Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

Interactive Rendering of Interior Scenes with Dynamic Environment Illumination

Yonghao Yue[1], Kei Iwasaki[2], Bing-Yu Chen[3], Yoshinori Dobashi[4] and Tomoyuki Nishita[1]

[1]The University of Tokyo [2]Wakayama University [3]National Taiwan University [4]Hokkaido University
[1]yonghao,nis@nis-lab.is.s.u-tokyo.ac.jp [2]iwasaki@sys.wakayama-u.ac.jp [3]robin@ntu.edu.tw [4]doba@ime.ist.hokudai.ac.jp

Abstract

A rendering system for interior scenes is proposed in this paper. The light reaches the interior scene, usually through small regions, such as windows or abat-jours, which we call portals. To provide a solution, suitable for rendering interior scenes with portals, we extend the traditional precomputed radiance transfer approaches. In our approach, a bounding sphere, which we call a shell, of the interior, centered at each portal, is created and the light transferred from the shell towards the interior through the portal is precomputed. Each shell acts as an environment light source and its intensity distribution is determined by rendering images of the scene, viewed from the center of the shell. By updating the intensity distribution of the shell at each frame, we are able to handle dynamic objects outside the shells. The material of the portals can also be modified at run time (e.g. changing from transparent glass to frosted glass). Several applications are shown, including the illumination of a cathedral, lit by skylight at different times of a day, and a car, running in a town, at interactive frame rates, with a dynamic viewpoint.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Radiosity

1. Introduction

Environment lighting [Deb98] has been recognized as a powerful lighting model, and many efficient rendering algorithms have been presented. In previous methods, the rendering of exterior scenes lit by environment lighting was the primary target, while interior scenes is also important for many applications, such as interior lighting design. In this paper, we present a rendering system for interior scenes under dynamic environment lighting. We assume that the interior scenes, such as an architecture or a car, consist of static objects, which are surrounded by a dynamic environment that may have moving objects. Possible examples of the scenes are the interior of a church (Figure 1), lit by time-varying skylight, or the interior of a car, running in a town.

In an interior scene, the environmental light penetrates into the scene, usually through some small regions, such as windows or abat-jours, which we call *portals*. Since most interior scenes have windows, it is important to take into account the light transferred through these portals. There are several distinct property differences between interior and exterior scenes, which make difficult the direct use of the previous PRT (precomputed radiance transfer) approaches.

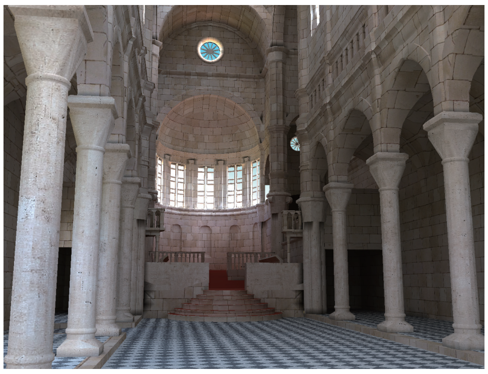


Figure 1: *Sibenik cathedral, lit by skylight issuing through portals, rendered using our method at around 16 fps. Both direct and indirect lighting components are calculated.*

Firstly, due to the portals, only a fraction of the environment light from the portals is transferred into the interior scene, which makes the traditional *spherical harmonics based PRT* approaches unsuitable for the direct lighting component. Secondly, the precomputation time would become extended. In an interior scene, the inter-reflection effects become more important, and usually several bounces of inter-reflections need to be taken into account.

In this paper, we consider the above distinctive properties and propose an efficient method for rendering interior scenes with dynamic environmental lighting. Firstly, we compute the direct and indirect lighting components in different ways. We selectively use Haar wavelets and pulse functions to encode the direct lighting component and spherical harmonics for the indirect lighting component. Secondly, our method treats portals explicitly, and regards each portal as a non-diffuse area light source for efficient precomputation and rendering.

The assumptions in our method are as follows. The interior scene is static and consists of polygonal meshes, and the radiances are computed at vertices of the objects, thus the scene needs to be finely subdivided. Our method can handle any low-frequency BRDFs (Bidirectional Reflectance Distribution Functions). Using our method, we can render the scene at interactive frame rates with manipulating the viewpoint, light sources, and the material of portals. In addition, our method can handle dynamic exterior environment.

2. Related Work

Precomputed radiance transfer (PRT) [SKS02] provides very high rendering frame rates and convincing results for exterior scenes. However, it becomes inaccurate in computing the direct lighting component of interior scenes, especially when the portal is small (see Figure 2 for detailed discussion). This inaccuracy is due to the truncation of the high-frequency components. Possible alternatives would be to use other bases [NRH03, TS06, GKMD06] to project the light transfer, at the cost of long precomputation time. While accelerating the calculation is an interesting approach, e.g. [KTHS06], it is still challenging to make the method applicable for scenes composed of a large number of polygons.

For the sake of designing BRDFs under environment light sources, Sun *et al.* [SZC*07] proposed to use precomputed transfer tensors and Cheslack-Postava *et al.* [CPWAP08] proposed to use non linear cut approximation.

Additionally, while previous PRT methods assume diffuse uniform area light sources, our technique can essentially handle non-diffuse area light sources, which are basically the portals introduced in this paper.

Direct-to-indirect light transport proposed by Hašan *et al.* achieved high quality rendering with static scenes and fixed viewpoint in real-time [HPB06]. Wang *et al.* used a spectral mesh basis to encode the direct-to-indirect transfer [WZH07]. It is possible to dynamically change the surface albedo using their method, but the frame rate drops according to the number of bounces of the inter-reflection. To incorporate large scenes, a meshless approach [LZT*08] would be a good combination with the direct-to-indirect light transport to handle diffuse surfaces.

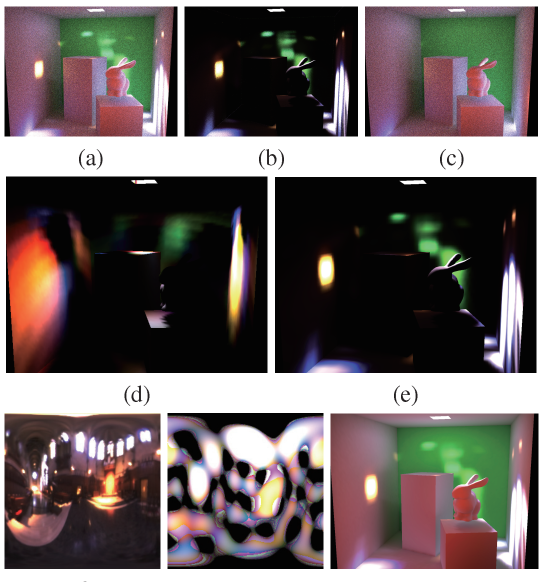


Figure 2: *A box scene, with an abat-jour on the ceiling, lit with the light probe image (f), and rendered using path tracing (a). (b) and (c) show the direct and indirect lighting components of (a), respectively. (d) and (e) show the direct lighting component, approximated using spherical harmonics (SH) and our method, respectively. We can clearly notice the large error when using SH to encode the direct lighting component. This is because the light traveling through the small abat-jour is only a small portion of the whole environmental illumination, which is only a small portion of (g). (g) shows the light probe image represented in SH up to the 10-th order. (h) shows the global illumination rendered using our method.*

Direct solution of the rendering equation [Kaj86] is another interesting research direction. Recently, there are a few solutions for interactive global illumination, without imposing any static condition or precomputation. Dachsbacher *et al.* [DSDD07] and Dong *et al.* [DKTS07] extended radiosity [CG85, NN85] A drawback of these methods is that they do not scale well regarding the number of polygons in the scene. Ritschel *et al.* [RGK*08] extended instant radiosity [Kel97] and used imperfect shadows maps. When the scene contains glossy objects or large light sources, the method needs a large number of virtual point lights for good approximation. Wang *et al.* [WWZ*09] implemented a global illumination solution based on photon mapping and final gathering fully

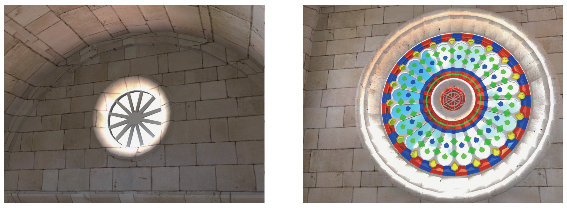


Figure 3: *Examples of portals. Left: An abat-jour in the Sibenik scene. Although the portal consists of several separated glasses, they can be treated as one portal. Right: A stained glass in the Sibenik scene. The colored glasses can be treated as several portals according to their colors. If all the materials of the glasses are fixed during rendering, we can treat all the glasses as one portal.*

on the GPU. In their applications the scene is lit by only point light sources or spot light sources.

Considering that the interior scene is static, we chose to build our system on the previous PRT framework. We compute the direct and indirect lighting components separately to overcome the weakness when just using a single previous method, with only a small performance degradation. The precomputation can be performed efficiently as shown in Section 4.

3. Lighting Computation

To handle the light transport from the exterior dynamic environment into the interior scene, we introduce the two concepts of *portals* and *shells*.

3.1. Definition

The concept of *portal* was first introduced by Teller *et al.* [TS91] to indicate the spaces through which the light transfers between rooms. In our case, a *portal* represents a region through which the light travels from the exterior into the interior. We assume that each portal is represented as a collection of triangles (usually, a triangle mesh or several triangle meshes (Figure 3)). Typical examples of a portal are a single glass of a window, a collection of several windows, or a portion of a window.

A shell is a bounding sphere, enclosing the interior scene, centered at a portal. Each shell acts as an environment light source and its intensity distribution is determined by rendering images of the exterior scene, viewed from the center of the shell.

Strictly, each portal has a one-to-one relation with a shell. On every location in a portal, the surrounding environment is assumed to be identical to that seen from the center of the portal (*i.e.*, identical to what is projected onto the corresponding shell). To ensure this assumption, if the size of a window is too large, the triangle primitives of the window are classified into several portals (see Section 6 for details).

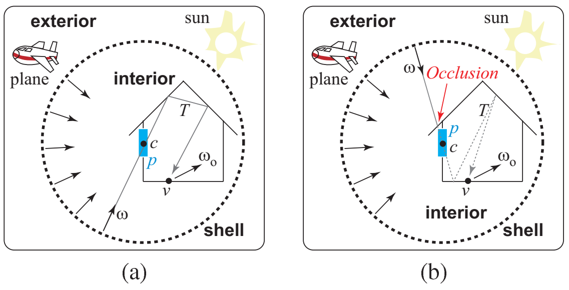


Figure 4: *(a) A shell (dotted-circle), centered at the center c of a portal p. T denotes the light transferred from the shell through c with more than zero inter-reflections before reaching a vertex v in the interior. (b) The light is occluded before reaching c.*

3.2. Mathematical Formulation

Let shell s be centered at the center c of portal p (Figure 4(a)), and let $T_v(c,\omega,\omega_o)$ be the transfer function from shell s, passing through point c from direction ω, and reaching vertex v in the interior, leaving in the viewing direction ω_o. In this paper, we define ω and ω_o in world coordinates. We encode the direct and indirect light transport (probably with several bounces of inter-reflection) in $T_v(c,\omega,\omega_o)$. We also handle the occlusion before reaching point c in $T_v(c,\omega,\omega_o)$ as shown in Figure 4(b). Then, we can write the differential radiance $dL_v(c,\omega_o)/dx$ transferred from shell s to vertex v through a small opening dx around point c in the portal as follows,

$$\frac{dL_v(c,\omega_o)}{dx} = \int_{S^2} L_s(\omega) T_v(c,\omega,\omega_o) d\omega, \tag{1}$$

where $L_s(\omega)$ is the radiance leaving the shell s, and S^2 indicates directions in sphere and in world coordinates. Here, we naturally extended the definition of $T_v(c,\omega,\omega_o)$ to include ω in any direction, by defining the light emitted from the other side of the portal to 0. Since all the points on portal p share the same shell s, we can write the light $L_v(\omega_o)$ transferred from shell s to vertex v through portal p as follows,

$$L_v(\omega_o) = \int_{S^2}\int_A L_s(\omega) T_v(x,\omega,\omega_o) dx d\omega, \tag{2}$$

where A is the region of the portal. Since the interior is static, the transfer function for each portal p to vertex v can be precomputed as $T_{p,v}(\omega,\omega_o) = \int_A T_v(x,\omega,\omega_o)dx$. Then,

$$L_v(\omega_o) = \int_{S^2} L_s(\omega) T_{p,v}(\omega,\omega_o) d\omega. \tag{3}$$

3.3. Discussion

Advantages of using Eq.(3) are as follows. Firstly, since $L_s(\omega)$ is dependent on each portal, we can apply some effects to it (*i.e.*, by replacing $L_s(\omega)$ with $L'_s(\omega) = f(L_s(\omega))$) to simulate the light transfer through portals with different

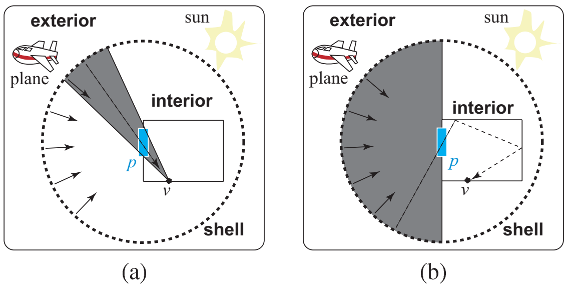


Figure 5: *The portion of the light from the shell contributing (a) directly into the scene, (b) indirectly to the scene.*

properties. For example, multiplying $L_s(\omega)$ with a color coefficient would result in different colors of the glasses in the portal, and applying a Gaussian filter to $L_s(\omega)$ would give an approximate effect as if the portal is made of frosted glass. Secondly, this formulation gives us a trivial treatment of how to efficiently sample the whole transfer function by explicitly handling the portals. From the formulation, we can regard a portal as a non-diffuse area light source. A distinct aspect is that we also need to check the visibility in the backward direction (Figure 4(b)), which we call *backward occlusion*.

If we have many different windows with different properties, we evaluate each component of the radiance corresponding to each portal. On the other hand, if the materials of the windows are fixed, and the environment light comes from infinitely far away, then one shell and one portal, which is the collection of all the windows, are sufficient. In this case, we encode the transport property of the portals into the transport function $T_{p,v}$, and our method is analogous to a traditional PRT framework. Thus, our method can be considered as an extension of the PRT framework.

3.4. Basis Representation

We represent both $L_s(\omega)$ and $T_{p,v}(\omega,\omega_o)$ in orthonormal bases for efficient calculation. From Figure 5, we can see a distinct property of the transfer function, compared to the case in exterior scenes; the portion of the light contributing directly into the scene is far smaller than that contributing indirectly due to the existence of the portals. This property will guide us in choosing the bases for the direct and indirect lighting components.

For the indirect lighting component, we chose to use spherical harmonics rather than wavelets for the following reasons. Firstly, it is sufficient to use only low-order spherical harmonics (see Section 4.2), therefore, is convenient for practical use. Secondly, to use wavelets to encode the indirect lighting component is itself a very challenging problem. Also, if we limit the number of wavelet coefficients to be comparable to that for spherical harmonics, which means approximately 2 by 2 coefficients for a single face of the cubemap, then it seems difficult to obtain accurate approximation. If we increase the resolution, we need much more computational time.

For the direct lighting component, it can be efficiently handled by directly sampling the transfer function on a cubemap, and keeping only a small number of non-occluded samples. To encode the direct lighting component, we use one of the following two bases, depending on each vertex for better compression ratio. The first type constitutes the set of pulse functions on each cell of the cubemap, so that the bases form a trivial set of orthogonal bases. The second type constitutes Haar wavelet bases. Intuitively, as long as the portal is not too wide, the former type outperforms the wavelet type. If the portal is small and the resolution of the cubemap is low, using only the wavelet bases will need 3 to 4 times more storage than using only the pulse functions as the bases. On the other hand, if the portal is large and the resolution of the cubemap is high, using the wavelet bases will reduce the storage to about 40%.

We briefly review the lighting computation using orthonormal bases in Appendix for completeness of the paper. Readers can find detailed description on spherical harmonics in [SKS02] and on the Haar wavelets in [NRH03].

4. Precomputation

In this section, we first describe the algorithms to precompute indirect and direct lighting components of the transfer function, then an evaluation on using low-order spherical harmonics to approximate the indirect illumination.

4.1. Precomputation of Transfer Function

4.1.1. Indirect Lighting Component

To compute the indirect lighting component, we utilize photon mapping and final gathering for the calculation of the light transport response for each basis of the spherical harmonics. That is, each photon carries not the actual energy related to a particular environment light source, but the light transport response for a certain basis function. We gather the light transport response in the final gathering step. If a particular shell is given during rendering, the radiance at each vertex is obtained by a linear combination of the precomputed response.

We summarize the precomputation steps as follows. The steps are repeated for each portal.

1. For each basis function
    a. trace photons from the portal
    b. estimate radiance at each vertex
2. For each vertex
    a. perform final gathering for each basis function

Next, we describe the implementation details. For each

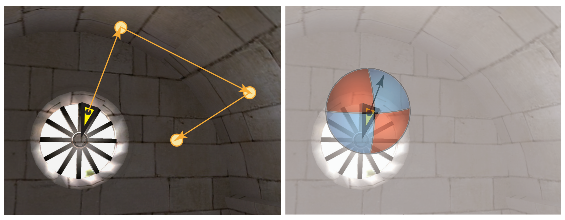


Figure 6: *Left: we choose a triangle (yellow one) of the portal and select a location (dark gray circle) on the triangle, then emit a photon in random direction and trace the photon. Orange circles indicates the locations where the light transport response is stored into the photon map. Right: the initial energy of the photon is assigned to be the value of the basis function, evaluated in the direction. Blue and red regions show the negative and positive values, respectively, of a particular basis function.*

basis function, we create a photon map. Firstly, we randomly select a triangle, with the probability being proportional to its area, belonging to the portal. Secondly, we randomly select a location on the triangle and emit a photon in a randomly selected direction (Figure 6 Left). To handle the backward occlusion, we trace a ray in the backward direction. If the ray cannot reach the shell, the photon is discarded. Thirdly, we assign the energy of the photon to be the value of the basis function at the emitting direction (Figure 6 Right). Lastly, we trace the photon path, record the light transport response at each hit point, and multiply the BRDF at the surface to the light transport response.

After all the photons are traced, we estimate the radiance at each vertex and store the radiances, as in [Chr99]. Then, we perform final gathering for each vertex using the stored radiances. If the vertex is on a diffuse surface, we obtain the irradiance through the final gathering. Otherwise, we compute the outgoing radiance distribution for the vertex by evaluating the BRDF. Then, we represent outgoing radiance distribution with spherical harmonics.

We would like to emphasize three aspects in which our method is efficient. Firstly, the process for the precomputation can be easily parallelized. For the process to render photon maps and estimate the radiance at each vertex, we can process several photon maps in parallel, and in the final gathering step, we can process several vertices in parallel. Secondly, the photons are traced from the portals rather than from the entire environment. This approach can be regarded as an explicit importance sampling of the transport function, which would benefit greatly since most light paths would be occluded by the walls. Lastly, the light transport is calculated for each order of the spherical harmonics (like [DKNY95]), and can be performed in a progressive manner. That is, if we ascertain that the accuracy of the approximation is not sufficient after computing the light transport for spherical harmonics up to the n-th order, we just continue the computation for the $(n+1)$-th order, rather than re-computing the light transport from scratch.

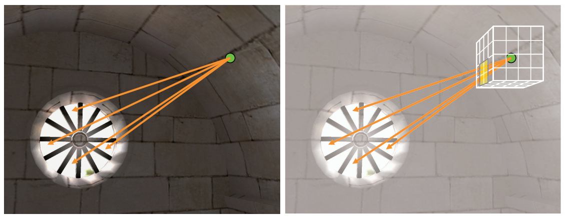


Figure 7: *Left: we choose a triangle of the portal and select a location on the triangle, then check the occlusion between the location and the vertex. Right: the directions of the rays are encoded as the IDs of the cells on the cubemap faces, and the contribution is accumulated for each cell.*

4.1.2. Direct Lighting Component

We summarize the precomputation steps as follows. The steps are repeated for each portal and each vertex.

1. Generate samples on portal
2. Check the visibility between the samples and the vertex
3. Multiply the visibility with the BRDF and accumulate the energy into the cubemap
4. Encode the energy distribution using
    a. pulse functions
    b. Haar wavelets
5. Quantize both sets of coefficients into 8 bits
6. Choose the bases resulting in a better compression ratio

Next, we describe the implementation details. For each direction at each vertex, we directly sample the contribution. For a vertex on a diffuse surface, we discretize the ranges of directions ω into the directions on the cubemap with the resolution required, *e.g.* 32×32 for each face. The discretized directions at each vertex are represented in world coordinates to make the computation simple. Since the solid angle of viewing a portal from a vertex is usually very small, we explicitly select a point on the portal and then check the visibility between the point and the vertex (Figure 7 Left) as well as the backward occlusion. If the ray is established, we encode the direction into the cubemap (Figure 7 Right), and sample the BRDF to obtain the irradiance. The irradiance is finally accumulated for each direction. If a vertex is on a non-diffuse surface, we also discretize the range of directions ω_o into the directions on the cubemap, and calculate the radiance between the direction of the ray and each discretized direction of ω_o.

After obtaining the irradiance and radiance, we encode them into our bases. For a vertex on a diffuse surface, we have obtained a distribution of the irradiances on the cubemap. We try two choices and select the better one. The first choice is just to quantize the distribution into 8 bits, and the second one is to perform a Haar wavelet decomposition for each cubemap face before the quantization. For the quantization, we calculate the maximum contribution among all the directions for the vertex in our implementation. Usually, there are only a small number of directions on the cubemap that are not entirely occluded, thus, the wavelet representation would perform less well. On the other hand, if a vertex is near a portal, wavelet representation will perform better. Therefore, we choose the bases depending on each vertex, according to the compression ratio. For a vertex on a non-diffuse surface, the radiance distribution on ω is encoded for each discretized direction of ω_o.

4.2. Evaluation on Using Low-order SH

Using low-order spherical harmonics to approximate the indirect illumination will lead to ignore the high frequency components of the indirect portion of the transfer function. This will result in high frequency components of the source lighting begin ignored as well. We, therefore, evaluated if this assumption is valid by accounting for the maximum possible contribution carried by each order of the spherical harmonics. To briefly summarize the evaluation result, in general, the maximum energy carried by a certain order of the bases would decrease as the order increases, which results in that the 4-6th order approximation shall cover over 90% of the whole energy. Thus, low-order approximation is quite good for general scenes.

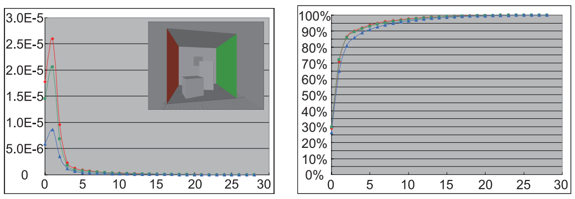
Starting by considering an arbitrary environment lighting $L(\omega)$, its spherical harmonics expansion is then given by $L(\omega) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} c_{l,m} y_{l,m}(\omega)$, where $c_{l,m}$ is the coefficient for the m-th basis $y_{l,m}$ in the l-th order. In the same way, the transfer function for diffuse vertex v, due to portal p can be expanded using another coefficient $t_{l,m}^{p,v}$. Then, radiance L_v at vertex v is calculated as $L_v = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} c_{l,m} t_{l,m}^{p,v}$. We define the squared average of the radiances of all the vertices due to the l-th order as

$$\overline{L_l}^2 = \frac{1}{N_v}\sum_{v=1}^{N_v}\Big(\sum_{m=-l}^{l} c_{l,m} t_{l,m}^{p,v}\Big)^2, \qquad (4)$$

where N_v is the number of vertices. To examine the maximum energy the l-th order coefficients will carry, we solve a maximization problem as

$$max\ \overline{L_l}^2 \ \ s.t.\ \sum_{m=-l}^{l}(c_{l,m})^2 = 1. \qquad (5)$$

By plotting the maximum of $\overline{L_l}^2$, we can see how the energy carried by certain order coefficients decreases, as shown in Figure 8 (a). We can also calculate the summation of the energy up to a certain order, which is shown in Figure 8 (b). This means that given an environment light source with its energy equidistributed in all the orders, the approximated energy of using coefficients up to a certain order can be obtained from the figure. From this illustration, we can also say



(a) (b)

Figure 8: *The Cornell box scene with a window (shown in the upper right of (a)) is used as the test case. N_v is 24K. Horizontal axes in (a) and (b) show the orders of the spherical harmonics. (a): energy contribution $\overline{L_l}^2$ of the l-th order bases are plotted. (b): energy ratio when using the approximation up to the l-th order bases are plotted. Red, blue, and green lines show the red, blue, and green contributions, respectively. Since the scene has only red, green and gray walls, the red and green contributions are dominant.*

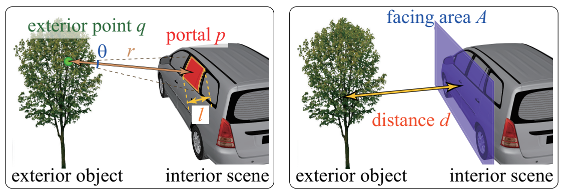
that if the accuracy in this squared average sense is needed to be a certain amount, *e.g.* 90%, it is sufficient to approximate the energy using the coefficients up to the 5th order. For the evaluation of a particular environment light source, we can weight the summation by the energy distribution of the source.

The order of the spherical harmonics required depends on the size of the portal. If the size of the portal becomes 1/16 of that used in the test case in Figure 8, then we need the coefficients up to the 8th order to obtain total energy being larger than 90%. Generally, as the size becomes smaller, the required order gets higher.

5. Rendering

To render the scene, we first capture the environmental light distribution for each shell, and then calculate the radiance for each vertex. We determine the environmental light distribution for a shell by rendering images of the scene, viewed from the center of the shell. The environmental light distribution is then modified according to the change in the material of the portal, if necessary. Next, we represent the environmental light distribution in spherical harmonics bases, cubemap representation and wavelet bases. The rotation of the outside environment is handled intrinsically, since we directly capture the environmental light distribution.

To calculate the indirect lighting component, the spherical harmonic representation of the environmental light distribution is multiplied by that representation of precomputed $T_{p,v}$ at each vertex using Eq.(6) or (7). The direct lighting component is calculated analogously. Finally, the colors are modulated with textures at the surfaces. Currently, we compute the direct and indirect lighting components using CPU and GPU, respectively. We are planning to implement all the computations using GPU in the future.



(a) (b)

Figure 9: *Illustration of the criteria for the conditions to handle dynamic exterior objects.*

6. Dynamic Environment

In this section, we discuss two conditions for handling dynamic exterior objects.

First, our method assumes that all the points in a portal share the same shell. We have to decide the valid distance between the exterior object and the portal which does not break the assumption. To do this, we check the directional error due to this assumption. Let q be an exterior point, l be the length between the maximum distance among any two points in portal p, and r be the distance between point q and the center of the portal (Figure 9(a)). Then among all the points in the portal, the angles of viewing point q will vary at most θ, which can be approximated by $\theta \approx l/r$. Since we assume point q is viewed from the center of the portal when we project the dynamic environment onto the shell, we can say that the directional error is approximately at most $\theta/2$. Thus, by setting a threshold θ_M, if $l/(2r) < \theta_M$ holds, then we can say that the assumption is valid. To give an example, if we set θ_M to be 5°, and an exterior object will not be closer than 3m, then the size of the portal can be as large as 50cm. Thus, in a car running scene, we can treat each window as a separate portal.

Second, in order to ensure that an exterior object will receive little inter-reflected light from the interior scene, the solid angle of the interior scene viewed from the exterior object must be sufficiently small, since the solid angle is a determinant of the quantity of the reflected light. Let A be the area of the interior scene facing towards the exterior object, and d be the distance between the exterior object and the interior scene (Figure 9(b)). Then, by setting a threshold ratio R_ω, if $A/(4\pi d^2) < R_\omega$ holds, then we assume the inter-reflected light is sufficiently small. To give an example, A can be assumed at most $2m^2$ for cars like sedans, and if we set R_ω to be 0.02, then distance d must be longer than 2.8m. Thus, in a car scene, the reflected light from the interior to exterior objects is generally very small and can be ignored.

From the above two conditions, we can handle a certain range of dynamic exterior objects. We can also see that, when the light comes from only infinitely far away and the materials of all the windows are same, then all the windows can be classified as one portal from the first condition.

7. Results

First, we show some simple experimental results using the box scene. We then show the applications of our method using two scenes: the Sibenik cathedral and a car. The precomputation and rendering are all conducted using a PC with an Intel Core 2 Extreme QX9650 CPU, 2GB memory, and an NVIDIA GeForce 9800 GTX+ GPU.

We show four examples of the box scene for different settings of the portal in the top row of Figure 10. The environment light source used to render these results is a sunny sky, shining through a window on the wall. The exposures are set to be the same for all these results to show the differences in the luminance due to the sizes of the portals. Figures 10(e) and (f) show the direct and indirect lighting components of Figure 10(g), respectively. We conducted a comparison between the rendered results of the scene using our method and a pure Monte Carlo method (Figures 10(g) and (h)). Figure 10(i) shows the difference between (g) and (h). The average difference between Figures 10(g) and (h) is 1.5%. From these results and the comparison, we can say that the illumination effects are accurately calculated using our method. We also conducted an experiment on handling large number of polygons. To do this, we subdivided the box scene and obtained 1.1M polygons in the scene. The subdivided box scene can be rendered at 27 fps. The storage for the precomputed data is approximately 300 MB.

Next, we show some rendered results for glossy surfaces by replacing the diffuse bunny with a glossy one in Figures 11(a) and (b). Figure 11(c) shows a reference image rendered using a pure Monte Carlo method, and Figure 11(d) shows the difference between (b) and (c). The average difference is 2.1%. The difference arises mainly in the direct glossy component since our method calculates radiances per-vertex while the Monte Carlo method per-pixel. The Phong exponent of the glossy bunny is 10. We can see that the shadows on the wall due to the bunny and the nature of the glossy surface are accurately rendered.

The Sibenik cathedral scene contains many portals as shown in Figure 12(a). Figure 12(b) shows a rendered result of the scene at sunset. Enlarged views of Figure 12(b) are shown in (c) and (d). The direct lighting issuing through the portals, as well as the shadows, are well-captured. Figure 12(e) shows a top view of (a), and (f) shows a result lit by daylight. By using our method, good visual effects due to the environment light passing through the portals are obtained.

Figure 13 shows an example of a car running in a town. The car is considered as the interior scene, and the town, which consists of 680K triangles, is a dynamic exterior environment in this case. We can see that the illumination distribution changes as the car moves.

We set the resolution of the cube face to precompute the direct lighting component to be 32 × 32 for all the scenes. We used spherical harmonics up to the 4th order to represent

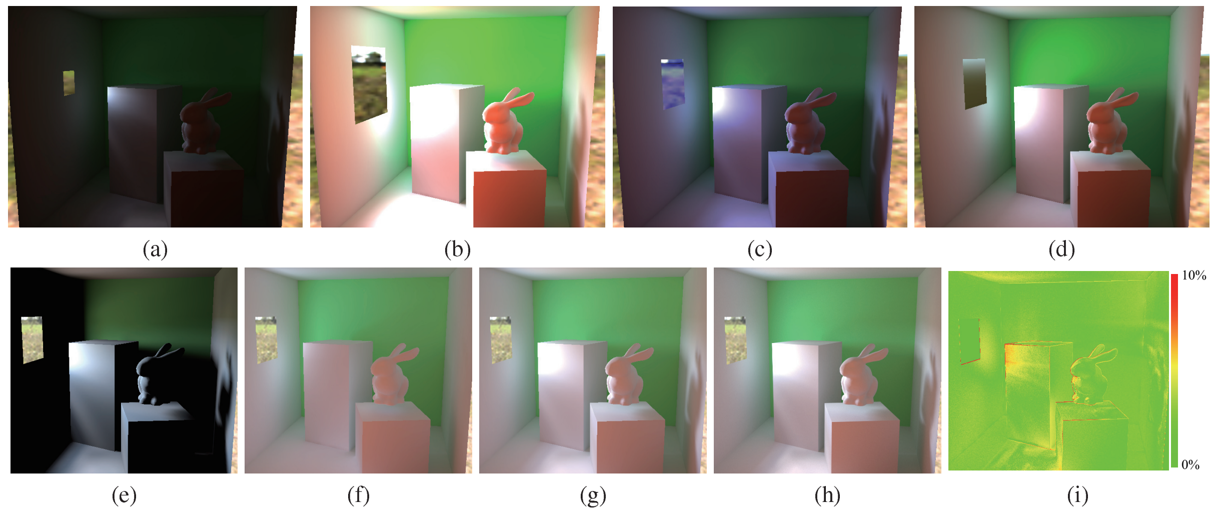


Figure 10: *Top row: the box scene with different portals. (a): small, (b): large, (c): blueish, and (d): made of frosted glass. (e) and (f): the direct and indirect lighting components of (g). (g): our result, (h): reference image using a pure Monte Carlo method, and (i): showing the difference between (g) and (h) using false colors.*

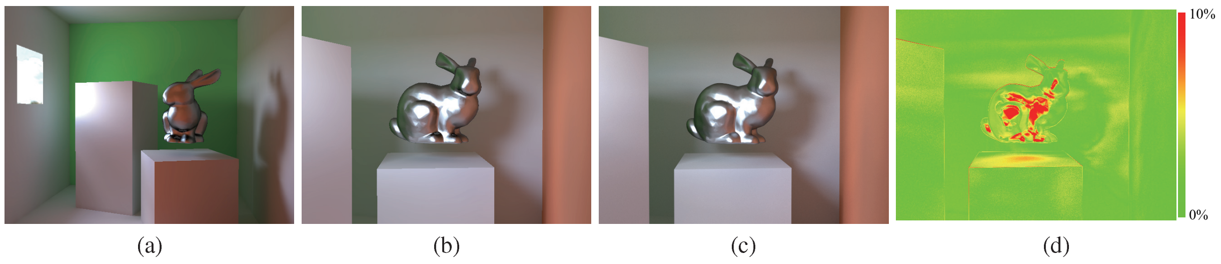


Figure 11: *(a): a rendered result of the box scene, (b): the color of the glossy surface changes when viewed from another direction, (c): reference image using a pure Monte Carlo method, and (d): difference between (b) and (c) in false colors.*

the indirect lighting component. We tested 4,096 rays from each vertex to estimate the direct lighting component for the box scene and 16,384 rays for the Sibenik and car scenes. We traced 2M photons for each basis function to precompute the indirect lighting component, and 4,096 rays at each vertex were used for final gathering. We considered interreflected light up to 9 times bounces. The statistics for the precomputation and rendering are given in Table 1. We can see that the size of the precomputed data is acceptable for modern graphics hardware.

Although there are no interior light sources in the examples shown here, our method can easily handle them by using precomputed local radiance transfer [KAMJ05].

8. Limitations and Discussions

There are mainly two limitations in our method. Firstly, since the direct lighting component is captured on a per-vertex-basis, some jaggy artifacts occurred when the discontinuity of the mesh does not adapt to the discontinuity in the radiance [LTG92]. This problem should be resolved by applying some sophisticated geometry tessellation algorithm. Also, we need to finely subdivide glossy objects to accurately capture the direct glossy component.

Secondly, we need to provide the portals to render the scene. Usually, an interior scene, *e.g.* an architecture scene or a vehicle scene, contains glass polygons, which can be directly used as polygons for the portals. Otherwise, we have to provide the polygons for the portals explicitly.

9. Conclusions and Future Work

In this paper, we have proposed a rendering system for interior scenes, lit by dynamic environment illumination, issuing through portals. By representing and computing the direct and indirect lighting components in different ways, the lighting due to the environment illumination can be efficiently handled. An implementation of fast precomputation is also presented. Several examples to demonstrate the importance of the environment lighting and the efficiency of our method for rendering interior scenes are also displayed. The glossy

Table 1: *Statistics of the scenes. 'Box' indicates the scene shown in Figure 11. $T(dir/ind)$ denotes the times for precomputation of the direct and indirect lighting components in minutes. $M(dir/ind)$ denotes the required storage sizes for the direct and indirect lighting components in MB.*

Scene	Box	Sibenik	Car
#Triangles	69,410	185,676	329,070
#Vertices	44,763	188,394	182,470
$T(dir/ind)$	12 / 11	56 / 14	60 / 23
FPS	70	16	12
$M(dir/ind)$	90.8 / 112.4	87.0 / 35.3	70.6 / 34.2

surfaces and complex scenes are also handled well. Using our method, we can render at interactive frame rates, manipulate the viewpoint, light sources, and the material of portals, and we can also handle dynamic exterior environment.

For the future work, we are planning to explore a way of subdividing the geometry of scenes, making the number of resulting triangles as small as possible, as well as capturing the illumination accurately. We are also willing to further improve the performance for precomputation and for computing the direct lighting component during rendering by using GPUs. We believe that GPU based tracing and photon mapping [WWZ*09] are useful to accelerate the precomputation. Other challenging research directions would be to consider a bidirectional light transfer (not only from the exterior to the interior but also from the interior to the exterior) and to handle moving portals.

Acknowledgments

We gratefully acknowledge helpful comments and thoughtful suggestions from the anonymous reviewers. This work was supported by Grant-in-Aid for JSPS Fellows (20 · 7968). We would also like to thank WORKS ZEBRA Co.,Ltd for providing the car model.

References

[CG85] COHEN M. F., GREENBERG D. P.: The hemi-cube: a radiosity solution for complex environments. In *SIGGRAPH '85* (1985), pp. 31–40.

[Chr99] CHRISTENSEN P. H.: Faster photon map global illumination. *J. Graph. Tools 4*, 3 (1999), 1–10.

[CPWAP08] CHESLACK-POSTAVA E., WANG R., AKERLUND O., PELLACINI F.: Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph. 27*, 5 (2008), 22.

[Deb98] DEBEVEC P.: Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98* (1998), pp. 189–198.

[DKNY95] DOBASHI Y., KANEDA K., NAKATANI H., YAMASHITA H.: A quick rendering method using basis functions for interactive lighting design. *Computer Graphics Forum 14*, 3 (1995), 229–240.

[DKTS07] DONG Z., KAUTZ J., THEOBALT C., SEIDEL H.-P.: Interactive global illumination using implicit visibility. In *Pacific Graphics '07* (2007), pp. 77–86.

[DSDD07] DACHSBACHER C., STAMMINGER M., DRETTAKIS G., DURAND F.: Implicit visibility and antiradiance for interactive global illumination. *ACM Trans. Graph. 26*, 3 (2007), 61.

[GKMD06] GREEN P., KAUTZ J., MATUSIK W., DURAND F.: View-dependent precomputed light transport using nonlinear gaussian function approximations. In *I3D '06* (2006), pp. 7–14.

[HPB06] HAŠAN M., PELLACINI F., BALA K.: Direct-to-indirect transfer for cinematic relighting. *ACM Trans. Graph. 25*, 3 (2006), 1089–1097.

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86* (1986), pp. 143–150.

[KAMJ05] KRISTENSEN A. W., AKENINE-MOLLER T., JENSEN H. W.: Precomputed local radiance transfer for real-time lighting design. *ACM Trans. Graph. 24*, 3 (2005), 1208–1215.

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97* (1997), pp. 49–56.

[KTHS06] KONTKANEN J., TURQUIN E., HOLZSCHUCH N., SILLION F. X.: Wavelet radiance transport for interactive indirect lighting. In *EGSR '06* (2006), pp. 161–171.

[LTG92] LISCHINSKI D., TAMPIERI F., GREENBERG D. P.: Discontinuity Meshing for Accurate Radiosity. *IEEE Comput. Graph. Appl. 12*, 6 (1992), 25–39.

[LZT*08] LEHTINEN J., ZWICKER M., TURQUIN E., KONTKANEN J., DURAND F., SILLION F. X., AILA T.: A meshless hierarchical representation for light transport. *ACM Trans. Graph. 27*, 3 (2008), 37.

[NN85] NISHITA T., NAKAMAE E.: Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *SIGGRAPH '85* (1985), pp. 23–30.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22*, 3 (2003), 376–381.

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. 27*, 5 (2008), 129.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02* (2002), pp. 527–536.

[SZC*07] SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic brdfs. *ACM Trans. Graph. 26*, 3 (2007), 27.

[TS91] TELLER S. J., SÉQUIN C. H.: Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Comput. Graph. 25*, 4 (1991), 61–70.

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph. 25*, 3 (2006), 967–976.

[WWZ*09] WANG R., WANG R., ZHOU K., PAN M., BAO H.: An efficient gpu-based approach for interactive global illumination. *to appear in SIGGRAPH 2009* (2009).

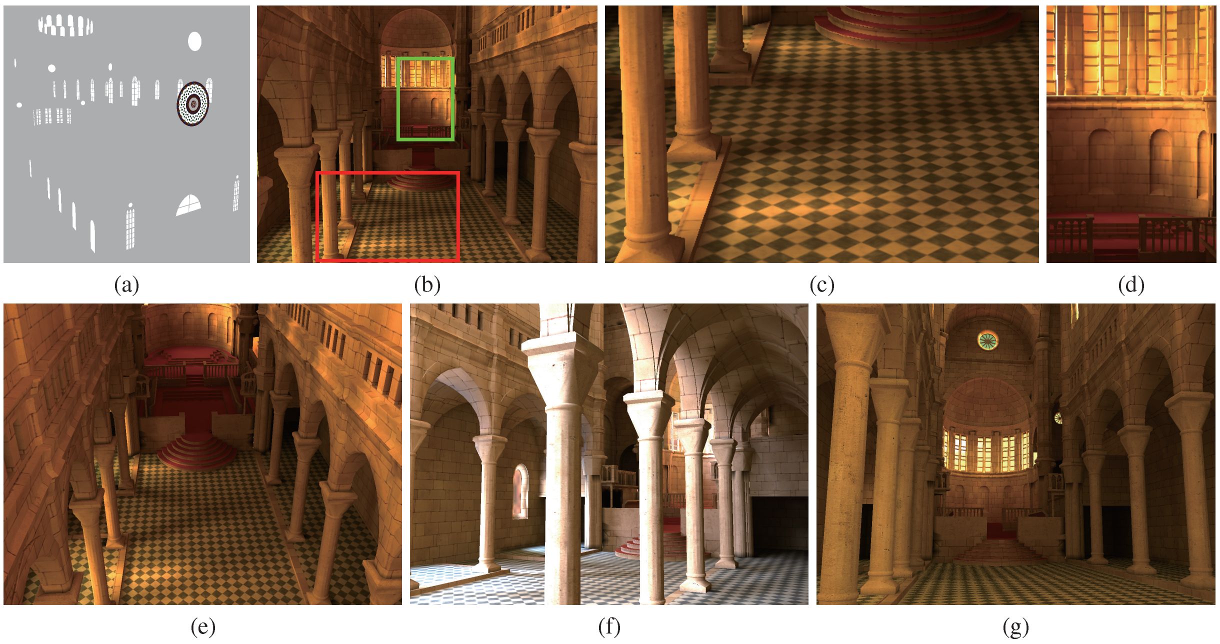


Figure 12: *(a): the portals in the Sibenik scene (portals with transparent glasses are shown by white color, and colored glasses shown by the colors), (b): at sunset. (c) and (d): close-ups of (b). (e) and (f): the Sibenik scene at sunset and lit by day light. (g): Figure 1 at sunset.*

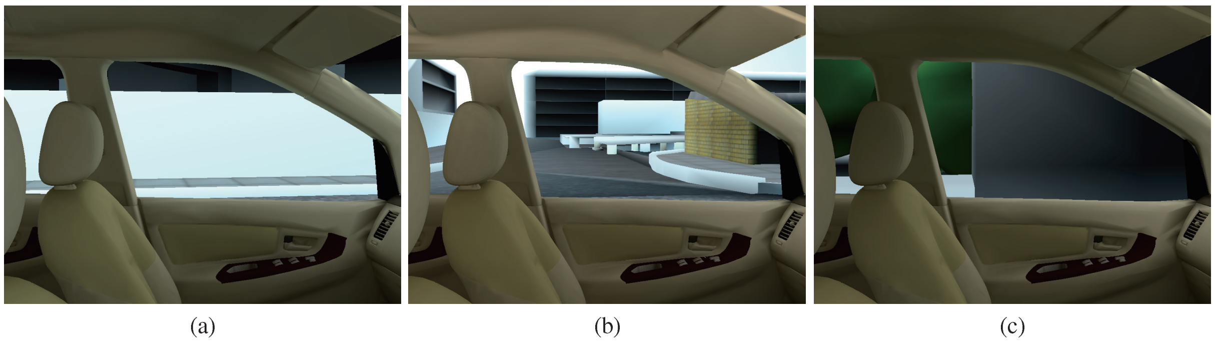


Figure 13: *The results of a car running in a town.*

[WZH07] WANG R., ZHU J., HUMPHREYS G.: Precomputed Radiance Transfer for Real-time Indirect Lighting using a Spectral Mesh Basis. In *EGSR '07* (2007), pp. 13–21.

Appendix : Lighting computation using orthonormal bases

First, the light distribution of the shell can be represented as, $L_s(\omega) \approx \sum_{i=0}^{k-1} l_i \Psi_i(\omega)$, where, k is the number of the bases, l_i is the coefficient of i-th basis function $\Psi_i(\omega)$. Next, the transfer function can be represented as follows. For non-diffuse vertices, $T_{p,v}^{non-diff}(\omega, \omega_o) \approx \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} t_{ij} \Psi_i(\omega) \Psi_j(\omega_o)$, where, t_{ij} is the coefficient of the combination of i-th basis function $\Psi_i(\omega)$ and j-th basis function $\Psi_j(\omega_o)$, and can be represented as a matrix. For diffuse vertices, $T_{p,v}^{diff}(\omega, \omega_o) \approx \sum_{i=0}^{k-1} t_i \Psi_i(\omega)$, where t_i is the coefficient of i-th basis function $\Psi_i(\omega)$, and can be represented as a vector. Then, Eq.(3) is approximated by

$$L_v^{non-diff}(\omega_o) \approx \sum_{j=0}^{k-1} \Psi_j(\omega_o) \sum_{i=0}^{k-1} t_{ij} l_i, \tag{6}$$

for non-diffuse vertices, and

$$L_v^{diff}(\omega_o) \approx \sum_{i=0}^{k-1} t_i l_i, \tag{7}$$

for diffuse vertices. The computation of Eq.(6) is a matrix-vector multiplication, followed by an evaluation of the basis function in the direction ω_o, and the computation of Eq.(7) is an inner product of the coefficient vectors.