Interactive Volume-based Indirect Illumination of Dynamic Scenes

Athanasios Gaitatzes and Pavlos Mavridis and Georgios Papaioannou

Abstract In this paper we present a novel real-time algorithm to compute the global illumination of dynamic scenes with arbitrarily complex dynamic illumination. We use a virtual point light (VPL) illumination model on the volume representation of the scene. Unlike other dynamic VPL-based real-time approaches, our method handles occlusion (shadowing and masking) caused by the interference of geometry and is able to estimate diffuse inter-reflections from multiple light bounces.

1 Introduction

In order to synthesize photo-realistic images we need to capture the complex interactions of light with the environment. Light follows many different paths distributing energy among the object surfaces. This interplay between light and object surfaces can be classified as local illumination and global illumination. Local illumination algorithms take into account only the light which arrives at an object directly from a light source. Global Illumination algorithms, on the other hand, take into account the entire scene, where the light rays can bounce off the different objects in the environment or be obstructed and absorbed. Reflection, refraction and diffuse inter-reflection are examples of complex light interactions with a high computational cost that is usually not available for real-time applications.

Athanasios Gaitatzes
University of Cyprus, 75 Kallipoleos St., P.O.Box.20537, Nicosia CY-1678, Cyprus,
e-mail: gaitat@yahoo.com

Pavlos Mavridis
Athens University of Economics and Business, Dept. of Informatics, 76 Patission St., Athens 10434, Greece, e-mail: pmavridis@gmail.com

Georgios Papaioannou
Athens University of Economics and Business, Dept. of Informatics, 76 Patission St., Athens 10434, Greece, e-mail: gepap@aueb.gr

In this paper we propose a method that produces photo-realistic images of diffuse, dynamic environments in real time, by estimating the illumination at discrete locations in the environment and applying the results on the scene geometry. This way, we can capture shadowing effects as well as diffuse inter-reflections from multiple secondary light bounces. The method we propose uses a uniform discretization of the scene, incorporating geometry information in the discretization structure. Instead of using the shadow map data as virtual point lights (VPLs) [3] [4] [9], our method performs a complete scene voxelization and is thus able to include occlusion information along with any direct, indirect and self-emitted illumination. Furthermore, it is capable of calculating global illumination from multiple light bounces and include energy from emissive materials in the process.

2 Previous Work

The use of a regular or hierarchical space discretization in global illumination is not new. Several non-real-time algorithms in the past have utilized volume-based acceleration methods and volume data caching to increase their performance. In the past two years, both the porting of global illumination algorithms to the GPU and the inception of new, real-time methods for approximating indirect lighting have gained significant interest from the research community and the game industry.

Radiosity based methods in voxel space have addressed the illumination problem, like Greger et al. [6] and Chatelier et al. [1] but their results were not computed in real-time and had large storage requirements. Modern advances of the same approach, Kaplanyan [9], yielded much faster results than before, but ignored indirect occlusion and secondary light bounces.

The Irradiance Volume, which was first introduced by Greger et al. [6], regards a set of single irradiance samples, parameterized by direction, storing incoming light for a particular point in space (i.e. the light that flowed through that point). The method had large storage requirements as neither an environment map nor spherical harmonics were used for the irradiance storage. With a set of samples they approximated the irradiance of a volume, which was generally time-consuming to compute but trivial to access afterwards. With an irradiance volume they efficiently estimated the global illumination of a scene.

Instant radiosity methods, introduced by Keller [10], approximate the indirect illumination of a scene using a set of VPLs. A number of photons are traced into the scene and VPLs are created at surface hit points, then the scene is rendered, as lit by each VPL. The major cost of this method is the calculation of shadows from a potentially large number of point lights but since it does not require any complex data structures it is a very good candidate for a GPU implementation. Lightcuts [18] reduce the number of the required shadow queries by clustering the VPLs in groups and using one shadow query per cluster, but the performance is still far from real time.

Reflective shadow maps [3] consider the pixels of a shadow map as VPLs, but the contribution of these lights is gathered without taking scene occlusion into account. To achieve interactive frame rates, screen space interpolation is required and the method is limited to the first bounce of indirect illumination. An extension of this method by the same authors [4] reorganizes the computation of the indirect light to achieve better performance, but it still ignores occlusion for the indirect lighting. Imperfect shadow maps [14] use a point based representation of the scene to efficiently render extremely rough approximations of the shadow maps for all the VPLs in one pass. They achieve interactive frame rates but indirect shadows are smoothed out considerably by the imperfections and the low resolution of the shadow maps.

Jensen [7] introduced the concept of photon mapping, where in a first pass photons are traced from the light sources into the scene and stored in a k-d tree and in a second pass the indirect illumination of visible surface points is approximated by gathering the k nearest photons. McGuire [12] computes the first bounce of the photons using rasterization on the GPU, continues the photon tracing on the CPU for the rest of the bounces and finally scatters the illumination from the photons using the GPU. Since part of the photon tracing still runs on the CPU, a large number of parallel cores are required to achieve interactive frame-rates.

Ritchell [15] extends previous methods for screen space ambient occlusion calculation [16] and introduces a method to approximate the first indirect diffuse bounce of the light by only using information in the 2D frame buffer. This method has a very low computational cost but the resulting illumination is hardly accurate since it depends on the projection of the (visible only) objects on the screen.

The concept of interpolating indirect illumination from a cache was introduced by [20]. Accurate irradiance estimates are computed using ray tracing on a few surface points (irradiance sample points) and for the remaining surface points fast interpolation is used. Wang [19] presents a method to calculate the irradiance sample points in advance and implements the algorithm on the GPU. The method is accurate but it achieves interactive frame rates only in very simple scenes.

Nijasure [13] uses spherical harmonics to store the incoming radiance of the scene in a uniform grid structure. The surfaces are rendered by interpolating the radiance from the closest grid points. This method supports multiple bounces and indirect occlusion but it's very expensive because it requires the complete scene to be rendered in a cube map for the radiance estimation on each grid point.

Kaplanyan [9] also uses a regular grid to store the scene radiance, but for its calculation he uses a propagation scheme to calculate the radiance distribution on the scene. Radiance is initially injected in VPL positions and then it is iteratively propagated through empty space. The method achieves very high performance but completely ignores occlusion for the indirect light and secondary bounces.

3 Mathematical Background

3.1 Review of Spherical Harmonics

The spherical harmonics (SH) are a set of orthonormal basis functions defined over a sphere, in the same manner that the Fourier series is defined over an N-dimensional periodical signal. The *Spherical Harmonic* (SH) functions in general are defined on imaginary numbers, but since in graphics we are interested in approximating real functions over the sphere (i.e. light intensity fields), we use the real spherical harmonics basis. Given the standard parameterization of points on the surface of the unit sphere into spherical coordinates

$$(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \rightarrow (x, y, z)$$

the the real SH basis functions of order l is defined as:

$$Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & m > 0 \\ \sqrt{2}K_l^m \sin(-m\phi)P_l^{-m}(\cos\theta) & m < 0 \\ K_l^0 P_l^0(\cos\theta) & m = 0 \end{cases} \tag{1}$$

where $l \in \mathbb{R}^+$, $-l \leq m \leq l$, P_l^m is the *associated Legendre polynomial*, K_l^m is the scaling factor to normalize the functions and is defined as:

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} \tag{2}$$

The spherical harmonics possess several important properties, such as rotation invariance. This means that the SH approximation $\hat{f}$ of a spherical function f rotated by some rotation operator R is the same regardless of the order of application of the rotation: rotating the SH projection of f gives the same approximation as rotating f and then projecting it. This prevents aliasing artifacts from occurring while rotating functions and means that we can simply rotate the projection of a function instead of re-projecting a rotated function.

Similar to the Fourier series expansion, a function on the sphere $f(\theta,\phi)$ can be represented in terms of *spherical harmonics coefficients* $f_{l,m}$ as:

$$f(\theta,\phi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_{l,m}Y_l^m(\theta,\phi) \tag{3}$$

A signal over a sphere is approximately reconstructed using a a truncated SH series, by projecting the initial function f onto a finite set of SH coefficients up to order $l = n$, $l \in \mathbb{N}$. Typically, in computer graphics a maximum order of 6 is used.

Additionally, because of orthnormality, the integral of two reconstructed spherical functions that have been projected in the SH basis is reduced to the inner product

of the vectors of their SH coefficients. For band-limited SH functions of order n, the integral becomes:

$$\int \tilde{f}(\theta,\phi)\tilde{g}(\theta,\phi) = \sum_{l=0}^{n}\sum_{m=-l}^{l} f_{l,m}g_{l,m} \tag{4}$$

3.2 Radiance Transfer

In order to accurately model light in an environment, the complete energy transfer has to be evaluated on each surface location. The *Rendering Equation*, proposed by Kajiya [8], associates the outgoing radiance $L_o(\mathbf{x},\vec{\omega}_\mathbf{o})$ from a surface point $\mathbf{x}$ along a particular viewing direction $\vec{\omega}_o$, with the intrinsic light emission $L_e(\mathbf{x},\vec{\omega}_\mathbf{o})$ at $\mathbf{x}$ and the incident radiance from every direction ω_i in the hemisphere Ω above $\mathbf{x}$, using a BRDF that depends only on the material properties and the wavelength of the incident light. The hemisphere-integral form of the rendering equation can be written as:

$$L_o(\mathbf{x},\vec{\omega}_o) = L_e(\mathbf{x},\vec{\omega}_o) + \int_\Omega L_i(\mathbf{x},\vec{\omega}_i) f_r(\mathbf{x},\vec{\omega}_i \rightarrow \vec{\omega}_o) cos(\theta) d\vec{\omega}_i \tag{5}$$

where $f_r(\mathbf{x},\vec{\omega}_i \rightarrow \vec{\omega}_o)$ is the bidirectional reflectivity distribution function of the surface at point $\mathbf{x}$, expressing how much of the incoming light arriving at $\mathbf{x}$ along direction $\vec{\omega}_i$ is reflected along the outgoing direction $\vec{\omega}_o$. $L_i(\mathbf{x},\vec{\omega}_i)$ is the light arriving along direction $\vec{\omega}_i$.

If we group the cosine term and f_r into a single *transfer function* T, (Sloan et al. [17]), which expresses how point $\mathbf{x}$ responds to incoming illumination, then Equation 5 becomes:

$$L_o(\mathbf{x},\vec{\omega}_o) = L_e(\mathbf{x},\vec{\omega}_o) + L_r(\mathbf{x},\vec{\omega}_o) = L_e(\mathbf{x},\vec{\omega}_o) + \int_\Omega L_i(\mathbf{x},\vec{\omega}_i) T(\mathbf{x},\vec{\omega}_i \rightarrow \vec{\omega}_o) d\vec{\omega}_i \tag{6}$$

The above generic energy transfer equation can be used in fact to model any variation of the rendering equation. In our case, where a point in space (voxel center) is illuminated, it is more convenient to consider an integral over the entire sphere surrounding the voxel center. Furthermore, the voxel center can behave as a spherical particle, receiving energy with maximum flow from every direction. Therefore, the cosine term is dropped as the projected solid angle towards the emitting location along $\vec{\omega}_i$ always equals $d\vec{\omega}_i$.

Using the orthonormality property of the SH function basis (Eq. 4) and considering for the moment only the reflected radiance, the integral in Eq. 6 can be approximated with a finite set of terms as:

$$L_r(\mathbf{x}, \vec{\omega}_o) \approx \sum_{l=0}^{n} \sum_{m=-l}^{l} L_{l,m} T_{l,m} \tag{7}$$

4 Method Overview

We have extended the work of Kaplanyan [9], in order to take into account occlusion in the light transfer process and secondary light bounces. Our method is based on the full voxelization of the geometry instead of the injection of only the reflection shadow map points (VPLs) in a volume grid. This way, the presence of geometry that is unlit by the direct illumination is also known and light interception and reflection is possible. The voxelization records — among others — direct illumination and scalar occupancy data, thus enabling the indirect illumination from emissive materials and the transmission through transparent elements.

Our method consists of three stages. First the scene (or a user-centered part of it) is discretized to a voxel representation. Next, the radiance of each voxel is iteratively propagated in the volume and finally, during image rendering, the irradiance of each surface point is calculated by sampling the radiance from the nearest voxels.

To this effect, we use several 3D volume buffers. An *accumulation volume buffer* is used for the storage of radiance samples when light bounces off occupied voxels. This buffer is sampled during the final rendering pass to reconstruct the indirect illumination. For each color band it stores a spherical harmonic representation of the radiance of the corresponding scene location (4 coefficients encoded as RGBA float values). It is initialized with zero radiance. For the iterative radiance distribution, a *propagation volume buffer* is used (see Section 4.2). The propagation buffer stores a spherical harmonic representation of the radiance to be propagated in the next propagation iteration. It is initialized with the radiance from the first bounce VPLs (direct illumination). Both the accumulation and the propagation buffers are read and write buffers. Our algorithm also reads information from one more read-only volume buffer that contains information about the scene normals and surface albedo (see Fig. 1(b),(c)). Average normals and space occupation (scalar voxelization value, also accounting for transparency) are compacted into a single voxel value.

To discretize our scene in real time, we create a uniform spatial partitioning structure (voxel grid - see Section 4.1) on the GPU, where we store the geometry and radiance samples. We inject VPLs in our voxel space, which are essentially hemispherical lights with a cosine falloff. The VPLs are then approximated by a compact spherical harmonic coefficients representation. See Fig. 1(d) and (e) respectively. Similar to Kaplanyan [9], we use an iterative diffusion approach on the GPU to propagate energy within space. In contrast to [9] though, since we obtain the space occupation information from the voxelization (not-just VPLs), energy is propagated only in void space, from one voxel boundary to the next. The propagated radiance is *reflected* on occupied (voxelized) volume grid points and accumulated at these locations in the accumulation buffer (see Section 4.2). The new propagation direction is determined by the stored average voxel normal of the occupied voxel.

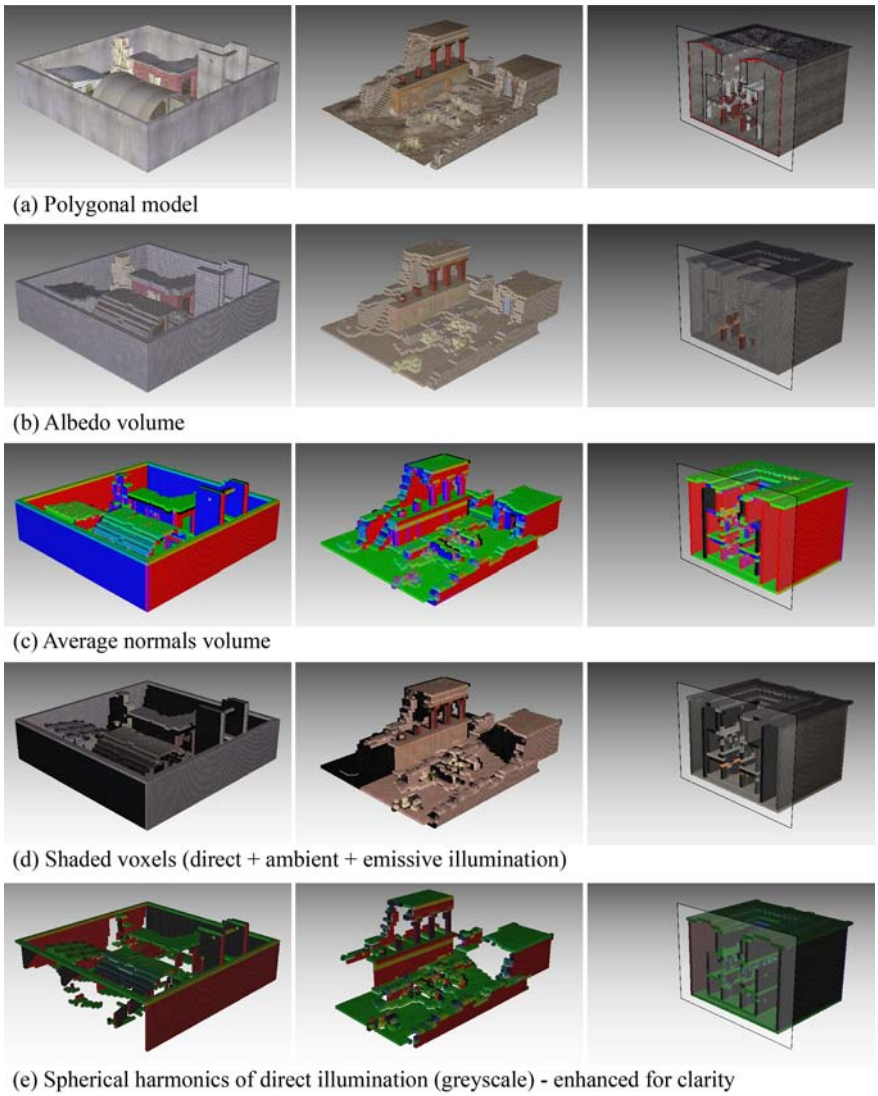

Fig. 1 Several environments voxelized into a 64^3 grid. Column 1: a model of 10,220 triangles (arena). Column 2: a model of 109,170 triangles (Knossos). Column 3: the sponza II atrium of 135,320 triangles (cross section depicted). All buffers are of floating point precision.

During the rendering pass (see Section 4.3), for each fragment, the volume is queried as a texture and the closest texels (accumulated irradiance) are used to estimate the global illumination at that point in the scene.

4.1 Real-Time Voxelization

Instead of applying one of the fast binary GPU voxelization methods, such as Eisemann's et al. [5], we use a variant of Chen's et al. algorithm [2] because we need to store multi-channel scalar data in each voxel. More specifically, we use the same steps as Chen's algorithm, for the slicing of the volume. The main difference is that we do not use the originally proposed XOR operation because in practice, very few models are watertight and many volume attributes cannot be defined for interior voxels. Therefore, our method produces only volume shells.

In brief, for every volume slice (see Algorithm 1), a conventional scan-conversion of the scene geometry takes place and the generated fragments correspond to the voxels of this slice (Fig. 2). Rasterization is incremental and requires that the slope of the dependent variable on the increment is less than 1. As far as the scan conversion of a polygon onto the (slice-oriented) view plane is concerned, there are no holes generated but when the fragments are stored as voxels, discontinuities along the Z-axis occur (slicing direction). The XOR operation indirectly solved this problem (by filling in the missing fragments) but in our case this is not an option. The problem is solved by repeating the scan-conversion process 3 times, once for each primary axis. This way, we ensure that the depth-discontinuity in one orientation of the view plane will be remedied in one of the two others (see Fig. 2(right)).

During the above 3-way voxelization, the radiosity of each grid cell is computed using direct illumination (complete with shadows and emissive illumination). Three buffers on the GPU are needed to store the temporary volume results of the three slicing procedures (one for each different orientation of the object). Finally, those three volumes are combined into one buffer, using the MAX frame buffer blending operation. The maximum radiocity of each cell is stored as a spherical harmonic representation. These values will be used as the initial radiance distribution in the propagation buffer for the iterative radiance distribution.

Algorithm 1: Scene Voxelization

generate a bounding box of the scene;

for $i \leftarrow 1$ **to** *N volume slices* **do**
    define a voxel-deep, thin orthogonal viewing frustum along X-axis;
    execute 2D scan conversion for all object faces;
    store result in slice *i* of volume buffer-X;
end

repeat above loop for Y and Z axes;

combine the three temporary volumes, buffer[XYZ], into one final volume keeping the MAX value for each corresponding cell in all three volumes;

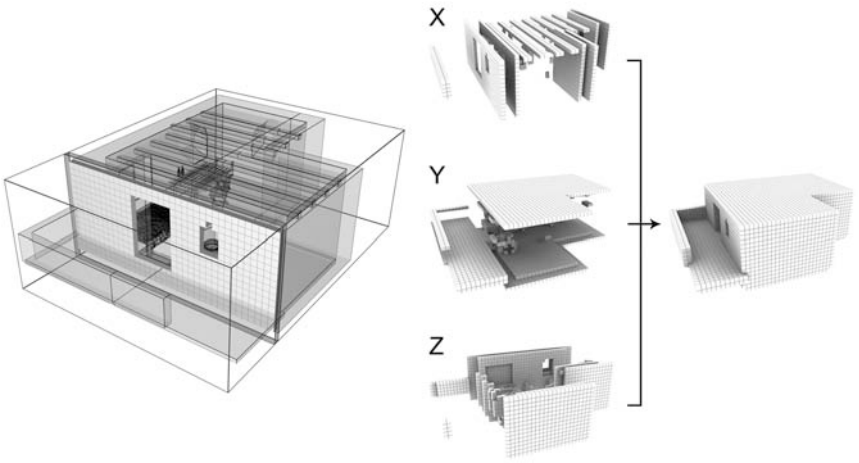

Fig. 2 Slice-based voxelization (left) and composition of the three sub-volume passes into one voxelized volume (right).

4.2 Iterative Radiance Distribution

Once we have injected the VPLs into the initial 3D volume, we need to propagate their initial radiance to their neighboring voxels. The propagation stage consists of several sequential iterations performed entirely on the GPU. Each iteration represents one discrete step of radiance propagation in the (empty) 3D volume. We effectively perform radiance shooting at each volume location by gathering radiance instead from each one of the voxel's neighbors (see Fig. 3) and interpolating the weighted sum of the corresponding directional contributions on the GPU.

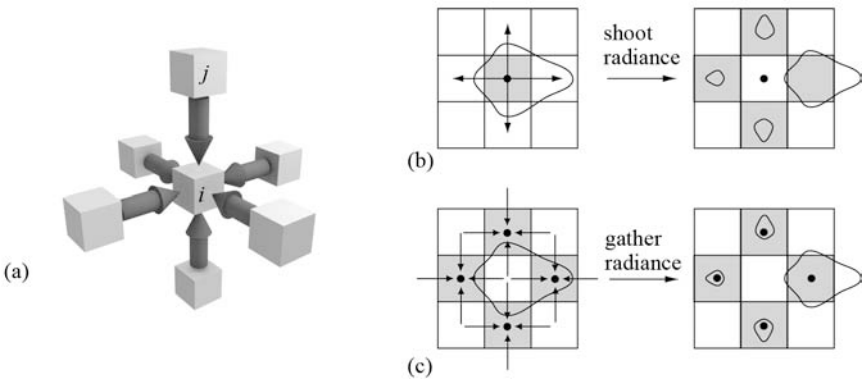

Fig. 3 Radiance Gathering Illustration (a). The radiance for the center voxel is gathered from the values stored at the voxels of the surrounding cells. Radiance shooting (b) in the radiance propagation procedure is equivalent to radiance gathering (c).

Similar to [9], we split the integral of radiance gathering (Eq. 6 for spherical integration domain) into six sub-domains corresponding to the six sides of the receiving voxel. Instead of considering only unobstructed propagation, though, the transfer function $T_{j\to i}$ between the neighboring voxel j and the current voxel i is split into a geometric (transfer) term $T_{cone(j)}$ and a reflective term $T_{r(j)}$. The four $T_{cone(j)}$ coefficients are pre-computed from the six rotated spherical harmonic functions of a 90-degree cone. $T_r(j)$ is used for the deflection of the incident radiance.

When voxel i corresponds to void space, radiance is propagated in the direction from voxel j to i. This means that when no obstacle is encountered, $T_r(j)$ coefficients are equal to 1. On the other hand, when voxel i is occupied, the spherical harmonic function of the incident radiance from voxel j should be mirrored with respect to the plane that is perpendicular to the plane of reflection and parallel to the average normal direction stored in the volume buffer. This requires two SH rotations and a mirroring operation. See [11] for details on the rotation of real spherical harmonics. For speed and simplicity though, this operation is replaced by a mirror reflection on the voxel boundary, i.e. along one of the three primary axes. Therefore, the four coefficients of $T_{r(j)}$ are 1 except the one corresponding to the mirror direction. Taking into account the above factors, the gathering operation becomes:

$$\int_S L_i(\mathbf{x},\vec{\omega}_i)T(\mathbf{x},\vec{\omega}_i\to\vec{\omega}_o)d\vec{\omega}_i = \sum_{j=1}^{6}\sum_{l=0}^{1}\sum_{m=-l}^{l} L_{(j),l,m}T_{cone(j),l,m}T_{r(j),l,m} \qquad (8)$$

Figure 4 demonstrates the propagation and radiance accumulation process.

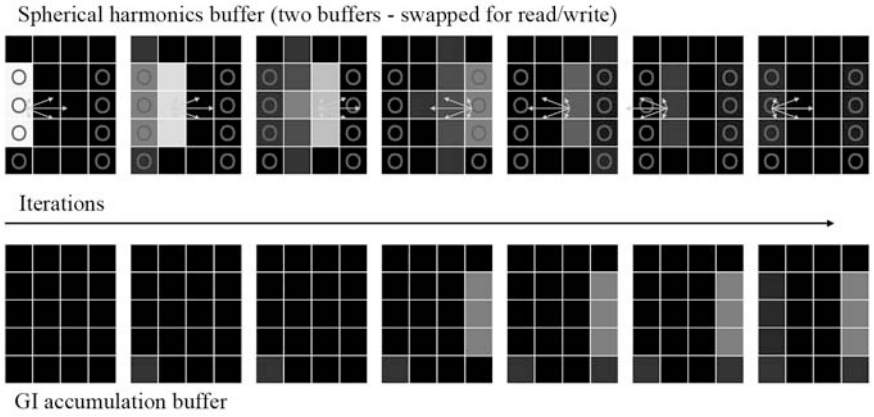

Fig. 4 Simplified example of the propagation and light reflection process.

4.3 Final Illumination Reconstruction

During the final rendering pass, the irradiance at each surface point is computed from the incident radiance L_i that is stored in our uniform grid structure (accumulation buffer). The irradiance E at point $\mathbf{x}$ can be derived by integrating the definition of incident radiance over the hemisphere above $\mathbf{x}$:

$$E(\mathbf{x}) = \int_{\Omega} L_i(\mathbf{x}, \vec{\omega}_i) cos\theta d\vec{\omega}_i \tag{9}$$

where θ is the angle between the surface normal and the incident radiance direction ω_i. The integration domain is the hemisphere Ω defined by the surface normal $\mathbf{n_x}$ at point $\mathbf{x}$.

In order to include the color filtering at the final gathering stage as well as the material emission, we can estimate the radiosity $B(\mathbf{x})$. For diffuse surfaces, $B(\mathbf{x})$ is given by the following hemisphere-integral equation, after multiplying Eq. 5 with π (and hense L_i):

$$B(\mathbf{x}) = B_e(\mathbf{x}) + \frac{\rho(\mathbf{x})}{\pi} \int_{\Omega} B_i(\mathbf{x}, \vec{\omega}_i) cos\theta d\vec{\omega}_i \tag{10}$$

where $\rho(\mathbf{x})$ is the albedo of the surface. If we change the integration domain to the full sphere Ω', the previous equation can be rewritten as follows:

$$B(\mathbf{x}) = B_e(\mathbf{x}) + \frac{\rho(\mathbf{x})}{\pi} \int_{S} B_i(\mathbf{x}, \vec{\omega}_i) T(\mathbf{n_x}, \vec{\omega}_i) \omega_i \tag{11}$$

where the function T is defined as follows:

$$T(\mathbf{n_x}, \vec{\omega}_i) = \begin{cases} \cos\theta, & \theta < \pi/2 \\ 0, & \theta > \pi/2 \end{cases} \tag{12}$$

This change of the integration domain is necessary because we are going to use spherical harmonics, which are defined over the sphere and not the hemisphere.

Equation 11 is directly evaluated per pixel to give the final indirect lighting. In our algorithm the radiance L is tri-linearly interpolated from the stored values in the uniform grid structure. From the eight closest grid points only the ones corresponding to occupied voxels are considered for interpolation. L_i is already stored and interpolated in spherical harmonic representation. We also build a spherical harmonic representation for the function T, as described in [9] and the integral is calculated per pixel as a simple dot product, as shown in Eq. 4.

5 Results

We have integrated the above algorithm in a real time deferred renderer using OpenGL and GLSL. Our proof of concept implementation uses a 2^{nd} order spherical harmonic representation, since the four SH coefficients, map very well to the four component buffers supported by the graphics hardware. All results are rendered on an nVIDIA GeForce GTX285 at 512x512 pixels with a 32^3 grid size. It should be noted here that, excluding the final interpolation stage, the performance of the indirect lighting computation in our method does not depend on the final screen resolution, but only on the voxel grid size and the number of propagation steps. This is a big advantage over instant radiosity methods, like imperfect shadow maps.

Table 1 shows comprehensive time measurements for all the scenes detailed below. All scenes are considered to have fully dynamic geometry and lighting conditions. In all cases our algorithm achieves real time frame rates and sufficient accuracy in the reproduction of the indirect diffuse illumination, even though our implementation is not optimized in any way.

We have found that the propagation stage of our method is limited by the available memory bandwidth and not the computational speed. This is to be expected, since the propagation kernel requires 52 floating point reads and 8 floating point writes per color band. To save memory bandwidth we do not store the diffuse color of the surfaces in the voxel structure, but after the first light bounce we consider it constant and equal to 0.5 for each color band.

Figure 5 shows a direct comparison of our algorithm with a reference solution on a simple test scene. We can see that our method reproduces the shape and the properties of the indirect illumination in the reference solution.

Figure 6 shows a room lit through a set of open windows. This is a challenging scene for global illumination algorithms, because only a small region on the left wall is directly lit by the sun and the majority of the lighting in the room is indirect. We can see that the simple light propagation method completely fails to reproduce the indirect lighting in the room, since it is not taking into account secondary light bounces and occlusion. At least two bounces of indirect lighting are required to get meaningful results in this case. In our method, when a grid of size N is used, the distance between the walls of the room is also N, so kN propagation steps are re-

Table 1 Time measurements of our test scenes in milliseconds. Only the voxelization and propagation times are relevant to our work. The total rendering time includes the direct lighting computation and other effects and is given as a reference.

	triangles	grid size	iterations	voxelization (ms)	propagation (ms)	total time (ms)
test scene	48	32^3	11	10	12	22
room	704	32^3	64	3	61	69
arena	10219	32^3	12	3	13	21
sponza	66454	32^3	11	10	11	28

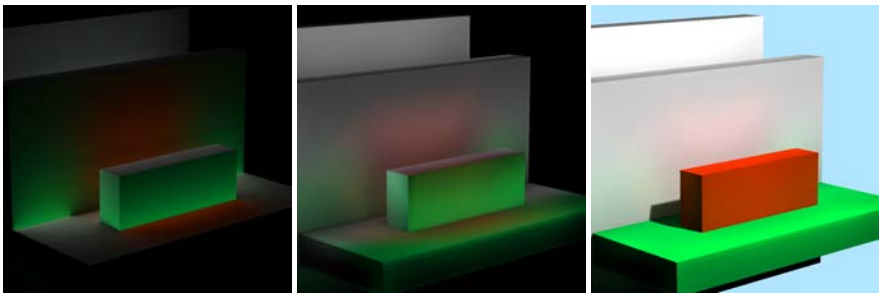

Fig. 5 From left to right: reference solution computed with ray tracing (indirect illumination only), our solution (indirect illumination only) and final image with direct and indirect lighting.

quired to compute k bounces of indirect illumination. This is a worst case scenario, as in typical scenes light interaction from one end of the scene to the other is not required. In this particular case we have used 64 propagation steps to simulate two bounces of light on a 32^3 grid. The resulting illumination is visually pleasing, giving high contrast on the edge of the walls and the staircase. Since our algorithm takes indirect occlusion in consideration, the area below the staircase is correctly shadowed. We observe some artifacts below the windows, due to the imprecision of the spherical harmonics and the fact that the grid cell on this area covers both the edge of the wall and the empty space inside the window. Even with a number of propagation steps this high, our method maintains easily an interactive frame-rate since the propagation stage takes only 61 ms to complete.

A nice characteristic of our method is the predictable performance of the propagation stage. We can easily calculate the time for the propagation step for each individual voxel. This time is constant and independent from the scene complexity. It should be noted of course that the performance may be constant and predictable, but the same is not true for the accuracy and the quality of the resulting illumination.

Figure 7 shows the Sponza Atrium II, a typical scene in the global illumination literature. The right part of the scene is directly lit by the sun, the left one is lit only indirectly. As we can see, using only eleven propagation steps our method successfully reproduces the low-frequency indirect illumination which is dominant on the left part of the scene with very few visible artifacts.

Figure 8 shows an enclosed arena scene, a typical outdoor scene in video games. Twelve propagation steps are used in this case and we can see that the resulting indirect illumination greatly improves the visual quality of the final image.

5.1 Discussion

A nice feature of our method is that for scenes with static or smoothly changing geometry and lighting conditions, the cost of the indirect illumination can be amortized among many frames without introducing any visible artifacts. In other words,

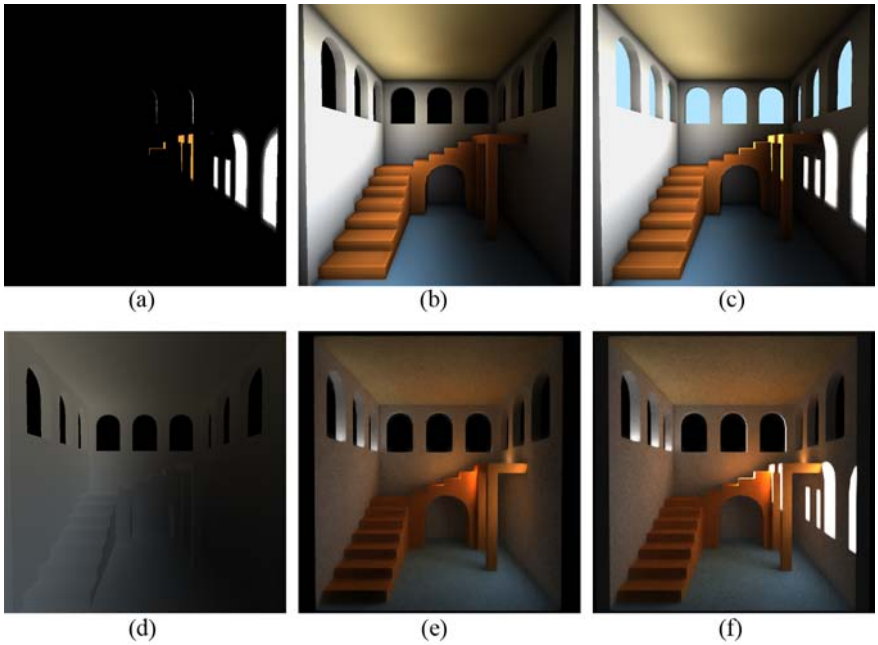

Fig. 6 (a) The room scene lit with direct lighting only. (b) Radiosity with 64 iterations. (c) Direct and indirect illumination using our method. (d) The indirect illumination using light propagation volumes [9]. (e) Reference radiosity using 2-bounce path tracing. (f) Reference final image using path tracing.

the rate of indirect lighting updates can be reduced to meet the final performance goals. For scenes with good temporal coherence — hence, with slow illumination changes — it is possible to perform the 3-way voxelization in an interleaved manner (one direction per frame). In this case the volume is completely updated after three frames but the voxelization cost is reduced by a factor of three.

Fig. 7 From left to right: direct lighting, indirect illumination only and final image with direct and indirect lighting.

Fig. 8 From left to right: direct only lighting, indirect illumination using our method and final image with direct and indirect lighting.

Since voxelization is a rough discretization of the scene geometry, secondary shadows from small scale geometric details cannot be reproduced accurately by our method. Higher voxel resolutions can always be used, but with a performance hit. Also, due to graphics hardware limitations, we only used second order spherical harmonics, which they do not have sufficient accuracy to represent high frequency indirect light. This is not crucial if the direct illumination covers large parts of a scene yielding only very low-frequency indirect shadows in the first place. Interestingly, imperfect shadow maps have exactly the same issue (but for different reasons) but we think that our method is preferable since it does not require the maintenance of a secondary point based scene representation and the performance is mostly independent from final image resolution.

The performance and quality of our method depends on two parameters: the volume resolution and the number of iterations. Empirically, we have found that a grid size of 32 is sufficient in most cases. For outdoor scenes we have found that a low iteration count (around 12) is sufficient but for indoor ones a much higher iteration count is required (around 64) to accurately simulate the bouncing of the light inside the building.

Fig. 9 The Knossos model (http://graphics.cs.aueb.gr/downloads/knossos.rar). From left to right: direct lighting, radiosity using our method and final image with direct and indirect lighting.

6 Conclusion and Future Work

We have presented a new method for the computation of indirect diffuse light transport in large and fully dynamic scenes in real-time. Unlike previous work, our method takes in to account indirect occlusion and secondary light bounces. We have demonstrated that our method gives good results in a variety of test cases and always maintains a high frame rate. Since the test results showed that the voxelization step is relatively costly, in the future we intent to introduce a much faster voxelization scheme. Furthermore, the possibility of a more accurate but still manageable radiance deflection mechanism will be investigated. Finally, another interesting direction of research is to extend this method to take specular light transport in to account.

References

1. Chatelier, P.Y., Malgouyres, R.: A low-complexity discrete radiosity method. Computers and Graphics **30**(1), 37–45 (2006). URL http://dx.doi.org/10.1016/j.cag.2005.10.008
2. Chen, H., Fang, S.: Fast voxelization of three-dimensional synthetic objects. Journal of Graphics Tools **3**(4), 33–45 (1998)
3. Dachsbacher, C., Stamminger, M.: Reflective shadow maps. In: I3D '05: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, pp. 203–231. ACM, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1053427.1053460
4. Dachsbacher, C., Stamminger, M.: Splatting indirect illumination. In: I3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, pp. 93–100. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1111411.1111428
5. Eisemann, E., Décoret, X.: Single-pass GPU solid voxelization for real-time applications. In: GI '08: Proceedings of Graphics Interface 2008, vol. 322, pp. 73–80. Canadian Information Processing Society, Toronto, Ontario, Canada (2008)
6. Greger, G., Shirley, P., Hubbard, P.M., Greenberg, D.P.: The irradiance volume. IEEE Computer Graphics and Applications **18**(2), 32–43 (1998). URL http://dx.doi.org/10.1109/38.656788
7. Jensen, H.W.: Global illumination using photon maps. In: Proceedings of the eurographics workshop on Rendering techniques '96, pp. 21–30. Springer-Verlag, London, UK (1996)
8. Kajiya, J.T.: The rendering equation. In: SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, vol. 20, pp. 143–150. ACM, New York, NY, USA (1986). DOI http://doi.acm.org/10.1145/15922.15902
9. Kaplanyan, A.: Light propagation volumes in cryengine 3. In: SIGGRAPH '09: ACM SIGGRAPH 2009 courses. ACM, New York, NY, USA (2009)
10. Keller, A.: Instant radiosity. In: SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 49–56. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997). DOI http://doi.acm.org/10.1145/258734.258769
11. Křivánek, J., Konttinen, J., Pattanaik, S., Bouatouch, K., Žára, J.: Fast approximation to spherical harmonics rotation. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches, p. 154. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1179849.1180042
12. McGuire, M., Luebke, D.: Hardware-accelerated global illumination by image space photon mapping. In: HPG '09: Proceedings of the Conference on High Performance Graphics 2009, pp. 77–89. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1572769.1572783

13. Nijasure, M., Pattanaik, S., Goel, V.: Real-time global illumination on the GPU. Journal of Graphics Tools **10**(2), 55–71 (2005)
14. Ritschel, T., Grosch, T., Kim, M.H., Seidel, H.P., Dachsbacher, C., Kautz, J.: Imperfect shadow maps for efficient computation of indirect illumination. In: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers, vol. 27, pp. 1–8. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1409060.1409082
15. Ritschel, T., Grosch, T., Seidel, H.P.: Approximating dynamic global illumination in image space. In: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, I3D '09, pp. 75–82. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1507149.1507161
16. Shanmugam, P., Arikan, O.: Hardware accelerated ambient occlusion techniques on gpus. In: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D '07, pp. 73–80. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1230100.1230113
17. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 527–536. ACM, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/566570.566612
18. Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., Greenberg, D.P.: Lightcuts: a scalable approach to illumination. In: ACM Transactions on Graphics, vol. 24, pp. 1098–1107. ACM, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1186822.1073318
19. Wang, R., Wang, R., Zhou, K., Pan, M., Bao, H.: An efficient gpu-based approach for interactive global illumination. In: SIGGRAPH '09: ACM SIGGRAPH 2009 papers, pp. 1–8. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1576246.1531397
20. Ward, G.J., Rubinstein, F.M., Clear, R.D.: A ray tracing solution for diffuse interreflection. In: SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques, pp. 85–92. ACM, New York, NY, USA (1988). DOI http://doi.acm.org/10.1145/54852.378490