive framerates (25-70FPS). It exhibits an almost scene-independent performance and can handle complex scenes with dynamic content thanks to an interactive octree-voxelization scheme. In addition, we demonstrate that our voxel cone tracing can be used to efficiently estimate Ambient Occlusion.

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

Keywords: global illumination, indirect lighting, final gather, cone-tracing, voxels, GPU, real-time rendering

1. Introduction

There is no doubt that indirect illumination drastically improves the realism of a rendered scene, but generally comes at a significant cost because complex scenes are challenging to illuminate, especially in the presence of glossy reflections. Global illumination is computationally expensive for several reasons. It requires computing visibility between arbitrary points in the 3D scene, which is difficult with rasterization based rendering. Secondly, it requires integrating lighting information over a large number of directions for each shaded point. Nowadays, with the complexity of rendering content approaching millions of triangles, even in games, computing indirect illumination in real-time on such scenes is a major challenge with high industrial impact. Due to real-time constraints, off-line algorithms used by the special-effect industry are not suitable, and fast, approximate, and adaptive solutions are required. Relying on precomputed illumination is very limiting because common effects such as dynamic light sources and glossy materials are rarely handled.

In this paper, we present a novel algorithm that computes two light bounces (using final gathering [Jen96]) and is entirely implemented on the GPU. It does not suffer from noise or temporal discontinuities, avoids costly precomputation steps, and is not restricted to low-frequency illumination. It exhibits almost scene-independent performance because we manage to mostly avoid involving the actual mesh in our computations. Further, our solution can be extended to out-of-core rendering, hereby handling arbitrarily com-

plex scenes. We reach real-time frame rates even for highly detailed environments and produce plausible indirect illumination (see Teaser).

The core of our approach is built upon a pre-filtered hierarchical voxel representation of the scene geometry. For efficiency, this representation is stored on the GPU in the form of a dynamic sparse voxel octree [CNLE09, LK10] generated from the triangle meshes. We rely on this pre-filtered representation to quickly estimate visibility and integrate incoming indirect energy splatted in the structure from the light sources, using a new approximate voxel cone tracing technique. We handle fully dynamic scenes, thanks to a new real-time mesh voxelization and octree building and filtering algorithm that efficiently exploits the GPU rasterization pipeline. This octree representation is built once for the static part of the scene, and is then updated interactively with moving objects or dynamic modifications on the environment (like breaking a wall or opening a door).

The main contributions of our work are the following:

- A real-time algorithm for indirect illumination;
- An adaptive scene representation independent of the mesh complexity together with a fast GPU-based mesh voxelization and octree-building algorithm;
- An efficient splatting scheme to inject and filter incoming radiance information into our voxel structure;
- An efficient approximate cone-tracing integration;

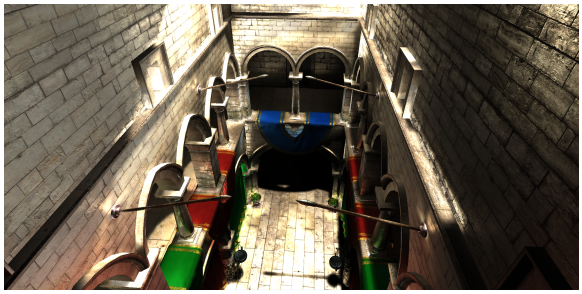


Figure 2: *Our method supports diffuse and glossy reflections.*

2. Previous Work

There are well established off-line solutions for accurate global-illumination computation such as path tracing [Kaj86], or photon mapping [Jen01]. These have been extended with optimizations that often exploit geometric simplifications [TL04, CB04], or hierarchical scene structures [WZPB09], but do not achieve real-time performance. Fast, but memory-intensive relighting for static scenes is possible [LZT*08], but involves a slow preprocessing step. Anti-radiance [DSDD07, DKTS07] allows us to deal with visibility indirectly, by shooting negative light, and reaches interactive rates for a few thousand triangles. Recent GPU implementations of photon mapping [Hac05, WZPB09] use clustering and exploit the spatial coherence of illumination to reduce the sampling cost of final gathering. These approaches allow multiple bounce global illumination, but they still do not reach real-time performances, and suffer from temporal flickering artifacts.

To achieve higher framerates, the light transport is often discretized [DK09]. Particularly, the concept of VPLs [Kel97] is interesting, where the bounced direct light is computed via a set of *virtual point lights*. For each such VPL, a shadow map is computed, which is often costly. Laine et al. [LSK*07] proposed to reuse the shadow maps in static scenes. While this approach is very elegant, fast light movement and complex scene geometry can affect the reuse ratio. Walter et al. [WFA*05] use lightcuts to cluster VPLs hierarchically for each pixel, while Hasan et al. [HPB07] push this idea further to include coarsely sampled visibility relationships. In both cases, a costly computation of shadow maps cannot be avoided and does not result in real-time performance. ISM-like solutions [RGK*08, REG*09, HREB11, RHK*11] reach real-time performance by using a point-based scene approximation to accelerate the rendering into the VPL frusta, but cannot easily ensure sufficient precision for nearby geometry.

The most efficient real-time solutions available today work in the image-space of the current view, but ignore off-screen information [NSW09]. Our approach is less efficient than such solutions, but does not require similarly strong approximations. In particular, we achieve high precision near the viewer which is important for good surface perception [AFO05]. Thiedemann et al. [THGM11] rely on a regular voxel grid to speed-up ray-tracing used for computing visibility with VPLs stored inside a *reflective shadow map* (RSM) [DS05]. It offers real-time performances but only for near-field single bounce indirect illumination and diffuse materials. It is restricted to low-resolution voxel grids, and suffers from classical precision problems of RSMs, as well as noise and flickering artifacts coming from the limited number of rays that can be used in real-time to perform final gathering. Our approach does not suffer from such artifacts and always produce smooth temporally coherent results, thanks to the use of our voxel cone-tracing.

A near-interactive approach has been proposed in [LWDB10] to render glossy reflections with indirect highlights, caused by two successive reflections on glossy surfaces. But this solution is not fast enough to provide real-time performances. To support indirect highlights, our approach uses a simpler, less precise, Gaussian lobe representation to encode a NDF *(Normal Distribution Function)* and a distribution of light directions, that can be convolved with a set of pre-defined BRDFs to reconstruct the radiance response.

Our work derives a hierarchical representation of the scene that produces a regular structure to facilitate light transfer and achieve real-time performance. This structure is similar as the one in [KD10], where diffuse indirect il-

direction ω intersects the scene, else it is one. For practical uses (typically, indoor scenes which have no open sky) the visibility is limited to a distance since the environment walls play the role of ambient diffusors. Hence, we weight occlusion α by a function $f(r)$ which decays with the distance (in our implementation we use $\frac{1}{(1+\lambda r)}$). The modified occlusion is $\alpha_f(p+r\omega) := f(r)\alpha(p+r\vec{\omega})$.

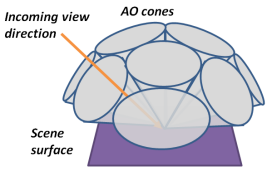


Figure 6: *Ambient Occlusion with a small set of voxel cones.*

To compute the integral $A(p)$ efficiently, we observe that the hemisphere can be partitioned into a sum of integrals: $A(p) = \frac{1}{N}\sum_{i=1}^{N} Vc(p,\Omega_i)$, where $Vc(p,\Omega_i) = \int_{\Omega_i} V_{p,\theta}(cos\theta)d\theta$. For a regular partition, each $Vc(p,\Omega_i)$ resembles a cone. If we factor the cosine out of the Vc integral (the approximation is coarse mainly for large or grazing cones), we can approximate their contribution with our voxel-based cone tracing, as illustrated in Fig. 6, left. The weighted visibility integral $V(p,\omega)$ is obtained by accumulating the occlusion information only, accounting for the weight $f(r)$. Summing up the contributions of all cones results in our approximation of the AO term.

Final Rendering

To perform the rendering of a scene mesh with AO effects, we evaluate the cone-tracing approximation in the fragment shader. For efficiency, we make use of deferred shading [ST90] to avoid evaluating the computation for hidden geometry. *I.e.,* we render the world position and surface normal pixels of an image from the current point of view. The AO computation is then executed on each pixel, using the underlying normal and position.

7. Voxel Shading

For indirect illumination, we will be interested not only in occlusion, but need to compute the shading of a voxel. For this, we have to account for the variations in the embedded directions and scalar attributes, as well as the span of the cone that is currently accumulating that voxel. As shown in [Fou92, HSRG07], this can conveniently be translated into convolutions, provided that the elements are decomposed in lobe shapes. In our case, we have to convolve the BRDF, the NDF, and the span of the view cone, the first already being represented as Gaussian lobes. We consider the Phong BRDF, *i.e.,* a large diffuse lobe and a specular lobe which can be expressed as Gaussian lobes. Nonetheless, our lighting scheme could be easily extended to any lobe-mixture BRDF. As previously discussed, the NDF can be computed from the length of the averaged normal vector $|N|$ that is stored in the voxels, via the approach proposed by [Tok05] ($\sigma_n^2 = \frac{1-|N|}{|N|}$). We fit a distribution to the view cone, by observing (Fig. 7), that the distribution of directions going from a filtered voxel towards the origin of a view cone is the same as the distribution of directions from the origin of the cone towards the considered voxel. We represent this distribution with a Gaussian lobe of standard deviation $\sigma_v = cos(\psi)$, where ψ is the view cone's aperture.

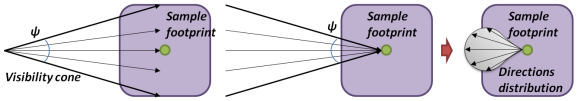


Figure 7: *Illustration of the direction distribution computed on a filtered surface volume from an incident cone.*

8. Indirect Illumination

To compute indirect illumination in the presence of a point light is more involved than AO. We use on a two-step approach. First, we capture the incoming radiance from a light source in the leaves of our scene representation. Storing incoming radiance, not outgoing will allow us to simulate glossy surfaces. We filter and distribute the incoming radiance over all levels of our octree. Finally, we perform approximate cone tracing to simulate the light transport. Writing the incoming radiance in the octree structure is complex, therefore we will, for the moment, assume that it is already present in our octree structure, before detailing this process.

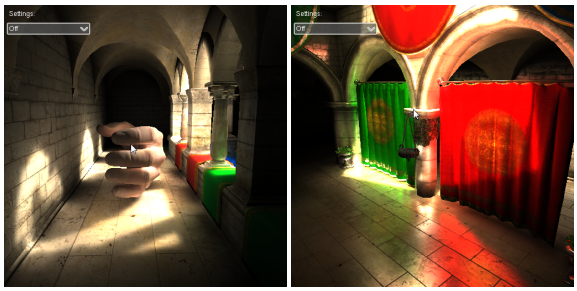
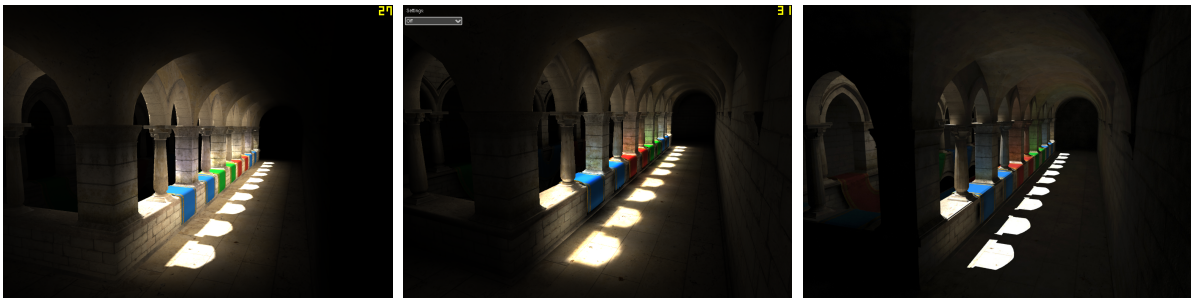


Figure 8: *Left: Indirect diffuse lighting with animated object. Right: Glossy reflections and color bleeding*

8.1. Two-bounce indirect illumination

Our solution works for low-energy/low-frequency and high-energy/high-frequency components of arbitrary material BRDFs, although we will focus our description on a Phong BRDF. The algorithm is similar to the one described in Sec. 6 for AO. We use deferred shading to determine for which surface points we need to compute the indirect illumination. At each such location, we perform a final gathering by sending out several cones to query the illumination that is distributed in the octree. Typically, for a Phong material (Fig. 6, right), a few large cones (typically five) estimate the diffuse energy coming from the scene, while a tight cone in



(a) (b) (c)

Figure 12: *Image quality and performance comparison between LPV (a, 37ms), our approach (b, 32ms) and ground truth (c, 14"12s) rendered with Mental Ray.*

a 512^2 screen resolution. In addition to these costs, the interactive update of the sparse octree structure for dynamic objects (in our case the Wald's hand 16K triangles mesh) takes approximatively 5.5ms per frame. This time depends on the size and number of dynamic parts. The initial creation of the octree for the static environment takes approximatively 280ms. The light injection and filtering pass is computed only when the light source is moved, and takes approximately 16ms. On the sponza scene, with one dynamic object and a moving light, the average framerate is 30FPS with no specular cone traced, and 20FPS with one specular cone, for a 512^2 viewport. For a $1024x768$ viewport (three diffuse and one specular cone), we get an average framerate of 11.4FPS.

Steps	Ras	Dir	Dif	Dir+dif	Spec	Total
Times	1.0	2.0	7.8	14.2	8.0	22.0

Table 1: *Timings (in ms) of individual effects : Mesh rasterisation, Direct lighting, Indirect diffuse, Total direct+indirect diffuse, indirect specular, direct+indirect diffuse and specular. Sponza scene, using 3 diffuse cones and one specular cone (10 deg. aperture)*

We compared the image quality of our approach (Fig.12(b)) with light propagation volumes (LPV) [KD10], the most closely-related real-time solution (Fig.12(a)), and a reference (Fig.12(c)) computed with Mental-Ray final gathering. We used NVIDIA's implementation of LPV in the Direct3D SDK. In all situations, our solution achieves higher visual quality (closer to the reference) and maintains better performance than LPV. Our approach can capture much more precise indirect illumination even far from the observer, with results close to ground truth in this case (cf. Fig. 12: Right wall and ceiling of the corridor, opposite exterior wall seen from left windows), while the LPV nested-grids representation prevents a correct capture of far indirect lighting. Both our approach and LPV suffer from light leaking due to the discrete representation of the geometry and irradiance, which can be observed on the interior side of the arches or the low wall on the left. Leaking appears very high for LPV, while it is more contained in our approach due to our higher voxel resolution.

One of our limitations is the relatively high memory consumption, even with our sparse structure, especially due to the support for indirect specularity. In practice, we allocate roughly 1024MB in the video memory. However, this consumption is usually lower than in [KD10] when trying to achieve comparable visual quality.

Cone aperture (deg)	10	20	30	60
AO	16.6	9.0	6.5	3.9

Table 2: *Timings in ms for the Sponza scene of the ambient occlusion computation for 3 cones and various cone apertures (512^2 resolution).*

Timings for ambient occlusion computations are shown in Table 2 and a comparison with a reference (path-tracing using NVIDIA OptiX) is shown Fig. 13. The differences are mainly due to the tested configuration using only 3 visibility cones. Wide cones suffer from a lack of precision for far visibility, that tends to overestimate occlusions.

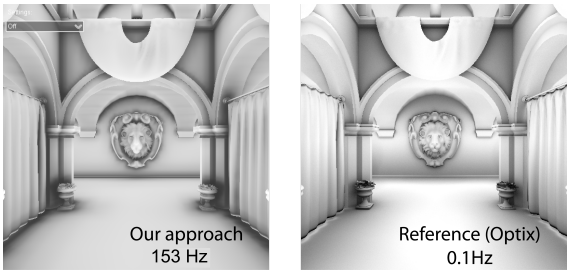


Figure 13: *Comparison with ground truth (OptiX) for ambient occlusion computation.*

11. Conclusion

We presented a novel real-time global-illumination algorithm using approximate voxel-based cone tracing to perform final gathering very efficiently. It is based on a sparse voxel octree structure encoding a pre-filtered representation of the scene geometry and incoming radiance. Using this adaptive representation, we are able to compute approximate indirect lighting in complex dynamic scenes. Our solution