Deferred Voxel Shading for Real-Time Global Illumination

José Villegas and Esmitt Ramírez
Computer Graphics Center, Computer Science Department
Faculty of Sciences, Central University of Venezuela
Caracas, Venezuela. 1010-A
villegasjose.gg@gmail.com, esmitt.ramirez@ciens.ucv.ve

***Abstract*—Computing indirect illumination is a challenging and complex problem for real-time rendering in 3D applications. We present a global illumination approach that computes indirect lighting in real time utilizing a simplified version of the outgoing radiance and the scene stored in voxels. This approach comprehends 2-bounce indirect lighting for diffuse, specular and emissive materials. Our voxel structure is based on a directional hierarchical structure stored in 3D textures with mipmapping, the structure is updated in real time utilizing the GPU which enables us to approximate indirect lighting for dynamic scenes. Our algorithm employs a voxel-light pass which calculates voxel direct and global illumination for the simplified outgoing radiance. We perform voxel cone tracing within this voxel structure to approximate different lighting phenomena such as ambient occlusion, soft shadows and indirect lighting. We demonstrate with different tests that our developed approach is capable to compute global illumination of complex scenes on interactive times.**

***Keywords*—Global Illumination, Voxelization, Voxel Shading, Cone Tracing, GPU, Real-Time Rendering**

I. Introduction

Realistic image synthesis has been always an important field in computer graphics. A precise computation of the light distribution within a scene is a main aspect to achieve realism, this computation is often expensive since light just does not follow a straight path, it propagates, bounces, scatters and it is absorbed by different objects on a scene.

The standard rendering pipeline utilizes triangles to represent the geometry of a scene, these triangles are later rasterized, shaded and finally showed on a screen. This representation is effective for local fragment operations such as direct lighting, but it possesses many limitations for more complex operations like global illumination.

Some modern techniques utilize a simplified version of the scene with voxels, even making possible to approximate global illumination at interactive framerates. A voxelized scene avoids many of the problems caused by the increasingly complex geometry in scenes and its relation with many global illumination approaches that depend on the polygon count.

In this paper we present an approach to calculate real-time global illumination on diffuse, glossy and emissive surfaces using a discretized version of the scene with voxels and cone tracing. The main contributions of our approach can be summarized as follows:

- A real-time voxelization process that stores the necessary information to capture direct and indirect lighting per voxel.
- An efficient voxel-based lighting pass to store the outgoing radiance that includes solutions for voxel occlusion, and voxel normal attenuation error.
- An efficient voxel global illumination process using cone tracing that extends the filtered outgoing radiance with the indirect diffuse term.
- A real-time algorithm for indirect illumination using voxel cone tracing.

Therefore, it is important to comprehend the main idea of global illumination approximations in the research field of computer graphics (see Section II). Also, Section III shows the related work essential to our research. Next, we overview the complete process of our approach in Section IV.

II. Global Illumination

Global illumination is a process that represents a huge challenge in 3D graphics applications, its goal is to calculate the distribution of the light energy within a scene. An accurate portrayal of this effect comprehends a plethora of light phenomena such as: indirect lighting, indirect shadows, caustics, reflections, color blending, soft shadows, scattering, ambient occlusion and others. These effects are subtle but they greatly increase final quality of an image adding realism, sensation of depth and increased immersion.

The fundamental light transport equation, which is used to describe the global illumination over a point on a surface, was introduced by Kajiya [1] in 1986. This equation is called the rendering equation and it describes the outgoing radiance from a point x in the direction Θ as $L(x \to \Theta)$, where this term is equivalent to the equation 1.

$$L_e(x \to \Theta) + \int_{\Omega_x} f_r(x, \Psi \to \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) dw_\Psi \tag{1}$$

The equation describes the outgoing radiance as the self-emitted outgoing radiance at the point x in direction Θ expressed as $L_e(x \to \Theta)$, plus the incident radiance from the lit hemisphere Ω^+ that is reflected at x in direction Θ. This integral comprehends the product of the bidirectional reflectance distribution function (BRDF) as f_r, the incident radiance as

$L(x \leftarrow \Psi)$, and the normal attenuation as $\cos(N_x, \Psi)$, where Ψ is the direction of the light, and N_x the surface normal at the point x.

A precise and complete calculation of the light distribution in a scene, at a steady state, is complex task even with current graphical hardware. This makes such task not suitable for real-time rendering on complex and dynamic scenes (e.g. video-games and simulations).

III. Previous Work

There are many well known approaches to compute precise global illumination. Techniques such as path tracing [1], photon mapping [2] or radiosity [3] provide off-line solutions with highly accurate results. These techniques aren not meant for real-time rendering applications.

With the increase of computing power in modern graphics hardware and the many features of current graphics APIs and the rendering pipeline, many real-time approximations have been proposed. These techniques all have disadvantages and advantages while only approximating a set of the many existing light phenomena. A complete study of several techniques based on geometry, screen-space or hybrid is presented by Ritschel et al. [4].

Certain approaches simplify the required global lighting equations including only certain lighting effects or reducing them to an approximation. Techniques such as ambient occlusion assume the incident light as uniform and white, while only considering the visibility of fragments, for this approximation approaches such as Screen Space Ambient Occlusion (SSAO) [5] and Horizon-Based SSAO [6] are widely used but they only approximate the ambient exposure of fragments in the scene.

Other techniques discretize the light distribution to achieve higher performance. Techniques such as Instant Radiosity [7] create a set of Virtual Point Lights (VPLs) to approximate the distribution of the light within a scene. For each generated VPL a shadow map is generated which can be heavy on performance, the approach also suffers singularities for geometry near the VPLs, these singularities also create temporal coherence issues. Reflective Shadow Mapping (RSM) [8] utilizes shadow mapping to obtain information of the first bounce of light, this technique can achieve real-time global illumination but only for diffuse surfaces, it also presents visibility problems for overlapping geometry. Light Propagation Volumes (LPV) discretizes the light into a regular grid and utilizes RSMs to propagate the light within this regular grid, this technique also only approximates only the indirect diffuse term and it presents light leaking and geometry occlusion issues.

Simplification of the geometry is another common approach to approximate global illumination [9], [10], [11]. Particularly, Thiedemann et al. [12] utilizes a voxelization approach and ray-voxel intersection to query indirect lighting, this technique only approximates the indirect diffuse term and it presents visibility issues for emitting geometry occluded for the camera. Crassin et al. [13] propose another approach to calculate indirect lighting with cone tracing and a pre-filtered voxel representation of the whole scene in a octree, this technique approximates global illumination for glossy and diffuse surfaces. Though this technique achieves real time performance its light injection and brick fill process present a heavy toll on performance for constant dynamic updates.

Our approach uses a voxelization process to simplify the geometry of the scene. We also employ GPGPU (General-Purpose Computing on Graphics Processing Units) to calculate the direct lighting, occlusion, and indirect lighting of this simplified voxel structure. The structure is also filtered onto different levels of details using GPGPU, these levels enable our technique to approximate indirect lighting with cone tracing on a final gathering scheme with pre-filtered values.

IV. Overview

Our proposed approach is a four step algorithm as seen in Fig. 1. First we discretize the scene geometry using a hierarchical voxel representation stored in 3D textures, for this a conservative voxelization process is used. Only the necessary lighting data for diffuse illumination, such as normal and albedo, is stored in voxels using an average operation for all the fragments within the space of a voxel, we also store emission to approximate light emitting surfaces.

In a second step the outgoing radiance is computed per every voxel. For direct diffuse illumination standard lighting techniques are used. An average operation is used during voxelization which can cause unwanted normal directions for the resulting averaged normal, this can produce shading issues for the normal attenuation term. To reduce this problem we propose a normal-weighted directional shading model for the normal attenuation. To calculate occlusion, apart from standard shadow mapping techniques [14], raycasting can also be used, a ray is traced from the voxel position in the light direction and a voxel volume is sampled to test for ray-voxel collision.

Then at the third step, we fill the levels of the voxel representation hierarchy using anisotropic filtering to generate directional voxels. For this, directional integration and an average operation is used to filter the values from a higher to a lower level of detail in the hierarchy per axis direction, the level of details are stored in the mipmap levels for 3D textures.

To approximate a second bounce of indirect light another step can be added after filtering the values from the outgoing radiance computation. The necessary information to calculate the indirect diffuse component using cone tracing is available in the voxel structure at this point. For each voxel we perform cone tracing to approximate the indirect diffuse component, then the filtering step is repeated, this way the outgoing radiance in the voxel representation now contains the direct lighting and the first bounce of indirect diffuse.

Finally, in our last step voxel the cone tracing algorithm [13] is used to approximate global illumination phenomena such as diffuse and specular reflection and ambient occlusion. For each visible fragment, direct and indirect illumination are computed. For the indirect component a final gathering

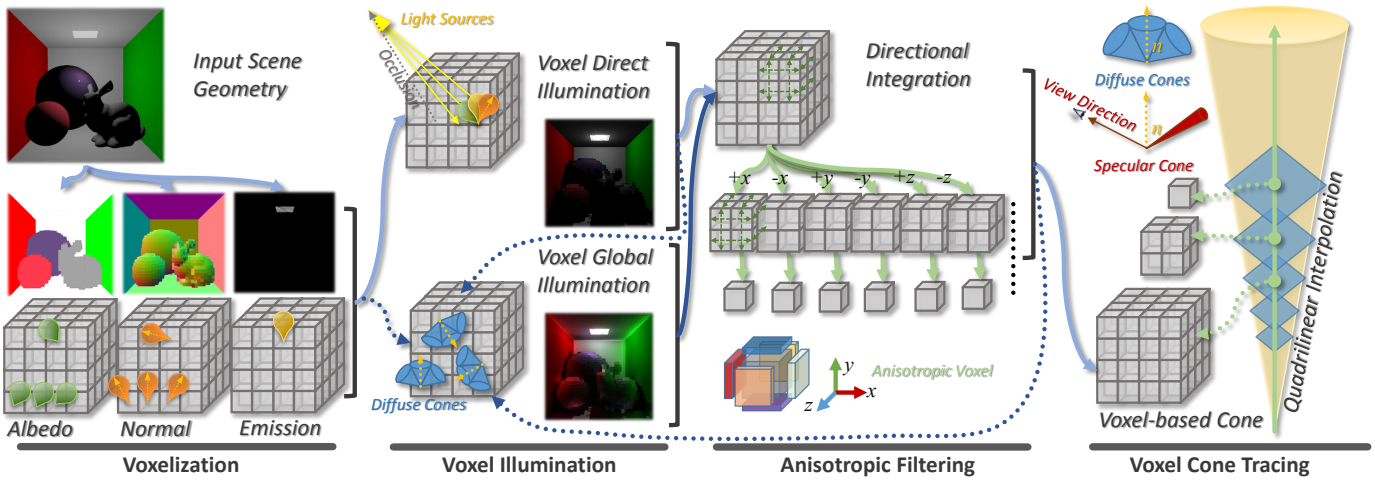


Fig. 1. Illustration of our four steps approach for global illumination.

scheme is used, cones are traced over the normal oriented hemisphere to collect the outgoing radiance stored within the voxel representation.

Next sections describe in detail each step of our proposal, including the test and applied results. Section V describes the voxelization algorithm, and the data structures involved to accomplish our proposal. In the Section VI, the approximation to the indirect illumination process is described. The third step of our proposal, the anisotropic filtering, is presented in Section VII and the fourth, the voxel cone tracing algorithm, is presented in Section VIII. In the Section IX we described the process to obtain the visual final composition of the scene. Tests are performed and shown in Section X to profile and show results of our approach. Finally, Section XI presents the conclusions and future work of this research.

V. Voxel Structure

Our voxel structure is inspired by deferred rendering [15], where a G-Buffer contains relevant data to later be used in a separate light pass. Normal, albedo and emission values are stored in voxels during the voxelization process, every attribute has its own 3D texture associated as it can observed at the leftmost side of Fig. 2. This information is sufficient to calculate the diffuse reflectance and normal attenuation on a separate light pass where, instead of computing the lighting per pixel it is done per voxel. The structure can be extended to support a more complicated reflectance model but this may imply a higher memory consumption to store additional data.

Furthermore, another structure is used for the voxel cone tracing pass. The resulting values of the lighting computations per voxel are stored in another 3D texture which we will call *radiance volume*. To approximate the incrementing diameter of the cone, and its sampling volume, levels of detail of the voxelized scene are used. For anisotropic voxels six 3D textures at half resolution of the radiance volume are required as shown in Fig. 2, one per every axis direction positive and negative, the levels of details are stored within the mipmap levels of these textures which we will call *directional volumes*.

The radiance volume represents the maximum level of detail for the voxelized scene, this texture is separated from the directional volumes. To bind these two structures, linear interpolation is used between samples of both structures when the mipmap level required for the diameter of the cone ranges between, the maximum level and the first filtered level of detail.

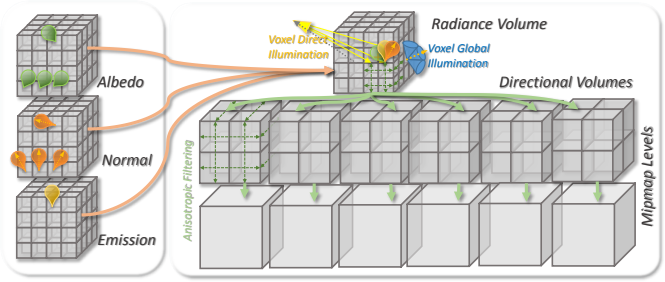


Fig. 2. A visualization of our voxel structure. The left box illustrates the volumes from the voxelization process. The right box illustrates the voxel structure used for cone tracing, this structure captures and filters the outgoing radiance onto mipmap levels.

A. Conservative Voxelization

The voxelization process is done using a conservative voxelization scheme, meaning that if geometry exists within the space of a voxel there must be a voxel in that space. For this process we utilize a single pass algorithm as described by Crassin and Green [16]. This technique exploits the observation made by Schwarz and Seidel [17] where, a thin surface voxelization of a triangle A can be computed for each voxel B by testing if the plane defined by the vertexes of A intersects B, and the 2D projection of the triangle A along the dominant axis of its normal intersects the 2D projection of B.

To ensure conservative voxelization, a conservative rasterization approach as described by Hasselgren et al. [18] is

used. For each projected triangle, a slightly larger bounding polygon is generated to ensure that no matter how small the projected triangle is, it will generate at least one fragment. To create this polygon each triangle edge is shifted outward in order to enlarge the triangle. The bounding polygon does not overestimate the coverage of the projected triangle, therefore the excess fragments are discarded using an extended bounding box defined by the vertexes of the projected triangle.

A voxel can contain many fragments, and all those fragments can have different values for the attributes to store (i.e. albedo, normal and emission). To obtain a good approximation for the discretized geometry, an average operation for all the values within the space of a voxel per attribute is applied. Atomic operations to write and read attributes at each voxel position in their respective 3D texture must be used to ensure time coherent results.

B. Dynamic Update

For dynamic updates, the conservative voxelization of static and dynamic geometry are separated. While static voxelization happens only once, dynamic voxelization happens per frame or when needed. The voxelization of both types of geometry happen on the same process described above, hence a way to indicate which voxels are static is needed. In our approach we use a single value 3D texture to identify voxels as static, this texture will be called *flag volume*.

During static voxelization, after a voxel is generated a value is written to the flag volume indicating this position as static. In contrast, during the dynamic voxelization, before generating a voxel, a value is read from the flag volume at the writing position of the voxel, if the value indicates this position is marked as static then writing is discarded, leaving the static voxels untouched.

To constantly revoxelize the scene it is necessary to clear from the 3D textures the previous stored dynamic voxels. This is done before the dynamic voxelization using general-purpose computing on graphics processing units (GPGPU). The flag volume is read to clear voxels under two conditions: if the voxel exists and if its dynamic.

VI. Voxel Direct Illumination

To properly approximate the indirect illumination during the cone tracing step, it is necessary to store the incoming radiance in the radiance volume of the voxel structure. Inspired by deferred rendering, a voxel light pass where we calculate the direct illumination per voxel is performed. Though, unlike deferred rendering where the light pass is executed for every visible pixel, here the lighting computations are done per every voxel visible or not, this is necessary to avoid visibility problems since the structure is used to approximate indirect lighting. In this case occluded or non-visible geometry for the camera may or may not contribute to the light distribution for the visible fragments. To speed up this process the GPU executes a thread per every voxel reserved for the radiance volume, based on GPGPU approach.

A. Shading

For the Lambert reflectance model, the shading can be calculated per voxel using the information stored in the 3D textures from the voxelization process. The following equation is calculated per voxel for every light source to describe the voxel radiance V_r:

$$V_r = L_i \frac{\rho}{\pi} max\{N \cdot \Psi, 0\} \tag{2}$$

where L_i is the light source intensity, ρ the albedo of the voxel, N the normal vector of the voxel and Ψ the light direction. For non-uniform light sources such as spotlights and pointlights, the position of the voxel is needed. This can be obtained projecting the position of the voxel in texture space to world space.

B. Normal-Weighted Attenuation

In the section VI-A when the normal vector of the voxel is used for the normal attenuation term $(N \cdot \Psi)$ it may give unwanted results. Since the normal vectors are stored using an average operation, the resulting averaged normal may end up pointing towards a non-convenient direction. This problem is notable when the normal vectors within the space of a voxel are uneven as show in Fig. 3.

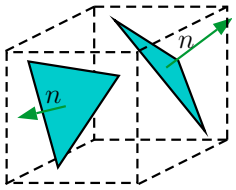


Fig. 3. Uneven normal vectors within the space of a voxel.

To reduce this issue our proposal utilizes a normal-weighted attenuation, where first the normal attenuation is calculated per every face of the voxel as follows:

$$D_{x,y,z} = (\hat{i} \cdot \Psi, \hat{j} \cdot \Psi, \hat{k} \cdot \Psi) \tag{3}$$

Then three dominant faces are selected depending on the axes sign of the averaged normal:

$$D_\omega = \begin{cases} max\{D_\omega, 0\}, & N_\omega > 0 \\ max\{-D_\omega, 0\}, & \text{otherwise} \end{cases} \tag{4}$$

Finally, the resulting attenuation is the product of every dominant face normal attenuation, multiplied with the weight per axis of the averaged normal vector of the voxel, the resulting reflectance model is computed as follows:

$$\begin{aligned} W &= N^2 \\ V_r &= L_i \frac{\rho}{\pi}(W_x D_x + W_y D_y + W_z D_z) \end{aligned} \tag{5}$$

C. Occlusion

To generate accurate results during the cone tracing step, voxels need to be occluded, otherwise voxelized geometry that is supposed to have little no outgoing radiance will contribute to the indirect lighting calculations.

The classic shadow mapping [14] or alike techniques can be used to compute the voxels occlusion. The position of the voxel is projected in light space and the depth of the projected point is compared with the stored depth from the shadow map to determine if the voxel is occluded. A simple improvement over this technique is: instead of using the voxel center position V_p, the position is translated along the normal vector of the voxel by half voxel size V_{size} as $V_p = V_p + N \times V_{size} \times 0.5$, this exposes the voxel position further in case the center position may be occluded by geometry near the voxel.

Our proposal also computes occlusion using raycasting within a volume. Any of the resulting volumes from the voxelization process can be used since the algorithm only needs to determine if a voxel exists at a certain position. To determine occlusion of a voxel, a ray is traced from the position of voxel in the direction of the light, the volume is sampled to determine if at the position of the ray there is a voxel (right image in Fig. 4), if this condition is true then the voxel is occluded.

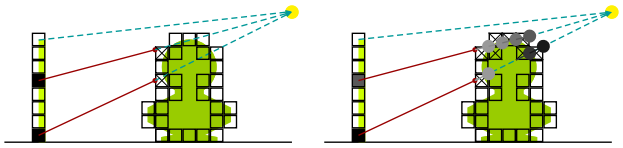


Fig. 4. Raycasting to determine if a voxel is occluded. Stopping as soon a collision is found (left) and accumulation per collision (right).

Instead of stopping the ray as soon a voxel is found, soft shadows can be approximated with a single ray accumulating a value κ per collision and dividing by the traced distance t, i.e. $\nu = \nu + (1-\nu)\kappa \div t$, where $1-\nu$ represents the occlusion value after the accumulation is finished. This technique exploits the observation that, from the light point of view, the number of collisions will usually be higher for the rays that pass through the borders of the voxelized geometry (left image in Fig. 4).

D. Emission

Adding to the final radiance value the emission term can be used approximate emissive surfaces such as area lights, neon lights, digital screens, etc., this provides a crude approximation of the emission term $L_e(x \rightarrow \Theta)$ in the Eq. 1. After the direct illumination value is obtained the emission term from the voxelization process is added to this value per voxel. During the cone tracing step, these voxels will appear to be bright even on occluded areas, hence indirect light is accumulated per cone from these regions of the voxelized scene.

VII. Anisotropic Voxels

For more precise results during the cone tracing step anisotropic voxels are used. The mipmapping levels, as seen in the directional volumes in Fig. 2, will store per every voxel six directional values, one per every directional axis positive and negative. Each cone has an origin, aperture angle and direction, this last factor determines which three volumes are sampled. The directional sample is obtained by linearly interpolating the three samples obtained from the selected directional volumes.

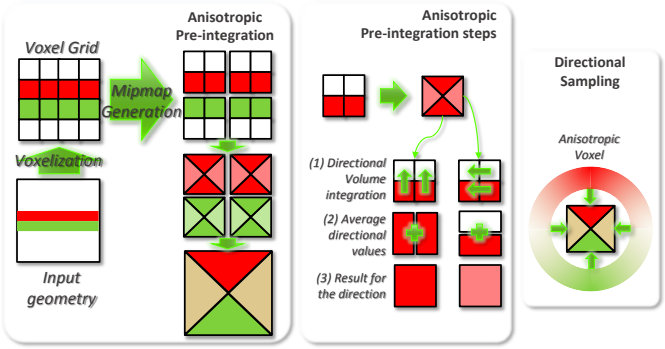


Fig. 5. The left box illustrates the voxel mipmapping process with anisotropic voxels. The process to calculate a directional value is illustrated in the middle box. The right box shows the range of values with directional sampling for a voxel. Image taken from [13].

To generate the anisotropic voxels we use the process detailed by Crassin et al. [13]. To compute a directional value a step of volumetric integration is done in depth and the directional values are averaged to obtain the resulting value for a certain direction, the Fig. 5 describes this process. In our approach this is done using GPGPU executing a thread per every voxel at the mipmap level that is going to be filtered using the values from the previous level, this process is done per every mipmap level.

VIII. Voxel Cone Tracing

Voxel cone tracing is similar to ray marching, the cone advances a certain length every step, except that the sampling volume increases along the diameter of the cone. The mipmap levels in the directional volumes are used to approximate the expansion of the sampling volume during the cone trace, to ensure smooth variation between samples quadrilinear interpolation is used which is natively supported with graphics hardware for 3D textures.

The shape of the cone is meant to exploit the spatial and directional coherence of the many rays packed within the space of a cone. This behavior is used in many approaches such as packet ray-tracing [19].

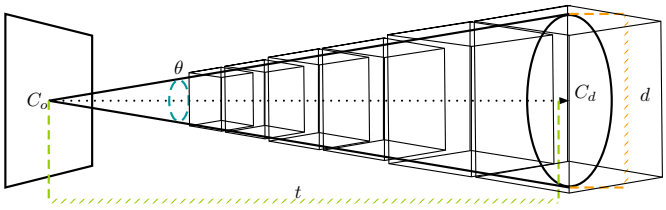


Fig. 6. Description of a cone used for voxel cone tracing.

As seen in Fig. 6 each cone is defined by an origin C_o, a direction C_d and an aperture angle θ. During the cone steps the diameter of the cone is defined by d, this value can be extracted using the traced distance t with the following equation: $d = 2t \times \tan(\theta \div 2)$. Which mipmap level should be sampled depending on the diameter of the cone can be obtained using the following equation: $V_{level} = log_2(d \div V_{size})$, where V_{size} is the size of a voxel at the maximum level of detail.

To accumulate values along the cone we utilize the emission-absorption model described by Nelson Max [20] and Hadwiger et al. [21]. For voxel cone tracing this process is described by Crassin et al. [13]. For each cone trace we keep track of the occlusion value α and the color value c which represents the indirect light towards the cone origin C_o. In each step we retrieve from the voxel structure the occlusion value α_2 and the outgoing radiance c_2. Then the c and α values are updated using volumetric front-to-back accumulation as follows: $c = \alpha c + (1 - \alpha)\alpha_2 c_2$ and $\alpha = \alpha + (1 - \alpha)\alpha_2$. To ensure good integration quality between samples, in our approach the distance d' between steps is modified by a factor β. With $\beta = 1$ the value of d' is equivalent to the current diameter d of the cone, values less than 1 produce higher quality results but require more samples which reduces the performance.

In our approach this process is done in screen-space for efficiency, using deferred rendering. For each visible fragment a set of cones are traced for different effects which will be detailed in this section.

A. Indirect Illumination

Indirect lighting is approximated with a crude Monte Carlo approximation. The hemisphere region for the integral in the rendering equation (Eq. 1) can be partitioned into a sum of integrals. For a regular partition, each partitioned region resembles a cone. For each cone we approximate their contribution using voxel cone tracing, the resulting values are then weighted to obtain the accumulated contribution at the cones origin.

The distribution of the cones matches the shape of the BRDF (Fig. 7), for a Blinn-Phong material a few large cones distributed over the normal oriented hemisphere estimate the diffuse reflection, while a single cone in the reflected direction, where its aperture depends on the specular exponent, approximates the specular reflection.

B. Ambient Occlusion

Ambient occlusion can be approximated using the same cones used for the diffuse reflection for efficiency. For the ambient occlusion term δ we only accumulate the occlusion value α_2, at each step the accumulated value is multiplied with the weighting function $f(r) = \frac{1}{1+\lambda r}$, where r is the current radius of the cone and λ an user defined value which controls how fast $f(r)$ decays along the traced distance. At each cone step the ambient occlusion term is updated as: $\delta = \delta + (1 - \delta)\alpha_2 f(r)$.

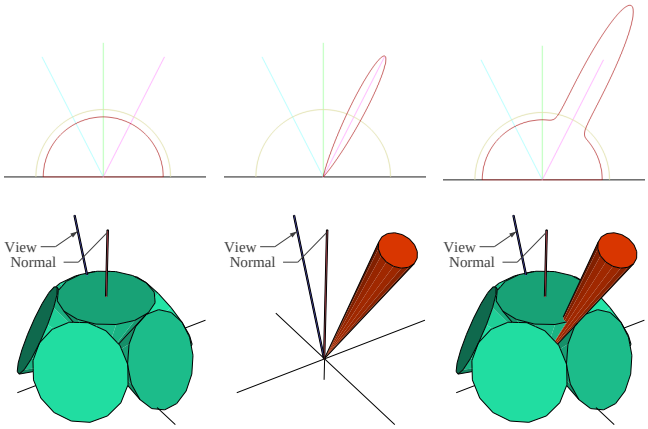


Fig. 7. Cone distribution a Blinn-Phong BRDF (down) and polar plots for the components of the Blinn-Phong BRDF (up), diffuse (right), specular (center), diffuse and specular (left).

C. Soft Shadows

Cone tracing can also be used to achieve soft shadows tracing a cone from the surface point x towards the direction of the light (Fig. 8). The cone aperture controls how soft and scattered the resulting shadow is. For soft shadows with cones we only accumulate the occlusion value α_2 at each step.

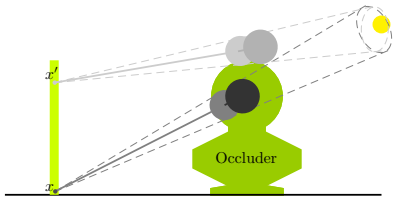


Fig. 8. Occlusion accumulation with a cone to achieve soft shadows.

IX. Voxel Indirect Diffuse

To calculate the diffuse reflection over a surface point using voxel cone tracing we need its normal vector, albedo and the incoming radiance at that point. Since we voxelize the geometry normal vectors and the albedo into 3D textures, all the needed information for the indirect diffuse term is available after calculating the voxel direct illumination. In our approach we perform voxel cone tracing per voxel using GPGPU to calculate the first bounce of indirect diffuse at each voxel. This step is done after the outgoing radiance values from the voxel direct illumination pass are anisotropically filtered.

For each voxel we use its averaged normal vector to generate a set of cones around the normal oriented hemisphere to calculate the indirect diffuse at the position of the voxel. The weighted result from all the cones is then multiplied by the albedo of the voxel and added to the direct illumination value. The resulting outgoing radiance for the radiance volume now stores the direct illumination, and the first bounce of indirect diffuse. The anisotropic filtering process needs be repeated for the new values. This enables us to approximate the second

bounce of indirect lighting during the final voxel cone tracing step per pixel.

X. Tests & Results

Tests on this section were executed on an AMD 380 R9 GPU with an AMD Phenom II X6 1055T CPU. The tests were performed to determine the performance and image quality of certain sections of the algorithm on different conditions. All the tests and images referenced in this section were generated with a voxel representation of 256^3 voxels and a screen resolution of $1280x720$ pixels unless indicated otherwise. All the times show in this section are in milliseconds (ms).

We employ a variety of scenes with increasing geometric complexity as show in the Table I. These scenes can be seen in the Fig. 9 rendered with our approach. For the tests the scenes are considered as static, to trigger dynamic voxelization we included dynamic objects such as Stanford Buddha with 50K triangles, Stanford Bunny with 13K triangles and Stanford Dragon with 70K triangles.

TABLE I
ATTRIBUTES FOR THE USED SCENES.

Name	Model	# of vertexes	# of triangles
S1	Cornell Box	72	36
S2	Sibenik Cathedral	40.479	75.283
S3	Crytek Sponza	153.635	278.163

The Table II, shows the times per frame for the dynamic and static voxelization of each scene. For the voxelization process, a higher resolution for the voxel representation can actually be beneficial, because it reduces thread collisions for the writing operations, in the scene S3 we observe a higher time for the static voxelization for this same reason, S3 has a higher triangle density per voxel meaning more syncing operations are needed.

TABLE II
DYNAMIC AND STATIC VOXELIZATION TIMES (MS).

	Static	Clear Dynamic	Dynamic
S1	0.51	0.78	1.30
S2	1.80	0.58	2.11
S3	11.29	0.60	2.03

After the voxelization step we proceed to compute the illumination per voxel using GPGPU. The Table III shows the results for voxel direct illumination with shadow mapping (SM) or raycasting for voxel occlusion (RC) and voxel global illumination, it also includes the time for anisotropic filtering (AF) after both steps. For voxel direct illumination the times are similar between all scenes using shadow mapping, while raycasting costs more performance it enables occlusion for any type of light source without shadow mapping. For raycasting, the amount of empty space in the scene affects how early the traced rays end, which affects the general performance. For high density scenes such as S3 most rays end early, in contrast the scene S1 takes a considerable amount of time using raycasting because the scene is mostly empty, this same condition also applies for the voxel global illumination step. For the voxel global illumination we utilize four cones with an aperture of 45 degrees to obtain the indirect diffuse term. The times for the anisotropic filtering are nearly the same for all scenes since this operation is done at each position of the regular grid of voxels, its performance only depends on the resolution of the voxel representation.

TABLE III
TIMES (MS) FOR VOXEL DIRECT ILLUMINATION WITH SHADOW MAPPING (SM) OR RAYCASTING (RC), VOXEL GLOBAL ILLUMINATION AND ANISOTROPIC FILTERING (AF).

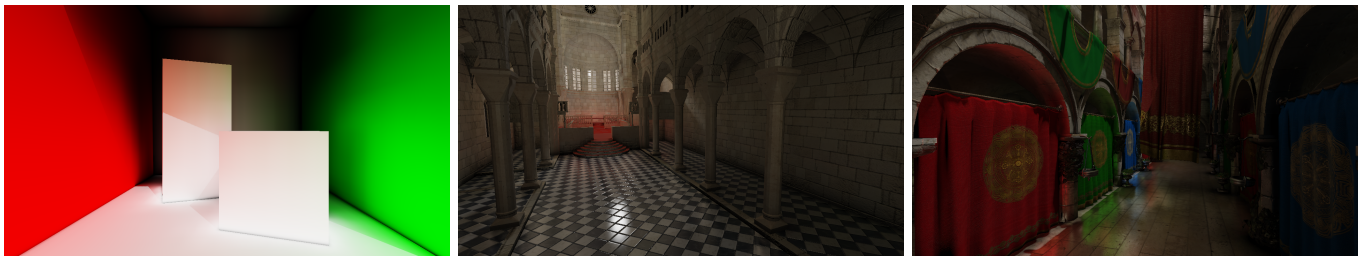
	Direct (SM)	Direct (RC)	AF	Global	AF
S1	1.33	20.32	1.39	8.41	1.38
S2	0.95	4.57	1.38	3.88	1.37
S3	1.13	3.31	1.37	5.44	1.38

For visual results on the voxel illumination step the Fig. 10 shows direct illumination (right) and global illumination (left) of voxels for the scenes S3 (top) and S1 (bottom). Lighting phenomena such as color blending are observable in both images. The scene S1 uses raycasting for occlusion and S3 shadow mapping. Fig 12 shows the difference between our normal-weighted attenuation and standard normal attenuation, color is recovered for many regions in the scene S3, specially the curtains.

The Fig. 13 shows the results for voxel occlusion with raycasting on the scene S2. The right image shows hard shadows, which means stopping the ray as soon a collision is found, and the left image shows soft shadows accumulating collision values along the traced distance.

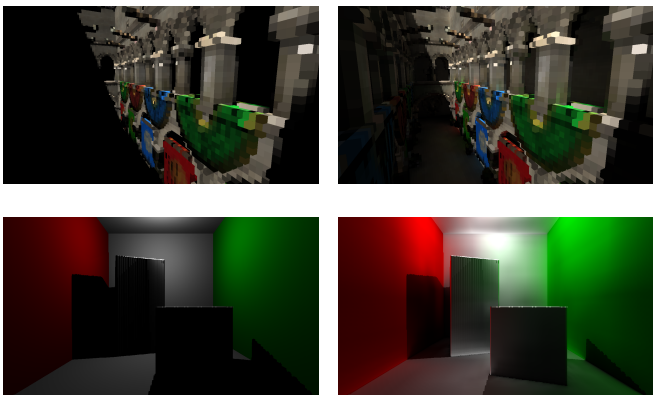
Once the voxel structure is anisotropically filtered we can proceed with voxel cone tracing per pixel for the final composition of the image. In our approach six cones with an aperture of 60 degrees are used to approximate the indirect diffuse term, for the indirect specular in these test we utilize a specular cone with an aperture of 10 degrees. The Table IV shows the performance for indirect lighting on different scenes. For all the scenes we achieve results over 30FPS ($<$ 33.3ms) under constant dynamic update, meaning objects and lights are changing and moving per frame. The dynamic update includes dynamic voxelization, voxel direct illumination, voxel global illumination and the necessary filtering steps. For a screen resolution of $1920x1080$ pixels we obtain an average framerate of 28.57ms for the scene S3, 27.02ms for S2 and 27.77ms for S1 under constant dynamic update, these results show that even with a high resolution we achieve over 30FPS for all the scenes.

Cone tracing can also be used to achieve soft shadows, the Table V shows the times for different cone apertures. A thinner cone means more time spend sampling the voxel structure, which affects performance, this condition also applies for the specular cones. The Fig. 14 shows the results for different cone



(a) Cornell Box (b) Sibenik Cathedral (c) Crytek Sponza

Fig. 9. Scenes used to tests our global illumination approach.



(c) Voxel Direct Illumination (d) Voxel Global Illumination

Fig. 10. Comparison of voxel illumination with direct light only (right) and indirect diffuse (left).

TABLE IV
TIMES (MS) DIRECT LIGHTING, INDIRECT LIGHTING AND CONSTANT DYNAMIC UPDATE.

	Direct	Indirect (Diff + Spec)	Direct + Indirect	Dynamic
S1	1.03	7.27	7.67	17.81
S2	1.32	7.62	8.32	17.60
S3	1.34	7.81	8.41	16.97

apertures, a bigger aperture affects how soft and scattered the resulting shadow looks.

TABLE V
TIMES (MS) FOR SOFT SHADOWS WITH CONES WITH DIFFERENT APERTURES.

Cone Aperture (degrees)	1	5	10	25
S1	14.25	10.31	8.22	7.63

Ambient occlusion can also be computed using the same cones for diffuse reflection. Fig. 11 shows the results for ambient occlusion only on all the scenes.

The resolution of the voxel representation affects all the steps of the algorithm. The Table VI shows the average times on constant dynamic update, with different scenes for different resolutions for the voxel representation. The decrease in performance is observable when using a higher resolution such as 512^3.

TABLE VI
TIMES (MS) CONSTANT DYNAMIC UPDATE WITH DIFFERENT RESOLUTIONS FOR THE VOXEL REPRESENTATION.

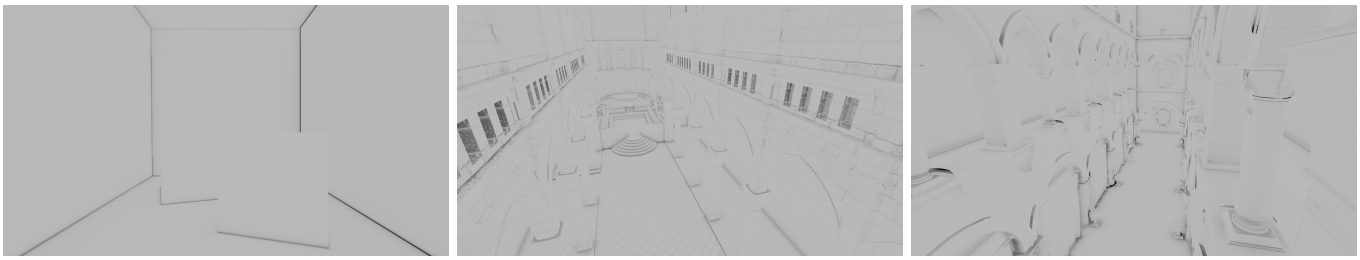
Voxels	S1	S2	S3
512^3	81.21	65.55	71.29
256^3	16.82	17.63	17.81
128^3	1.97	3.93	4.21

In the Fig. 15, we compared our approach in image quality and performance against the implementation of light propagation volumes (LPV) included in NVIDIA Direct3D SDK, and a reference image rendered in Blender Cycles. There is a difference in quality against the reference image as ground truth in the scene S3 for both approximations. LPV was chosen because this approach also uses a discretization of the scene. LPV only approximates indirect diffuse, therefore soft reflections on the ground and columns are missing. Our approach is closer to the reference compared to LPV, especially in the occluded areas on the right side of the image and the top corridor.

Fig. 16 shows lighting results from only emissive voxels, there are no light sources present in the scenes apart from the emissive materials. In the image (a) color blending between emissive materials is observable, note also that emissive materials can have any arbitrary shape. Image (b) and (d) show soft shadows, which are a by-product of the cone tracing step since it generates indirect shadows, in this case occluded geometry receives less outgoing radiance from the emissive voxels. Image (c) shows fine detail emission with texture mapping, left and right walls have different lighting colors.

XI. Conclusion & Future Work

We have presented a real-time global-illumination algorithm that uses a simplified representation of the scene with voxels, and cone tracing over this voxel representation to perform final gathering over a surface point. The voxelized representation stores the outgoing radiance of the scene and it is filtered onto many levels of details, for our approach we capture the direct light and the first bounce of indirect diffuse in this structure. For voxel direct illumination, we presented a reflectance model which avoids some issues caused by voxelized normal



(a) Cornell Box (S1) (b) Sibenik Cathedral (S2) (c) Crytek Sponza (S3)

Fig. 11. Voxel-based ambient occlusion with cone tracing.

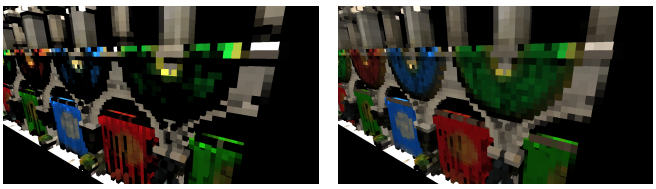
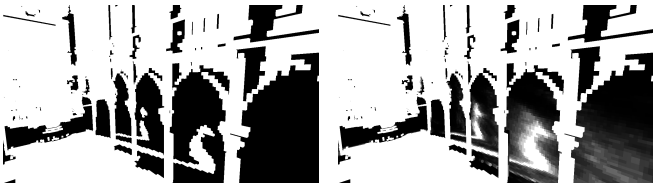
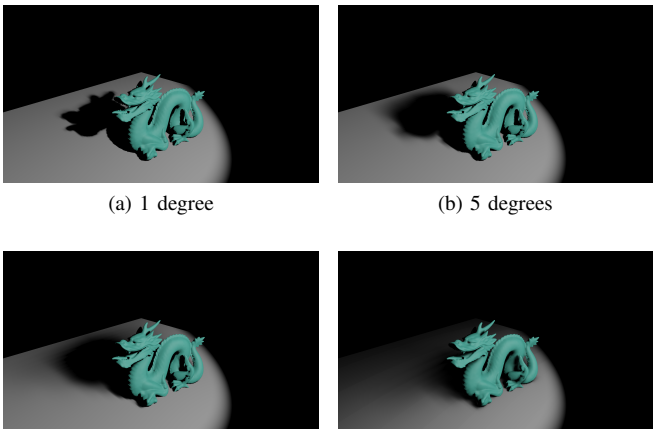


Fig. 12. Comparison of standard normal attenuation (right) and normal-weighted attenuation (left) for voxel direct illumination.



(a) Hard Shadows (b) Soft Shadows

Fig. 13. Comparison of voxel occlusion with hard shadows and soft shadows.



(a) 1 degree (b) 5 degrees

(c) 10 degrees (d) 25 degrees

Fig. 14. Different apertures soft shadows with cone tracing.

vectors and the normal attenuation term. As for occlusion we propose raycasting within a volume for voxels occlusion and soft shadows with cone tracing for fragments occlusion, this represents an alternative for light sources without shadow mapping. Our second bounce of indirect light is approximated calculating the first bounce of indirect diffuse within the voxelized representation with cone tracing.

Our approach approximates indirect lighting with one or two bounces of light for diffuse and glossy surfaces. This solution achieves many complex lighting phenomena such as diffuse reflection, specular reflection, indirect shadows, color blending, ambient occlusion, soft shadows and emissive materials. Although, this approach being an approximation does present some subtle image issues such as light leaking, product of the geometry simplification, and color banding, caused by the sampling steps within the voxel structure during cone tracing.

As future work we consider investigating other voxelization approaches such as solid voxelization to avoid light leaking issues and to include other effects such as translucent objects [22], or techniques as 3D clipmap [23]. Memory consumption is another issue since we use 3D textures, we are interested in tiled resources and sparse textures to produce a more compact voxel representation, without a considerable hit on performance. We are also interested in image-space methods since the cone tracing step can be separated as a post-process, for example upsampling could be used to perform the cone tracing at a lower resolution and dithering can reduce color banding issues.

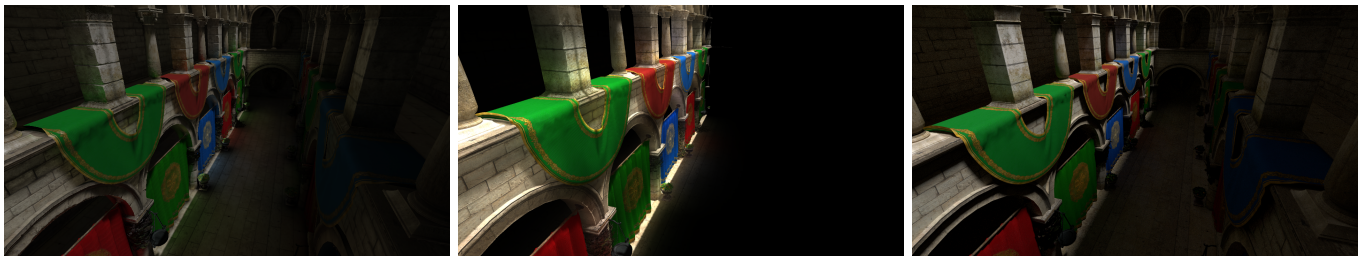
REFERENCES

[1] J. T. Kajiya, "The Rendering Equation," in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '86. New York, NY, USA: ACM, 1986, pp. 143–150. [Online]. Available: http://doi.acm.org/10.1145/15922.15902

[2] H. W. Jensen, *Realistic Image Synthesis Using Photon Mapping*. Natick, MA, USA: A. K. Peters, Ltd., 2001.

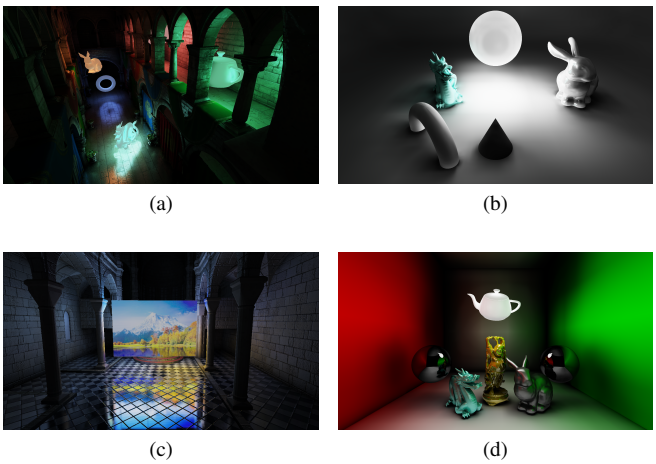
[3] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces," in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '84. New York, NY, USA: ACM, 1984, pp. 213–222. [Online]. Available: http://doi.acm.org/10.1145/800031.808601

[4] T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz, "The state of the art in interactive global illumination," *Comput. Graph. Forum*, vol. 31, no. 1, pp. 160–188, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2012.02093.x



(a) Our approach (17.34ms) (b) LPV (18.86ms) (c) Reference ($\sim$ 3 hours)

Fig. 15. Image quality and performance comparison between our approach, light propagation volumes (LPV) and a reference image.



(a) (b)

(c) (d)

Fig. 16. Collection of images with emissive materials.

[5] M. Mittring, "Finding next gen: Cryengine 2," in *ACM SIGGRAPH 2007 courses*. ACM, 2007, pp. 97–121.

[6] L. Bavoil, M. Sainz, and R. Dimitrov, "Image-space horizon-based ambient occlusion," in *ACM SIGGRAPH 2008 talks*. ACM, 2008, p. 22.

[7] A. Keller, "Instant radiosity," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 49–56.

[8] C. Dachsbacher and M. Stamminger, "Reflective shadow maps," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. ACM, 2005, pp. 203–231.

[9] F. Sans and E. Ramírez, "Real-Time Diffuse Global Illumination based on Voxelization," in *XXXIX Latin American Computing Conference*, ser. Latin American Symposium of Computer Graphics, Virtual Reality, and Image Processing, vol. 2. ACM, 2013, pp. 39–50.

[10] H. Malan, "Real-Time Global Illumination Using Slices," in *GPU Pro 6: Advanced Rendering Techniques*, W. Engel, Ed. A K Peters/CRC Press, 2015, ch. 5, pp. 267–293.

[11] Y.-Y. Chen and S.-Y. Chien, "Lighting-driven voxels for memory-efficient computation of indirect illumination," *The Visual Computer*, pp. 1–9, 2016.

[12] S. Thiedemann, N. Henrich, T. Grosch, and S. Müller, "Voxel-based global illumination," in *Symposium on Interactive 3D Graphics and Games*. ACM, 2011, pp. 103–110.

[13] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive Indirect Illumination Using Voxel Cone Tracing: A Preview," in *Symposium on Interactive 3D Graphics and Games*, ser. I3D '11. New York, NY, USA: ACM, 2011, pp. 207–207. [Online]. Available: http://doi.acm.org/10.1145/1944745.1944787

[14] L. Williams, "Casting Curved Shadows on Curved Surfaces," in *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '78. New York, NY, USA: ACM, 1978, pp. 270–274. [Online]. Available: http://doi.acm.org/10.1145/800248.807402

[15] T. Saito and T. Takahashi, "Comprehensible Rendering of 3-D Shapes," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '90. New York, NY, USA: ACM, 1990, pp. 197–206. [Online]. Available: http://doi.acm.org/10.1145/97879.97901

[16] C. Crassin and S. Green, "Octree-Based Sparse Voxelization Using the GPU Hardware Rasterizer," in *OpenGL Insights*, P. Cozzi and C. Riccio, Eds. CRC Press, July 2012, pp. 303–317, http://www.openglinsights.com/.

[17] M. Schwarz and H.-P. Seidel, "Fast Parallel Surface and Solid Voxelization on GPUs," in *ACM SIGGRAPH Asia 2010 Papers*, ser. SIGGRAPH ASIA '10. New York, NY, USA: ACM, 2010, pp. 179:1–179:10. [Online]. Available: http://doi.acm.org/10.1145/1866158.1866201

[18] J. Hasselgren, T. Akenine-Möller, and L. Ohlsson, "Conservative Rasterization," *GPU Gems*, vol. 2, pp. 677–690, 2005.

[19] I. Wald, W. R. Mark, J. Gnther, S. Boulos, T. Ize, W. Hunt, S. G. Parker, and P. Shirley, "State of the Art in Ray Tracing Animated Scenes," *Computer Graphics Forum*, vol. 28, no. 6, pp. 1691–1722, 2009. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2008.01313.x

[20] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, Jun. 1995. [Online]. Available: http://dx.doi.org/10.1109/2945.468400

[21] M. Hadwiger, J. M. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel, *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006.

[22] E. Eisemann and X. Décoret, "Single-pass GPU Solid Voxelization for Real-time Applications," in *Proceedings of Graphics Interface 2008*, ser. GI '08. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2008, pp. 73–80. [Online]. Available: http://dl.acm.org/citation.cfm?id=1375714.1375728

[23] A. Panteleev, "Practical Real-Time Voxel-Based Global Illumination for Current GPUs," in *ACM SIGGRAPH 2014 presentations*. ACM, 2014, available on http://on-demand.gputechconf.com/gtc/2014/presentations/S4552-rt-voxel-based-global-illumination-gpus.pdf.

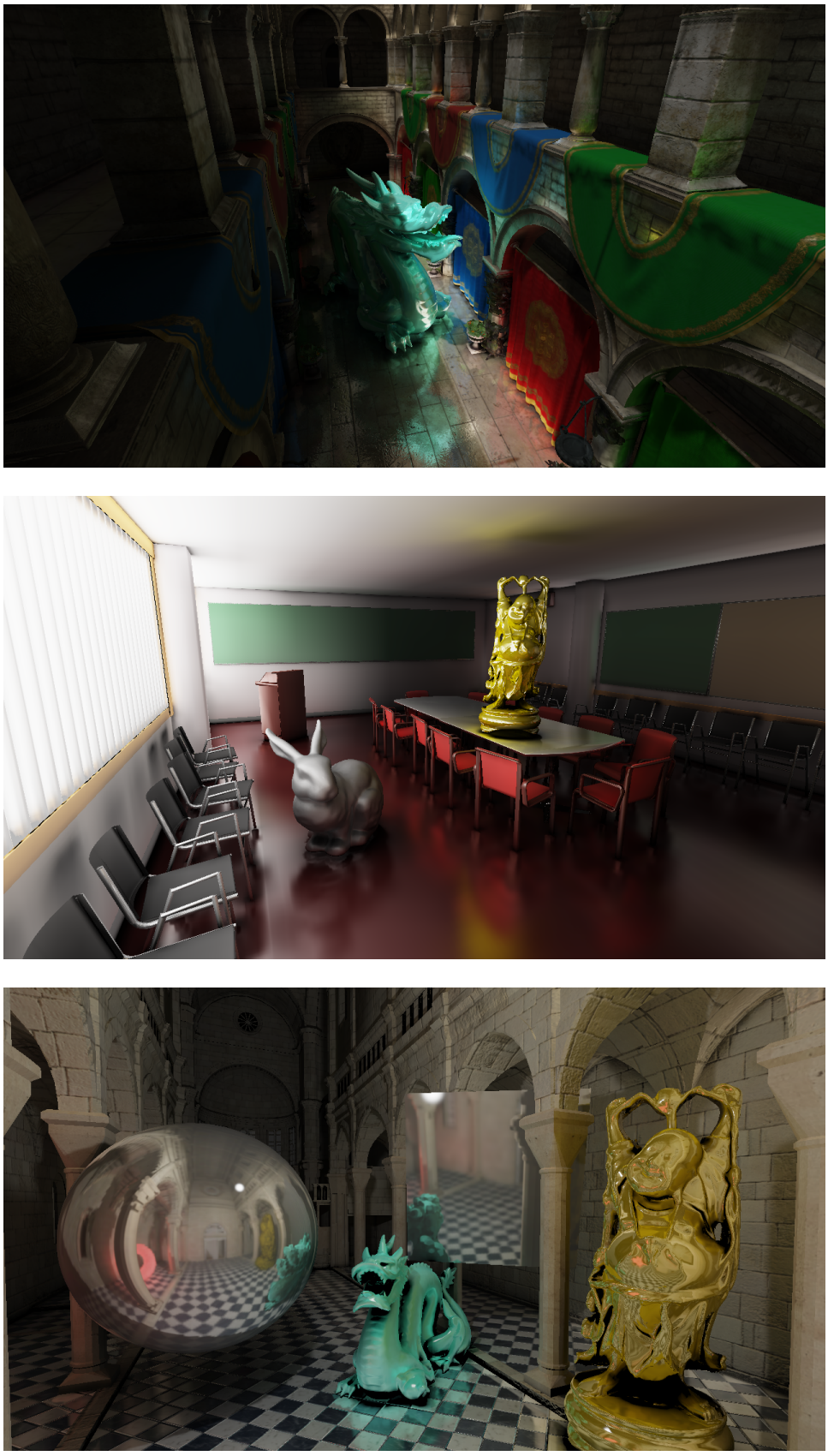


Fig. 17. Images of our approach for real time global illumination on different scenes. The first image shows diffuse reflection, second image only receives light through emissive objects and the third image demonstrates specular reflections.