Multimed Tools Appl (2018) 77:13571–13595
DOI 10.1007/s11042-017-4976-3

Efficient inverse transform methods for VPL selection in global illumination

Djihane Babahenini[1] · Adrien Gruson[2] · Mohamed Chaouki Babahenini[1] · Kadi Bouatouch[3]

Received: 21 November 2016 / Revised: 3 May 2017 / Accepted: 21 June 2017 /
Published online: 4 July 2017

Abstract In computer graphics, designing efficient Global Illumination methods is a hot research topic. These methods consist in computing the light distribution inside a 3D scene. There exist several global illumination-based rendering methods, but one popular approach is based on Virtual Point Light (VPL). It is a two-step approach. First, the algorithm generates VPLs that act as secondary light sources (indirect illumination). Second, the radiance of a pixel is computed by summing the contributions of a small set of VPLs (rather than all the VPLs) selected randomly. The most active issues rely on how to select a small set of VPLs that contribute more to the final image. In this paper, we propose two new VPL selection methods using the inverse transform method. To provide realistic images, we propose a Multiple Importance Sampling technique combining an inverse transform method with a gathering approach. The obtained results demonstrate the effectiveness of our methods in terms of image quality and rendering time.

✉ Djihane Babahenini
babaheninidjihene@gmail.com

Adrien Gruson
adrien.gruson@gmail.com

Mohamed Chaouki Babahenini
chaouki.babahenini@gmail.com

Kadi Bouatouch
kadi.bouatouch@irisa.fr

[1] Department of Computer Science, LESIA, University of Mohamed Khider, B.P 145, 07000 Biskra, Algeria

[2] Department of Creative Informatics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

[3] IRISA, University of Rennes 1, 35000 Rennes, France

Keywords Visibility · Indirect lighting · Voxelization · Inverse transform methods · Virtual point light

1 Introduction

Lighting simulation and the search for a solution to the problem of global illumination still is a very active research in computer graphics. Global illumination aims to simulate different lighting effects in a $3D$ scene. Several approaches exist, there are based on tracing rays or photons, such as: bidirectional path tracing [23], gathering, Metropolis-Hasting [16, 26], photon tracing [21], and methods based on Virtual Point Light (VPL) [22, 31, 32]. Many methods have been proposed in the literature to compute indirect lighting, some of them make use of VPLs. The use of VPLs can be an efficient way to compute global illumination. The VPLs are computed as follows. One classical camera is placed at each point light source. Rendering from this camera allows to compute a GBuffer containing for each pixel of this camera: the 3D position of the point visible to the point light source through the pixel, its normal, flux and color. This GBuffer is called RSM (Reflective Shadow Map). Note that a classical Shadow Map contains only the z coordinate (depth) of the visible point through a pixel, and an RSM is an extension of a Shadow Map. Each visible point stored in an RSM is called VPL. Each VPL acts as a secondary light source that could contribute to any point in the scene (indirect illumination).

Previous studies have showed that the critical step in global illumination computation is to determine visibility. This latter can be computed using algorithms based on Shadow Maps [31, 32]. The authors of these latter papers have reported a good approximation of visibility when using VPLs, but it is very expensive (in terms of memory storage) to associate a Shadow Map with each VPL. Once the VPLs have been computed, for efficiency purpose, the radiance of a pixel of the scene camera is computed by summing the contributions (to the point visible through the pixel of the scene camera) of a small set of VPLs (rather than all the VPLs) selected randomly using an inverse transform method (IT) requiring the computation of a cumulative distribution function (CDF). The way the CDF is computed is crucial for the quality of the rendered image.

Our proposed methods can be used in several application domains, such as Virtual Reality and Augmented Reality, to improve realism. We could also apply our methods in Multimedia and Video games applications to minimize the rendering time.

In this paper we are interested in global illumination methods based on VPLs. The VPLs are constructed by placing two PRSM (Paraboloid Reflective Shadow Map), each having a field of view of 180 degrees around a point light source. This way a visibility of 360 degrees is assigned to a point light source. Note that a classical RSM has a field of view of only 90 degrees. From now on, the set of two PRSMs is called DPRSM (Dual Paraboloid Reflective Shadow Map).

We propose two methods for efficiently computing a CDF (used to select randomly a small set of VPLs contributing to the radiance of a point visible from the viewpoint) as well as an MIS (Multiple Importance Sampling) method combining an IT method (Inverse Transform) with a gathering approach aiming at improving the quality of the rendered image. Visibility is computed using a voxel-based approach. We consider only single bounce indirect lighting, and diffuse objects.

Our main contributions are:

- use of Dual Paraboloid Reflective Shadow Maps (DPRSM): when randomly selecting a small set of VPLS, each of the two paraboloid reflective shadow maps (PRSM) of this DPRSM is randomly selected at a time according to a Russian roulette [2];
- novel methods for computing a CDF;
- an MIS (Multiple Importance Sampling) method combining an inverse transform method (for computing CDF) with a gathering approach.

The remaining of this paper is organized as follows. Related work is presented in Section 2. Then in Section 3 we present an overview of our methods. In Section 4 we show how to select one PRSM using Russian roulette. Our inverse transform methods (local CDF and gathering-based global CDF) aiming at selecting the more contributive VPLs are presented in Section 5. We detail in Section 6 our proposed combined technique based on the MIS principle. Some experimental results are provided in Section 7. Finally, conclusion and future work are given in Section 8.

2 Related work

State-of-the-art solutions [33] to algorithms, that are used for realistic image synthesis, rely on path tracing, photon mapping, and radiosity methods. Each of them can perform efficiency in terms of time rendering or image photo-realism. We will focus our relative work on VPL [22] and path tracing on the GPU. For the other techniques, the reader can refer to Ritschel's et al. state of the art [33].

Generating VPL In the literature, there are several methods to approximate global illumination using Monte Carlo estimator and Importance sampling [1] using a Probability Density Function (PDF). The main advantage of Importance Sampling is to minimize the variance error when the PDF is closer to the integrand. In off-line rendering, VPL generation can be performed using rejection sampling [12] or more complex sampling techniques based on Monte Carlo Markov Chain (MCMC) [13]. When real-time is targeted, an approach based on Instant Radiosity and Shadow Mapping [40], so-called Reflective Shadow Maps [7], has been proposed to approximate one bounce indirect lighting. It considers each pixel in the shadow maps as a secondary point light source defined by its world space coordinates, its normal and its flux, information that allows evaluating the contribution of each VPL. Computing the contribution of all the VPLs, stored in a Reflective Shadow Map, is time-consuming. This is why only a subset of VPLs is used to compute the indirect radiance of a pixel.

Evaluating visibility The determination of the visibility term is the most expensive operation, especially in real-time rendering. Instant Radiosity [22] is a popular technique that calculates indirect lighting due to a set of Virtual Point Lights (VPLs). Unlike the inverse transform method, Barák et al. [3] exploit the performances of Metropolis-Hastings algorithm [34] to determine the VPLs and use a small number of them to render the scene. Hedman et al. [17] have proposed a temporally coherent technique that allows sampling the VPLs in large scenes and enable frame to frame distribution for minimizing the VPL flickering.

Ritschel et al. [31], uses the observation that an approximate visibility term for VPLs is sufficient. The authors proposed a technique, called "Imperfect Shadow Map", which represents the scene surfaces by a set of points and splat them in parallel in the different shadow maps. The disadvantage of this technique is that the scene representation is not adaptative and may be not optimal for a large scene. This limitation has been solved by Ritschel et al. [32]. Moreover, the authors propose a new method to choose the VPLs which contribute much to the final image by creating a data structure, called Bidirectional Reflective Shadow Maps, based on a Cumulative Distribution Function (CDF). The method uses the inverse transform method which requires the computation of a CDF. A Rich-VPL method [35] handles glossy reflections with multiple primary light sources in the scene. Dammertz et al. [8] have proposed a progressive method to simulate indirect lighting. It combines and exploits the advantages of three methods: Virtual Point Lights, caustics, and specular gathering, to be able to render a large variety of global illumination effects.

Voxelizing the scene Usually directly using triangles (modelling the geometry of a scene) in real-time rendering scenario can be not efficient. To reduce the intersection computation, a scene can be subdivided into voxels. Several methods have been proposed [5, 6, 36]. Hu et al. [19] have presented a new ray tracing method, programmable on the modern GPU, which uses an A-Buffer and a grid voxelization to represent the scene geometry.

Clustering many lights methods rely on clustering to reduce the time needed to compute the contributions of the VPLs, such as the method proposed by Olsson et al. [29]. Dong et al. [9] use clustering to compute visibility. This clustering idea, in global illumination, has been wildly used [20, 34]. Hašan et al. [15] have introduced a Virtual Spherical Lights (VSL) method to resolve the singularity problem due to the VPLs. The lightcut methods [38, 39] try to avoid the VPL flickering.

Avoiding artifacts A real time based Instant Radiosity method has been proposed by Novák et al. [28] to approximate bias compensation and avoid the artifacts of the VPLs. Nabata et al. [27] have proposed a method to estimate more precisily the error due to VPL clustering.

Summary the goal of this paper is to provide an efficient algorithm for indirect computing illumination using one bounce indirect light and for selecting a set of VPLs with a higher contribution through Importance Sampling. To this end, we propose two inverse transform methods to compute a CDF used to efficiently select the most contributive VPLs, and we compare our methods to the one proposed by Ritschel et al. [32]. We also propose an MIS-based method combining two rendering methods: inverse transform and gathering-based.

3 System overview

In this section we summarize our contributions (Fig. 1): DPRSM construction, CDF computation, use of MIS combining inverse transform-based rendering and a gathering-based rendering. First, To reduce the cost of ray-object intersection, the scene is spatially subdivided into voxels according to the method proposed in [5]. Visibility computation is speed up using this voxelization. The scene is rendered from the camera viewpoint. We create a GBuffer that contains the position, normal and color of all the visible points (so-called gather points). Then, we build a Dual Paraboloid RSM (DPRSM) at each point light source.

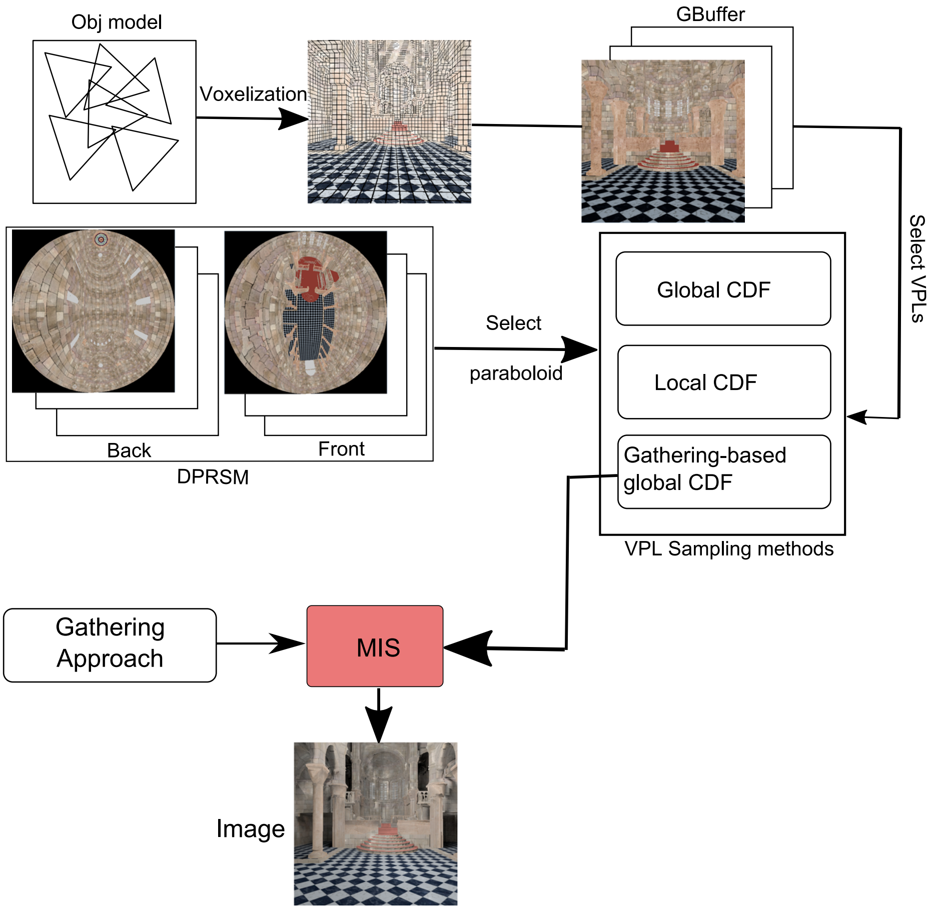


Fig. 1 Overview of our inverse transform methods and MIS for computing the indirect illumination at each gather point using different types of CDF

We propose two methods for computing the CDF: local CDF, and gathering-based global CDF. Recall that a CDF is used by an inverse transform method to randomly select a subset of VPLs (stored in the DPRSM) to compute the radiance of a gather point as the sum of the contributions of the selected VPLs. To improve the resulting rendered images we use an MIS approach combining an inverse transform method (based on local CDF or gathering-based global CDF) and a gathering-based rendering method.

Below, we summarize the state of the art method for computing a global CDF [32], while the main parts of our method (Fig. 1) are detailed in the following sections.

3.1 Computing a global CDF

The global CDF method refers to the method of [32] which consists in computing a CDF from just a small number of a selected gather points, say points visible to the viewpoint through pixels (Fig. 2). To compute a CDF, the method computes the average contributions (V_i) of all the VPLs to a small subset of gather points selected randomly. Then these average

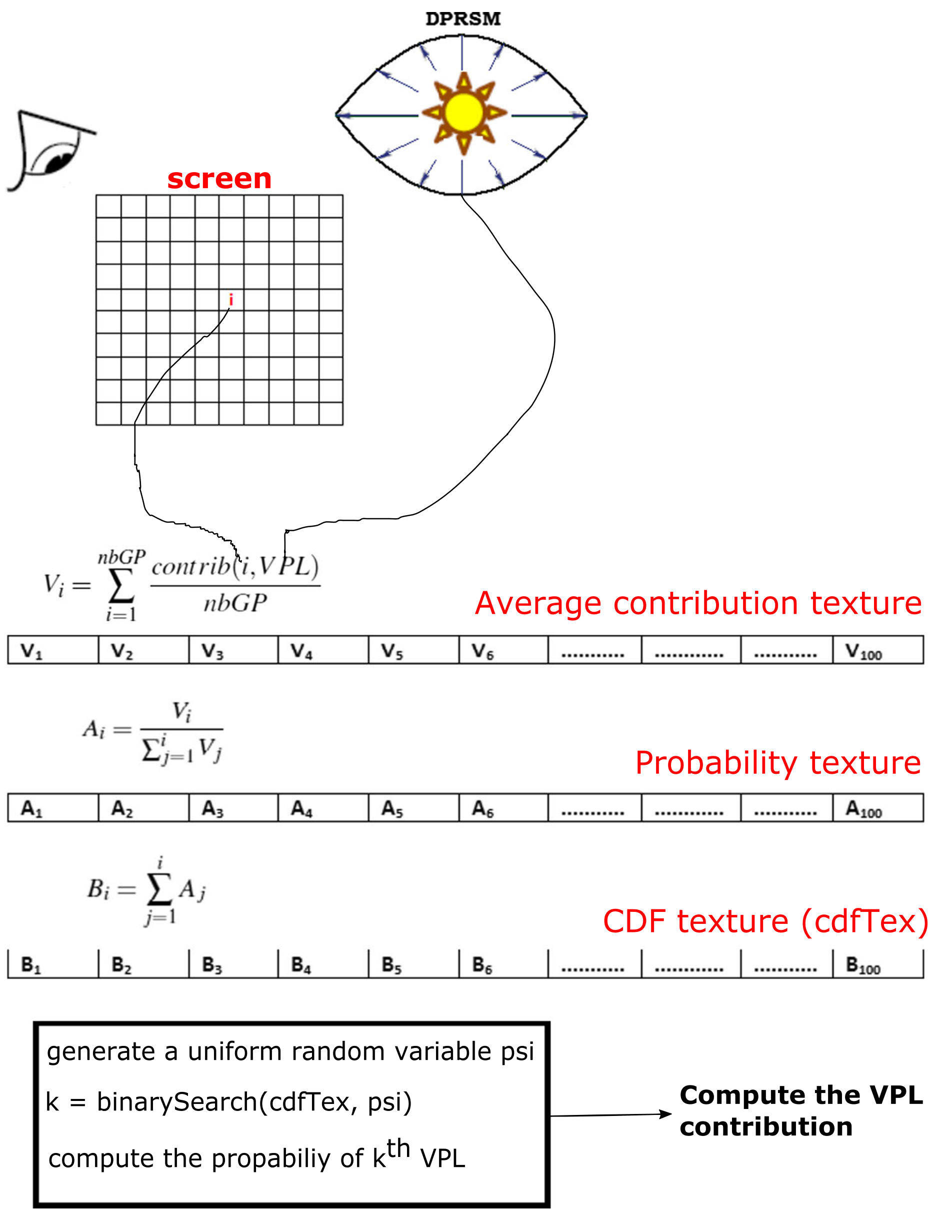

DPRSM
screen
i
$V_i = \sum_{i=1}^{nbGP} \frac{contrib(i, VPL)}{nbGP}$
Average contribution texture
V_1 | V_2 | V_3 | V_4 | V_5 | V_6 | .... | .... | .... | V_{100}
$A_i = \frac{V_i}{\sum_{j=1}^{i} V_j}$
Probability texture
A_1 | A_2 | A_3 | A_4 | A_5 | A_6 | .... | .... | .... | A_{100}
$B_i = \sum_{j=1}^{i} A_j$
CDF texture (cdfTex)
B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | .... | .... | .... | B_{100}
generate a uniform random variable psi
k = binarySearch(cdfTex, psi)
compute the propabiliy of kth VPL
Compute the VPL contribution

Fig. 2 Overview of global CDF method: illustration of the computation of a discrete CDF to generate mcVPLs

contributions are stored in a linear array (also stored in 1D texture) and then used to compute discrete probabilities (A_i) assigned to the VPLs. From these discrete probabilities, discrete CDF values (B_i) are computed and stored in a linear array.

3.2 Determining the most contributive VPLs

To determine the more contributive VPL (mcVPL), the inverse transform method is used in the discrete domain. In other words, a uniform random variable (ranging from 0 to 1) is generated, then a binary search is performed in the CDF array, to determine the CDF index k that represents the column $c = [\frac{k}{N}]$ and the row $r = modulo(k, c)$. The (r, c) pair represents the mcVPL coordinates in a PRSM.

In the rendering step, the contribution of each mcVPL is divided by its probability.

4 Dual paraboloid reflective shadow map (DPRSM)

A shadow map data structure has first been used for computing visibility from a point light to render cast shadows. It is an image of depth values rendered by placing a camera at the point light source position (called from now on light camera). Then, this shadow map has been augmented with other data such as $3D$ points visible from the point light source, the diffuse color of the objects containing these visible points, etc. This new shadow map (called Reflective Shadow Map) has been introduced by Dachsbacher et al. [7]. However, as the screen of the light camera is rectangular, its field of view is limited to a viewing pyramid. To cover a 180 degree field of view a Paraboloid Shadow Map (PSM) has been proposed by Heidrich et al. [18]. This kind of shadow map has been generalized by Gascuel et al. [11]. One of the more accurate Shadow Map, named Paraboloid Shadow Map, first proposed by [18], uses a new parameterization to represent the environment maps. [11] use a non-linear projection such as fish-eye lens with a larger field of view to cover all the $3D$ space. It requires only two rendering passes unlike a cube map [14] which needs six rendering passes. However, the main difficulty of this method is the conversion of triangles to curved triangles. A Dual Paraboloid Shadow Map, programmable on graphics hardware (GPU), has been proposed in [4] for an omnidirectional light source. In our system, we use a Dual Paraboloid Reflective Shadow Map (DPRSM) for each point light source. As in [11] this DPSM allows to provide a large field of view for covering the total $3D$ space, and requires two rendering passes.

We create two PRSMs, front and back, at the light source position oriented toward the z-axis of the associated $3D$ coordinates system. Then we transform each triangle into a single curved triangle, and project it into the PRSM space, each paraboloid containing the depth, normal and color of the points within the projected triangle.

Once the DPRSM has been created, for each visible point (from the view camera) a PRSM (front or back) is selected using a Russian roulette [2]. Then a VPL is randomly sampled from the selected PRSM using a CDF. Next, the contribution of the sampled VPL is computed. We repeat this process (iteration) until a subset of VPLs have been sampled. The contributions of all the sampled VPLs are summed to give the indirect radiance of the visible point. To apply a Russian roulette we compute the probability bF (respectively bB) of choosing a front PRSM (respectively a back PRSM) of the DPRSM by computing the sum of the average contributions vF and vB of all the VPLs stored in a PRSM (front or back) similarly to [32] (see Sections 3.1 and 3.2 for more details). The PRSM selection probabilities are the normalized average contributions of all the VPLs: bF = vF/(vF+vB) and bB = vB/(vF+vB). The Russian roulette algorithm is given by Algorithm 1.

Algorithm 1 face selectParaboloid(FACE face, random psi)

```
1: vF = sumContrib(front); // average contribution sum for front face
2: vB = sumContrib(back); //average contribution sum for back face
3: bF = vF/(vF+vB);
4: bB = vB/(vF+vB);
5: if psi < bF then
6:     return front_face;
7: else
8:     return back_face;
9: end if
```

5 VPL sampling methods

In this section we propose two methods for computing a CDF used in an inverse transform approach to sample VPLs from a DPRSM. From now on, they will be called *local* and *gathering-based*. Note that the objective is to sample the most contributive VPLs based on an importance function which is the CDF. Before detailing these two methods we show in the following subsection how to compute the radiance $L(x)$ at a point x due to a certain number of randomly selected VPLs.

5.1 Computing the contribution of a VPL to a gather point

Let us see now how to compute the radiance $L(x)$ at a point x resulting from the contributions of a certain number of randomly selected VPLs. Let us assume that a VPL is a very small surface with normal N and flux ϕ_v (emittance in this case, say flux emitted per unit surface). We can consider this VPL as a point light source which has an emittance ϕ_v and an intensity I_v (flux emitted per unit solid angle). The VPL intensity I_v is expressed as [7]:

$$I_v = \phi_v \frac{cos\theta_1}{\pi} \tag{1}$$

where θ_1 is the angle between the normal N at the VPL v and the lighting direction from the VPL. All the used notations are given in Fig. 3.

For a diffuse surface (here a VPL v) there is a relation between its emittance ϕ_v and its radiance L' [10]:

$$\phi_v = L'\pi \tag{2}$$

The radiance L' of the VPL v due to a point light source is given by:

$$L'(v) = I_s \frac{cos\alpha}{d_1^2} f_r^d(v) \tag{3}$$

where I_s is the intensity (flux emitted per unit solid angle) of the point light source and $f_r^d(v)$ the diffuse BRDF of the surface containing the VPL v. Finally, given a VPL v, we compute its radiance L' using (3), then its emittance ϕ_v (2) and its intensity I_v (1). Using the Monte Carlo integration, we perform these computations for all the VPLs to calculate the radiance $L(x)$ at a gather point x due N randomly selected VPLs as follows [12]:

$$L(x) = \sum_{v=1}^{N} I_v f_r^d(x) \frac{cos\theta_2}{d_2^2} \frac{1}{p_v} \tag{4}$$

where p_v is the pdf (probability density function, corresponding to the used CDF) of the accepted VPL v, I_v the intensity of VPL v, and $f_r^d(x)$ the BRDF at the visible point x (see Fig. 3 for the other notations).

Note that the radiance of the gather point x due to a VPL (not randomly selected) is given by:

$$L(x) = I_v f_r^d(x) \frac{cos\theta_2}{d_2^2} \tag{5}$$

5.2 Local CDF

In this section, we describe our first method (that we call local CDF) which uses N CDFs unlike a method which uses a single CDF also called global CDF. As detailed in Section 3.1 let us summarize how a global CDF is computed using the method presented in [32]. Once an RSM (in our methods we use a DPRSM) is computed, a subset S_{GP} of gather points GP is randomly selected (corresponding to a subset of pixels). Then, for each VPL within the DPRSM, its contribution to each GP of S_{GP} is computed using (5). Then each VPL v is assigned an average contribution which is equal to the sum of its contributions to the GP of the subset S_{GP} divided by the cardinal of S_{GP}. These VPL average contributions help build a CDF which will be used to sample VPLs from the constructed DPRSM (see Section 3.1 for more details). The way the $GPs \in S_{GP}$ are distributed over the image plane is crucial for an efficient calculation of a CDF (efficient importance sampling). To better distribute

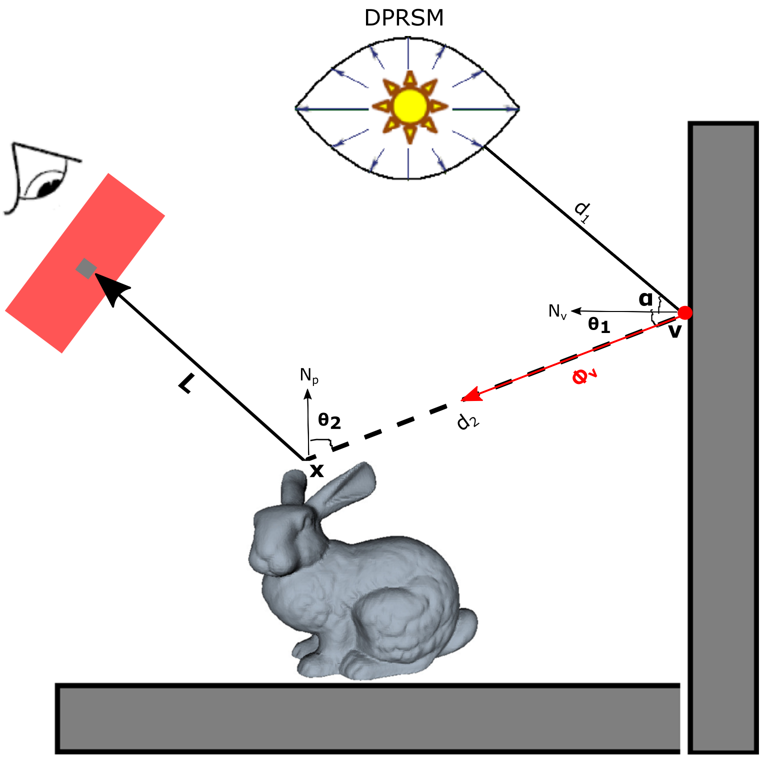


Fig. 3 Computation of indirect lighting due to VPLs

these GP, we propose to subdivide the image plane into N regions. For each region i, we perform a uniform sampling to get a subset of GPs, called S^i_{GP}, used to compute a local CDF (called CDF_i). Once all the local CDF_i are computed, to render an image we compute the radiance of all the GPs within a region i using only CDF_i. Even though this approach seems interesting, it is source of artifacts consisting of discontinuity at the region boundaries as shown in Fig. 4. To overcome this problem we proceed as follows. In the rendering step, to compute the radiance of each GP of the image plane, all the N CDF_i are sampled at the same time, then the contributions of the N selected VPLs are computed and assigned to the GP. This process is repeated N_{NG} times. Thus, the radiance of each GP requires the sampling of $N_{NG} \times N$ VPLs.

5.3 Gathering-based global CDF

We describe in this section our second method for computing a global CDF used in an inverse transform approach. We call this method: Gathering-Based Global CDF (called GBG from now on). This method is global because it does not need a subdivision of the image plane into regions. Our objective is to compute a more efficient global CDF than the one proposed by Ritschel et al. [32]. Our approach differs from this method in the way the VPL average contributions (see Section 3.1 for more details) are computed. Indeed, rather than computing these contributions for a small subset of GPs, our approach computes them for all the GPs. Our GBG method works as follows (Algorithm 2). At each GP, a hemisphere is placed above it, then a set of rays are randomly (according to a pdf) traced from the GP. For each ray, the first intersection point P is computed using Ray Marching through the voxel-based subdivision of the scene [5] (line 7). Then we project P onto the shadow map DPRSM (line 11). If P is visible from the DPRSM then it corresponds to a VPL. In this case we compute its contribution (see (5)) to the current GP which is stored in a GBuffer. This contribution is updated when considering the rest of the GPs. This process is repeated for all the GPs. The result is the total average contributions of all the VPLs stored

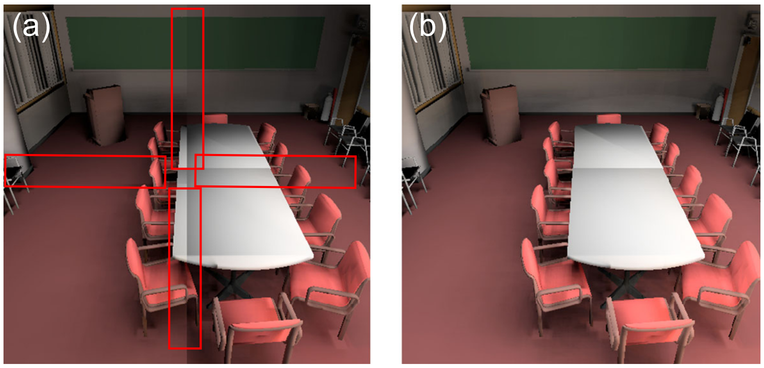


Fig. 4 Illustration of the discontinuity problem when we use one CDF for each gather point. In this example, the image plane is subdivided into 4 regions, **a** local CDF method with only one CDF for each GP of a region, we see discontinuity at the boundaries of regions; **b** local CDF method with 4 CDF per GP, the discontinuities have disappeared

in the DPRSM. The computed average contributions (line 24) are stored in a texture. As the method is based on ray sampling through a GP hemisphere, it may happen that some VPLs are not reached by the sampled rays, consequently there contribution is null, which corresponds to holes in the associated texture. To fill the holes, we propose to use a 3×3 median filter as a reconstruction filter.

The computation of an average contribution of all the VPLs is described by Algorithm 2.

Note that the average contributions is a luminance that is determined by converting an RGB color into a scalar value called luminance using the following formula:

$$Luminance = 0.299 * R + 0.587 * G + 0.114 * B \tag{6}$$

Algorithm 2 *2D_texture_image* compute_average_contribution()

```
1: gBuffer = GenerateGBuffer(); // a buffer containing the position, normal, color of the visible points from the camera view
2: dprsm = GenerateDualParaboloidRSM(); // front and back buffers containing position, normal, color of the visible points from the light source
3: for each gather point $(i, k)$ from the gBuffer do
4:     for j = 1 to #NBDIR do
5:         // NBDIR: number of random directions in the hemisphere
6:         dir = randomDirection();
7:         its = rayMarching(dir);
8:         if its == 1 then
9:             // 1 if intersection point found by Ray Marching
10:            P = computeOutgoing(); // intersection point P
11:            uvVPL = projectPraboloid(P, dprsm);
12:            // project the outgoing point P into the DPRSM and return its UV coordinates in the variable $uvVPL$
13:            if visible(uvVPL, front) then
14:                // if the VPL belongs to the front PRSM
15:                nbVPLFront = nbVPLFront + 1; // nbVPLFront number of the visible VPLs in the front face
16:            else
17:                if visible(uvVPL, back) then
18:                    // if the VPL belongs to the back PRSM
19:                    nbVPLBack = nbVPLBack + 1; // nbVPLBack number of the visible VPLs in the back face
20:                end if
21:            end if
22:        end if
23:    end for
24:    average_contrib[i]k] += computeContribution(uvVPL) / (nbVPLFront + nbVPLBack); // computeContribution() uses (5)
25: end for
26: return average_contrib; // image of average contributions
```

Given the VPL average contributions, we compute a CDF (called GBG) according to Ritschel et al.'s method (see Section 3.1). We render the scene using this CDF as follows (Algorithm 3). First from each gather point visible from the view camera, we randomly sample a small number of VPLs that are selected from the DPRSM. In using a Russian roulette, we sample the VPL from the front PRSM (line 9) or from the back PRSM (line 14). As we use the Russian roulette technique, the final contribution is divided by the propability of selecting the front or the back face (line 10 and line 15).

Algorithm 3 *2D_texture_image* renderingWithCDF()

```
1:  gBuffer = GenerateGBuffer(); // a buffer containing the position, normal, color of the visible points from the camera view
2:  dprsm = GenerateDualParaboloidRSM(); // front and back buffers containing position, normal, color of the visible points from the light source
3:  for each gather point $(i, k)$ from the gBuffer do
4:      for j = 1 to #NBVPL do
5:          // NBVPL: small number of VPLs
6:          psi = sample_uniform_random_variable(); // psi ranging from 0 to 1
7:          if selectParaboloid(face, psi) == front_face then
8:              //see Algorithm 1
9:              uvVPL = sampleFromFront(); // select the VPL of coordinates uvVPL from the front face
10:             image[i][k] += computeContribution(uvVPL) / bF //bF: probability of selecting the front face
11:         else
12:             if selectParaboloid(face, psi) == back_face then
13:                 //see Algorithm 1
14:                 uvVPL = sampleFromBack(); // select the VPL of coordinates uvVPL from the back face
15:                 image[i][k] += computeContribution(uvVPL) / bB //bB: probability of selecting the back face
16:                 // computeContribution() uses (4)
17:             end if
18:         end if
19:     end for
20: end for
21: return image; // rendered image
```

6 A multiple importance sampling approach

In this section, our objective is to improve a CDF-based rendering method (inverse transform) by combining it with a gathering-based rendering method. We propose *Multiple Importance Sampling* (MIS) to carry out this combination of two estimators.

6.1 Background on MIS

Let us compute two Monte Carlo estimators of the integral of $f(x)$, one with a sampling distribution (PDF) $p_1(x)$ and the other with PDF $p_2(x)$.

In our case we have two rendering strategies:

1. Gathering approach: Monte Carlo method sampling a hemisphere placed above a GP and using a cosine PDF;
2. Inverse transform method: using a CDF computed with any approach local or global.

The MIS strategy consists in combining the two Monte Carlo estimators as described by Veach [37]:

$$F = \frac{1}{n_1} \sum \omega_1(X_{1,j}) \frac{f(X_{1,j})}{p_1(X_{1,j})} + \frac{1}{n_2} \sum \omega_2(X_{2,j}) \frac{f(X_{2,j})}{p_2(X_{2,j})} \tag{7}$$

We use a weighting function defined by Veach [37] to combine the two estimators. The set of weights given by this function allows to generate samples $X_{1,j}$ or $X_{2,j}$ to reduce the variance.

Veach proposes two balance heuristic weights associated with each strategy:

$$\omega_1(X_{1,j}) = \frac{p_1(X_{1,j})}{p_1(X_{1,j}) + p_2(X_{1,j})} \tag{8}$$

$$\omega_2(X_{2,j}) = \frac{p_2(X_{2,j})}{p_1(X_{2,j}) + p_2(X_{2,j})}, \tag{9}$$

$X_{1,j}$ and $X_{2,j}$ are the samples of the random variable x generated with the PDF p_1 and p_2 respectively. In our case these samples are pairs (θ, ϕ) (elevation angle, azimuthal angle) representing a direction of a ray. Note that, when using p_2 a VPL is associated with each sample direction. The first distribution (PDF) $p_1(X_{1,j})$ is used to sample a hemisphere above a GP used by the gathering-based rendering method

$$p_1(X_{1,j}) = \frac{cos\theta sin\theta}{\pi}, \tag{10}$$

where θ is the polar angle formed by the sample ray and the normal at the GP, while the second distribution $p_2(X_{2,j})$ is used to sample VPLs according to one inverse transform method (local or GBG).

$$p_2(X_{2,j}) = \alpha_{ij} \tag{11}$$

where α_{ij} is the PDF values associated with the chosen CDF (see Section 5.3).

6.2 Description of our MIS method

We describe in this section our general algorithm that uses the MIS principle for rendering. We show how to combine the two estimators (gathering estimator and gathering-based global CDF estimator). To weight the contributions, we use the balance heuristic method as described in Section 6.1. Our main goal is to compute the PDF of each strategy when we generate samples from the other strategy ($p_1(X_{2,j})$ and $p_2(X_{1,j})$).

Our algorithm consists of four Parts:

1. Run a gathering method to generate the average contribution texture that will be used to compute the CDF: result = gathering-based global CDF (GBG CDF). See Section 5.3.
2. Run a classical gathering-based rendering: for each gather point and for each incident direction, we compute the incident radiance due to a VPL (VPL corresponding to the projection into the the DPRSM of the first ray-scene intersection point). The result is a contribution (contrib_gathering) to all the gather points weighted by ω_1 (see Algorithm 5).
3. For each gather point of the previous step, select N VPLs using the gathering-based global CDF: the result is a contribution (contrib_CDF) for all the gather points weighted by ω_2 brought by the N selected VPLs (see Algorithm 6).
4. Sum the two contributions contrib_CDF and contrib_gathering (using (7)).

Algorithm 4 summarizes our proposed method.

Algorithm 4 *2D_texture_image* MIS()

```
1: gBuffer = GenerateGBuffer(position, normal, color); // position, normal, color of the visible point
2: dprsm = GenerateDualParaboloidRSM(); // front and back buffers containing position, normal, color of the visible points from the light source
3: // pdf1 = $p_1(X_{1,(i,k)})$, pdf2 = $p_2(X_{1,(i,k)})$
4: contrib_gathering = MISdensityGathering(pdf1, pdf2);
5: // pdf3 = $p_2(X_{2,(i,k)})$, pdf4 = $p_1(X_{2,(i,k)})$
6: contrib_CDF = MISdensityVPL(pdf3, pdf4);
7: // Final contribution
8: for each gather point $(i, k)$ from the gBuffer do
9:     image[i][k] = (contrib_gathering[i][k] + contrib_CDF[i][k]) / NBDIR; // $NBDIR$: number of incident directions shooted from each gather point $j$
10: end for
11: return image; // rendered image with MIS and stored in a 2D_texture
```

To apply the MIS technique, we have to find the VPL PDF values of the j samples from the gathering strategy (noted $p_2(X_{1,(i,k)})$) and the gathering distribution of j samples from the gathering-based global CDF strategy when we compute the VPL contribution (noted $p_1(X_{2,(i,k)})$).

The method of computing $p_2(X_{1,(i,k)})$ is described in detail by Algorithm 5. The gathering PDF $p_1(X_{2,(i,k)})$ of the (i, k) sample, generated from the gathering-based global CDF, is computed as:

$$p_1(X_{2,(i,k)}) = p_2(X_{2,(i,k)}) \times p_{pv} \tag{12}$$

Where $p_2(X_{2,(i,k)})$ is the probability of selecting a VPL according to the gathering-based global CDF method, p_{pv} is the probability of connecting the gather point p to the selected VPL v. As this connection is deterministic its distribution (pdf) is equal to 1.

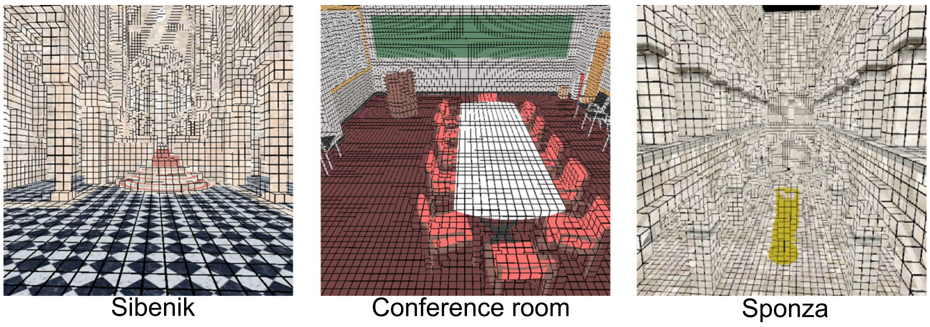


Fig. 5 **a**, **b** and **c** show the voxelization-based approach of the Sibenik, the Conference scene and the Sponza Buddha scene respectively. We use a voxel grid with a resolution equal to 128^3

Algorithm 6 illustrates the method of computing $p_1(X_{2,(i,k)})$.

Algorithm 5 *2D_texture_image* MISdensityGathering(PDF pdf1, PDF pdf2)

```
1:  gBuffer = GenerateGBuffer(position, normal, color); // generate the position, normal, color of the visible point from the view camera
2:  dprsm = GenerateDualParaboloidRSM(); // front and back buffers containing position, normal, color of the visible points from the light source
3:  for each gather point $(i, k)$ from the gBuffer do
4:      for j = 1 to #NBDIR do
5:          // NBDIR: number of random directions in the hemisphere
6:          dir = randomDirectionPropToCosine(x, y, z);
7:          pdf1 = dir.z / pi;
8:          its = rayMarching(dir);
9:          if its == 1 then
10:             // intersection found
11:             outgoing = computeOutgoing();
12:             uvVPL = projectPraboloid(outgoing, dprsm);
13:             // project the outgoing point into the DPRSM and return the UV coordinates of the associated VPL
14:             if visible(uvVPL) then
15:                 pdf2 = getPDF(uvVPL); // retrieve the pdf value associated with the VPL
16:                 w1 = balanceHeuristic(pdf1, pdf2); // see (8)
17:                 contrib_gathering[i][k] += $(w1 \times computeContribution(uvVPL))$ / pdf1; // computeContribution() uses (4)
18:             end if
19:         end if
20:     end for
21: end for
22: return contrib_gathering; // image generated with a gathering-based rendering method
```

Algorithm 6 *2D_texture_image* MISdensityVPL(PDF pdf3, PDF pdf4)

```
1: gBuffer = GenerateGBuffer(position, normal, color); // generate the position, normal, color of the visible point from the view camera
2: rsm = GenerateDualParaboloidRSM(); // front and back buffers containing position, normal, color of the visible points from the light source
3: for each gather point $(i, k)$ from the gBuffer do
4:     for j = 1 to #NBVPL do
5:         vpl = getVPL(); // selected VPL according to a global CDF
6:         pdf3 = getPDF(vpl); // pdf $p_2(X_{2,(i,k)})$ of the selected VPL determined by VPL method
7:         gp = pointFromGBuffer(gBuffer); // gp = gather point
8:         distance = length(vpl - gp);
9:         dir = normalize(distance);
10:        outgoing = rayMarching(gp, dir);
11:        if (outgoing - distance) ¡ psi then
12:            // visible gather point
13:            contrib = computeContribution(); // computeContribution() uses (4)
14:            theta = computeAngle(normal, dir);
15:            pdf4 = $(cos(theta) \times sin(theta))/pi$;
16:            w2 = balanceHeuristic(pdf4, pdf3); // see (9)
17:            contrib_CDF[i][k] += $(w2 \times computeContribution(vpl))/pdf3$; // computeContribution() uses (4)
18:        end if
19:    end for
20: end for
21: return contrib_CDF; // image generated with a GBG CDF-based method
```

To compute $p_1(X_{2,(i,k)})$ we search for the angle θ between the normal **N** of the gather point and the direction **pv** formed by the gather point and the selected VPL.

We have:

$$cos\theta = \langle \mathbf{N}, \mathbf{pv} \rangle \tag{13}$$

$$\theta = acos(\langle \mathbf{N}, \mathbf{pv} \rangle) \tag{14}$$

So we can compute $p_1(X_{2,(i,k)})$ as:

$$p_1(X_{2,(i,k)}) = \frac{cos\theta sin\theta}{\pi} \tag{15}$$

7 Results and evaluation

In this section, we show some results obtained with our two CDF-based methods (local CDF and gathering-based global CDF) as well as our MIS method. We validate our results in terms of computation speed, and objective and perceptual qualities. Our test scenes contain only static diffuse objects. We have used three scenes: Sibenik , Conference room and Sponza Buddha scenes. We have placed one point light source in these scenes and considered single bounce indirect illumination. Our methods run on the GPU (NVIDIA Geforce GTX 780 OC 6GB) and on the CPU (Intel i7 930 @ 2.80 GHZ, 8GO RAM running 64 bits) using OpenGL 4.3.

In the following, we describe the specifications of each scene. The three scenes used in this paper are available in [25]. Their common characteristic is that they contain only diffuse BRDF because our proposed methods run for diffuse objects only. The Conference room scene contains 331,179 triangles without textures. While the Sibenik scene is a closed scene containing 75,284 triangles with textured objects. On the other hand, the Sponza Buddha scene is an open scene.

We illustrate in Fig. 5 the voxelization-based method (to speed up visibility calculation) by Crassin and Green [5] for Sibenik, Conference, and Sponza Buddha scenes. To capture the scene details, we choose a voxel grid with a resolution 128^3.

The images, generated with our method, have been compared to reference images computed with Ritschel et al.'s global CDF method (see Section 3.1). These reference images have been generated using a global CDF computed with a large number ($4k$) of gather points (used to compute the average contributions of the VPLs) and a large number of VPLs ($10k$ VPLs) used to compute the radiance of each gather point during rendering. The resolution of the computed images is 512×512.

When rendering using a local CDF or gathering-based global CDF (GBG CDF), 800 VPLs are selected randomly according to the used CDF. One DPRSM is computed for the point light source placed in the scene. Recall that a DPRSM consists of two PRSMs: front and back. Each PRSM is assigned three buffers: position buffer, normal buffer and color buffer. Figure 6 shows the two color buffers (front and back) of the DPRSM associated with the Sibenik, Conference and Sponza Buddha scenes. The figure shows that each PRSM (front or back) covers a 180 degree field of view.

Figure 7 provides results obtained with our local CDF, our GBG CDF and our MIS methods together with the reference image of the three test scenes. Regarding the local CDF approach, we have subdivided an image into four regions (say $N = 4$, see Section 5.2). Images (b), (f) and (j) provide better results than those obtained with the global CDF-based method [32] (images (a), (e) and (i)) since some regions of the image are better shaded while they look dark when using the global CDF-based method. This can be explained by the fact that with a local CDF the gahtering points (used to build a CDF) are uniformly distributed over the image (similar to stratification).

For the same reasons, our GBG CDF method gives better results (images (c) (g) and (k)) compared to the global CDF method [32] (images (a), (e) and (i)). This is due to the fact the global CDF-based method assumes that a gather point is visible from all the VPLs when computing the CDF, which is a strong assumption. Rather, our GBG CDF computes visibility through Ray Marching in a voxel grid.

Table 1 gives some rendering times in milliseconds for four methods: global CDF, Local CDF, gathering-based CDF and MIS. Our GBG CDF and MIS methods are faster than the global CDF-based method. While generating better results, our local CDF-based method is slower than the global CDF-based method because it repeats four times (one for each region) the process of computing a CDF using a method similar to that of the global CDF-based method.

Table 2 summarizes the time for generating the GBuffer containing all the information (position, normal, color) of each visible point (gather point). It also provides the time for generating the DPRSM and the visibility time that is computed using a Ray Marching algorithm. Note that the shadow maps resolution is 512×512. The time is computed in milliseconds. We observe that the Ray Marching algorithm (visibility computation) is the most expensive step for each scene (Sibenik 79.86 ms, Conference 78.71 ms, and Sponza Buddha 80.27 ms). Furthermore, the DPRSM generation step takes a few milliseconds for

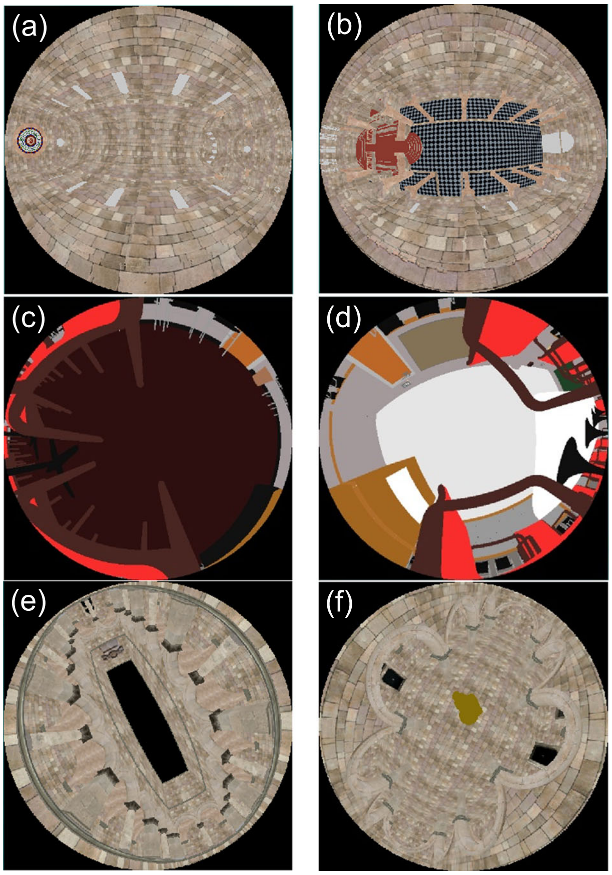


Fig. 6 **a**, **c** and **e** show the color buffer of the front face corresponds to all the triangles of the Sibenik scene, the Conference scene and the Sponza scene respectively. **b**, **d**, and **f** correspond to the back face color buffer for the three scenes respectively

the Conference (4.54 ms) and the Sponza Buddha (5.64 ms) scenes but it takes 33.85 ms for the Sibenik scene because the Sibenik scene contains a large number of points stored in the DPRSM. For the same reasons, we found that the GBuffer time generation is higher for the Sibenik scene (15.23 ms) compared to the Conference scene (0.82 ms) and the Sponza Buddha scene (1.06 ms).

Now we evaluate our local-based CDF and gatheirng-based global CDF methods by using the RMSE metric and the HDR-VDP-2 [24] metric which is a perceptual metric applied to HDR images (High Dynamic Range). Note that all the images generated by our method are HDR images and the reference images are generated with the global CDF-based method as explained above.

Figure 8 gives the RMSE for the local CDF-based and gathering-based CDF methods as a function of the number of VPLs selected during the rendering step. We can notice that the RMSE is small for the two methods and decreases when the number of VPLs increases. An interesting result is that, for the gathering-based CDF method, the RMSE reaches its smaller

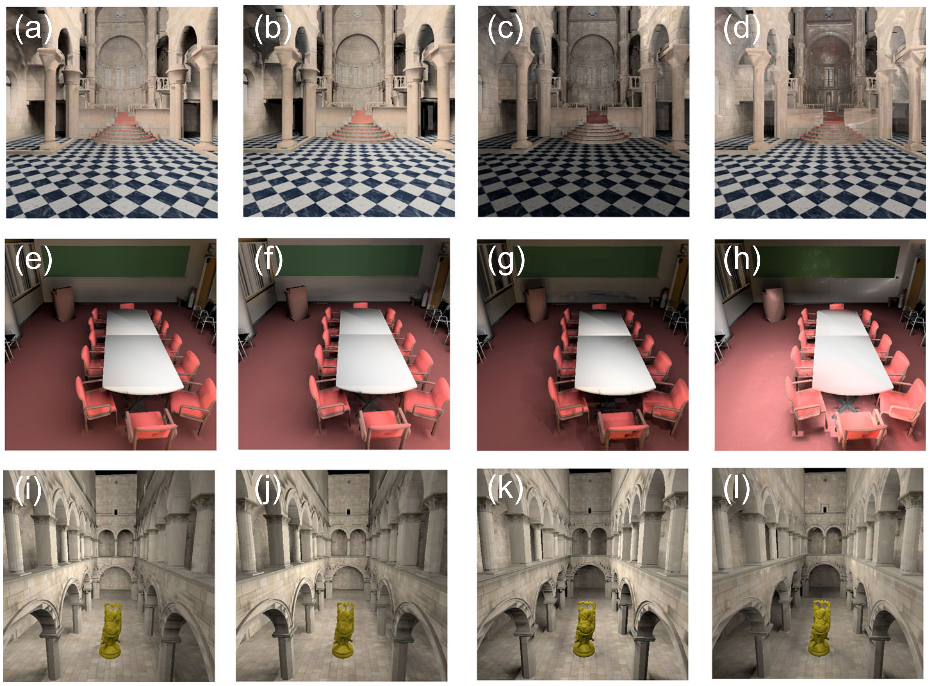


Fig. 7 The Sibenik, Conference, and Sponza Buddha scenes have been rendered using 800 randomly selected VPLs per gather point and 800 directions for the MIS method. Image (**a**), (**e**), and (**i**) are the reference images for the three scenes generated with the global CDF-based method [32], the contribution of each gather point is computed using a large number of VPLs (10*k* VPLs). Images (**b**), (**f**), and (**j**) have been computed using our local CDF. Images (**c**), (**g**) and (**k**) have been generated with our GBG CDF method. Our MIS method provides the images (**d**), (**h**), and (**l**)

Table 1 Time rendering of our local CDF, GBG CDF, and MIS methods compared to the global CDF method

Scene	Sibenik	Conference	Sponza Buddha
Global CDF	600 ms	560 ms	270 ms
Local CDF	630 ms	640 ms	410 ms
GBG CDF	350 ms	340 ms	190 ms
MIS method	460 ms	460 ms	230 ms

This time rendering is computed in milliseconds for the three scenes using the same number of VPLs (800 VPLs for each method)

Table 2 Time for generating GBuffer, DPRSM, Ray Marching for Sibenik, Conference, and Sponza scenes in millisconds. The Shadow Map resolution is 512 × 512

Scene	Sibenik	Conference	Sponza Buddha
GBuffer	15.23 ms	0.82 ms	1.06 ms
DPRSM	33.85 ms	4.54 ms	5.64 ms
Ray marching	79.86 ms	78.71 ms	80.27 ms

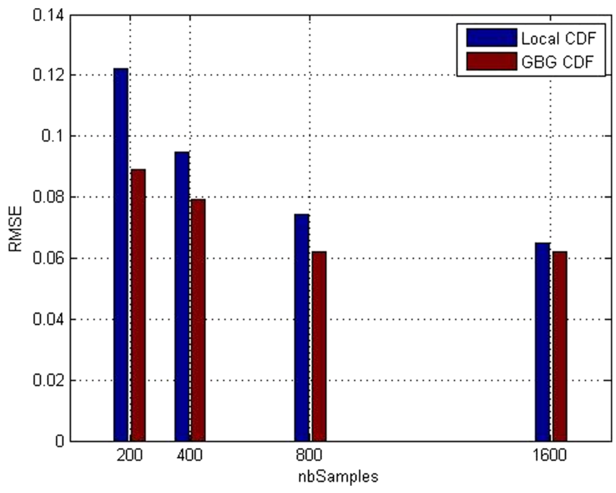


Fig. 8 RMSE results for the Conference scene as a function of the number of VPLs used for rendering

value for a number of 800 VPLs, which means that this number is sufficient for getting good quality results.

Figure 9 shows the perceptual difference between the images, generated with our local CDF-based and gathering-based CDF methods, and the reference images. The perceptual differences are evaluated using the HDR-VDP-2 [24] perceptual metric applied to HDR images.

We observe that most of the image of the test scenes is assigned a low error given by the HDR-VDP-2 metric (blue and green parts in figures a, c, e, g, i, k). Therefore, our local CDF and our GBG CDF images closely resemble the reference image. However, there exist small regions with an non negligible error (a, c, e, g, i, k). To reduce these high values (red parts), we can increase the number of VPLs during rendering.

Figure 10 shows results obtained with the gathering-based CDF and MIS methods for the Sibenik, Conference and Sponza Buddha scenes. We have used the same number of samples (a sample is: a VPL for the gathering-based CDF approach, and a direction for the MIS) to generate the images of Fig. 10.

8 Conclusion and future work

In this paper, we have presented three VPL-based rendering methods: local CDF, GBG CDF, and MIS. These methods allow to improve the selection of the most contributive VPLs stored in a Dual Paraboloid RSM (that we called DPRSM). Our methods consider only one bounce indirect illumination because we use a DPRSM placed at the point light source. We use the Russian roulette technique to select one face between the front and the back faces of the DRSM. All the results are concerned with only indirect illumination. For visualizing our results on LDR displays, our HDR images have been tone-mapped using Reinhard's operator [30]. Our results show that our local CDF-based method, that divides the screen into N regions, generates good images in term of quality but requires more computing time compared to our other methods. This is why we have proposed our gather-based global CDF to lower the time rendering.

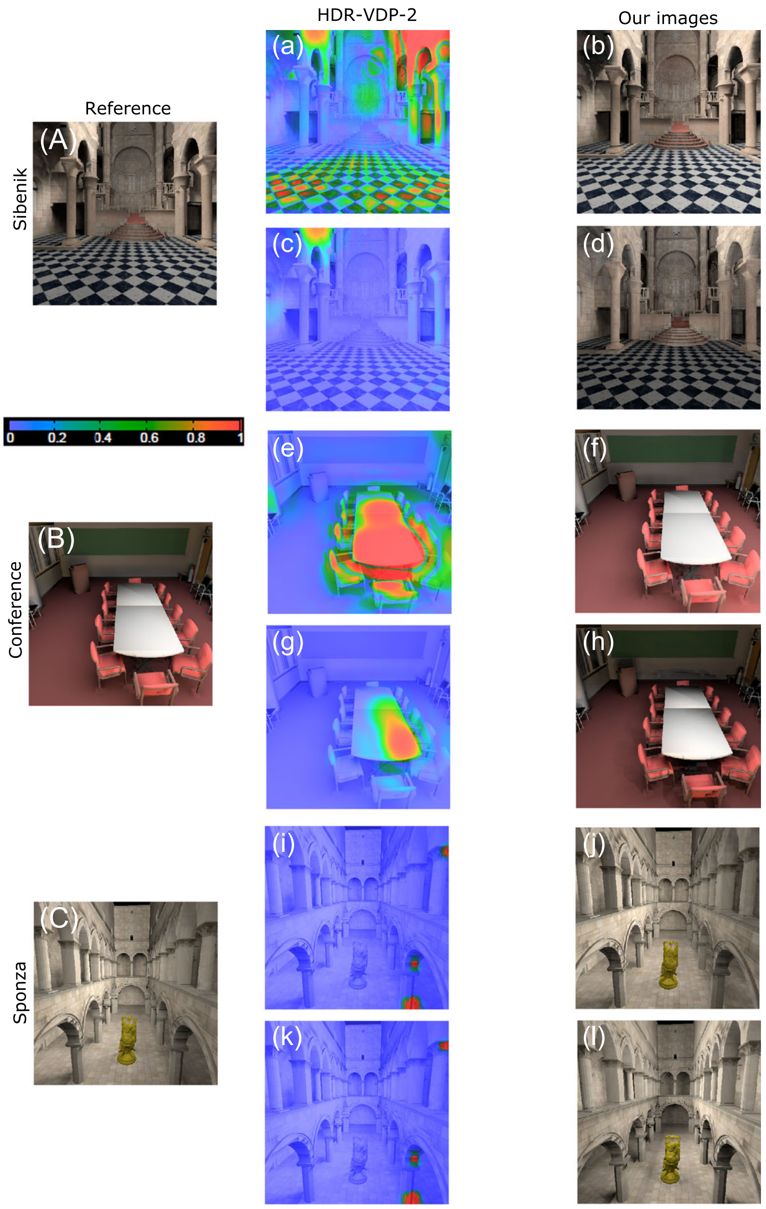


Fig. 9 HDR-VDP-2 metric between the reference images and those obtained with our methods for the three test scenes. The first column represents the reference image for the Sibenik (**A**), Conference room (**B**), and Sponza (**C**); the second column shows the HDR-VDP-2 images and the third one shows our results. Images (**b**), (**f**), and (**j**) have been generated with our local CDF method. Images (**d**), (**h**), and (**l**) have been generated with our GBG CDF. Images (**a**), (**e**), and (**i**) provide the HDR-VDP-2 metric between the reference images (image (**A**), (**B**), (**C**)) and the images (**b**), (**f**), and (**j**) respectively. Images (**c**), (**g**), and (**k**) represent the HDR-VDP-2 metric between the reference images (image (**A**), (**B**), (**C**)) and the images (**d**), (**h**), and (**l**) respectively

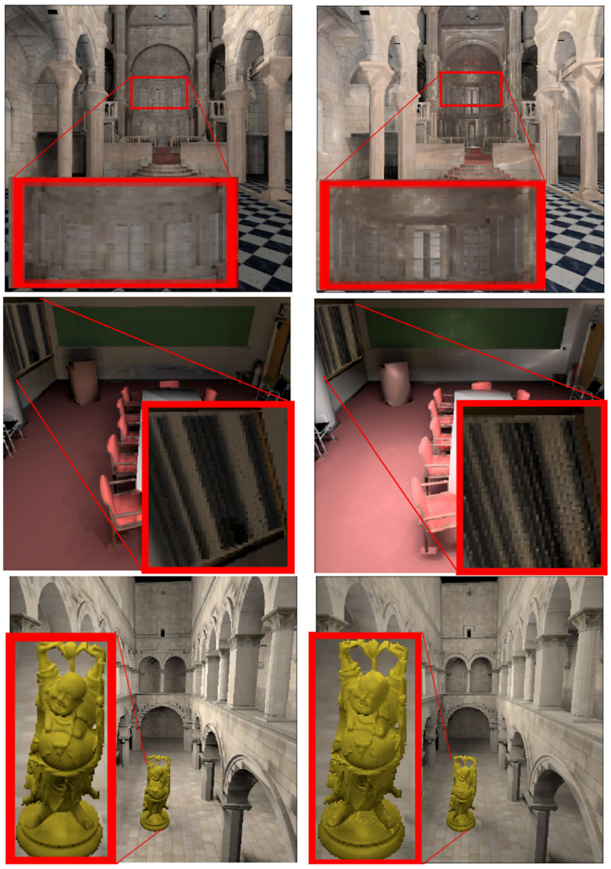


Fig. 10 Comparison of our GBG CDF method without MIS (left column) and our GBG CDF method with MIS (right column)

Furthermore, we have applied the HDR-VDP-2 metric to show the perceptual differences between our HDR images and the reference images obtained with the global CDF method [32] with a high number of sample VPLs.

We have focused on the diffuse surfaces only. So, as future work, it would be worth to adapt our algorithms to handle glossy surfaces and caustics. The shown results have been generated from the static scenes. How to extend our methods to dynamic scenes. This is left for future work.

References

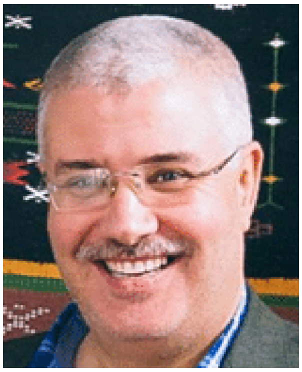
1. Agarwal S, Ramamoorthi R, Belongie S, Jensen HW (2003) Structured importance sampling of environment maps. ACM Trans Graph (TOG) 22(3):605–612
2. Arvo J, Kirk D (1990) Particle transport and image synthesis. ACM SIGGRAPH Computer Graphics 24(4):63–66
3. Barák T, Bittner J, Havran V (2013) Temporally coherent adaptive sampling for imperfect shadow maps. In: Computer graphics forum, wiley online library, vol 32, pp 87–96

4. Brabec S, Annen T, Seidel HP (2002) Shadow mapping for hemispherical and omnidirectional light sources, Springer
5. Crassin C, Green S (2012) Octree-based sparse voxelization using the gpu hardware rasterizer. OpenGL Insights 303–318
6. Crassin C, Neyret F, Lefebvre S, Eisemann E (2009) Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In: Proceedings of the 2009 symposium on interactive 3d graphics and games. ACM, pp 15–22
7. Dachsbacher C, Stamminger M (2005) Reflective shadow maps. In: Proceedings of the 2005 symposium on interactive 3d graphics and games. ACM, pp 203–231
8. Dammertz H, Keller A, Lensch HP (2010) Progressive point-light-based global illumination. In: Computer graphics forum, wiley online library, vol 29, pp 2504–2515
9. Dong Z, Grosch T, Ritschel T, Kautz J, Seidel HP (2009) Real-time indirect illumination with clustered visibility. In: VMV, pp 187–196
10. Dutré P (2003) Global illumination compendium. Computer Graphics, Department of Computer Science Katholieke Universiteit Leuven 6
11. Gascuel JD, Holzschuch N, Fournier G, Peroche B (2008) Fast non-linear projections using graphics hardware. In: Proceedings of the 2008 symposium on interactive 3d graphics and games. ACM, pp 107–114
12. Georgiev I, Slusallek P (2010) Simple and robust iterative importance sampling of virtual point lights. Proceedings of Eurographics (short papers) 4
13. Gilks WR (2005) Markov chain monte carlo. Encyclopedia of biostatistics
14. Greene N (1986) Environment mapping and other applications of world projections. IEEE Comput Graph Appl 6(11):21–29
15. Hašan M, Křivánek J, Walter B, Bala K (2009) Virtual spherical lights for many-light rendering of glossy scenes. In: ACM Transactions on graphics (TOG), vol 28. ACM, p 143
16. Hastings WK (1970) Monte carlo sampling methods using markov chains and their applications. Biometrika 57(1):97–109
17. Hedman P, Karras T, Lehtinen J (2016) Sequential monte carlo instant radiosity. ACM
18. Heidrich W, Seidel HP (1998) View-independent environment maps. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on graphics hardware. ACM, pp 39–ff
19. Hu W, Huang Y, Zhang F, Yuan G, Li W (2014) Ray tracing via gpu rasterization. Vis Comput 30(6-8):697–706
20. Hua BS, Low KL (2015) Guided path tracing using clustered virtual point lights. In: SIGGRAPH Asia 2015 posters. ACM, p 43
21. Jensen HW (1996) Global illumination using photon maps. In: Rendering techniques 96. Springer, pp 21–30
22. Keller A (1997) Instant radiosity. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques. ACM press/addison-wesley publishing co., pp 49–56
23. Lafortune EP, Willems YD (1993) Bi-directional path tracing
24. Mantiuk R, Kim KJ, Rempel AG, Heidrich W (2011) Hdr-vdp-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. In: ACM Transactions on graphics (TOG), vol 30. ACM, p 40
25. McGuire M (2011) Computer graphics archive. http://graphics.cs.williams.edu/data
26. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21(6):1087–1092
27. Nabata K, Iwasaki K, Dobashi Y, Nishita T (2016) An error estimation framework for many-light rendering. In: Computer graphics forum, wiley online library, vol 35, pp 431-439
28. Novák J, Engelhardt T, Dachsbacher C (2011) Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In: Symposium on interactive 3D graphics and games. ACM, pp 119–124
29. Olsson O, Billeter M, Sintorn E, Kämpe V, Assarsson U (2015) More efficient virtual shadow maps for many lights. IEEE Trans Vis Comput Graph 21(6):701–713
30. Reinhard E, Stark M, Shirley P, Ferwerda J (2002) Photographic tone reproduction for digital images. ACM Trans Graph (TOG) 21(3):267–276
31. Ritschel T, Grosch T, Kim MH, Seidel HP, Dachsbacher C, Kautz J (2008) Imperfect shadow maps for efficient computation of indirect illumination. ACM Trans Graph (TOG) 27(5):129
32. Ritschel T, Eisemann E, Ha I, Kim JD, Seidel HP (2011) Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. In: Computer graphics forum, wiley online library, vol 30, pp 2258–2269

33. Ritschel T, Dachsbacher C, Grosch T, Kautz J (2012) The state of the art in interactive global illumination. In: Computer graphics forum, wiley online library, vol 31, pp 160–188
34. Segovia B (2007) Interactive light transport with virtual point lights. These de doctorat en informatique. Université, Lyon 1
35. Simon F, Hanika J, Dachsbacher C (2015) Rich-vpls for improving the versatility of many-light methods. In: Computer graphics forum, wiley online library, vol 34, pp 575–584
36. Thiedemann S, Henrich N, Grosch T, Müller S (2011) Voxel-based global illumination. In: Symposium on interactive 3D graphics and games. ACM, pp 103–110
37. Veach E (1997) Robust monte carlo methods for light transport simulation. Stanford University, PhD thesis
38. Walter B, Fernandez S, Arbree A, Bala K, Donikian M, Greenberg DP (2005) Lightcuts: a scalable approach to illumination. In: ACM Transactions on graphics (TOG), vol 24. ACM, pp 1098–1107
39. Walter B, Arbree A, Bala K, Greenberg DP (2006) Multidimensional lightcuts. In: ACM Transactions on graphics (TOG), vol 25. ACM, pp 1081–1088
40. Williams L (1978) Casting curved shadows on curved surfaces. In: ACM Siggraph computer graphics, vol 12. ACM, pp 270–274

Djihane Babahenini is a Ph.D. student in computer graphics at the university of Biskra working under the supervision of Professor Mohamed Chaouki Babahenini and Professor Kadi Bouatouch. She is a member of real-time rendering group at LESIA Laboratory. She received her master's degree in rendering image from the University of Biskra under the guidance of Professor Mohamed Chaouki Babahenini. Her main research interests are GPU based rendering and visualization.

Adrien Gruson received his Ph.D. in 2015 at the University of Rennes 1 under the supervision of Prof. Kadi Bouatouch and Remi Cozot. Now, he is a post-doc researcher at the University of Tokyo in the Computer Graphics Group headed by Prof. Toshiya Hachisuka. His research interests include physically based rendering with robust rendering techniques. He is also a member of Japanese-French Laboratory of Informatics (JFLI).



Mohamed Chaouki Babahenini is a researcher and head of real-time rendering group at LESIA Laboratory, he is also an associate professor at the Department of Computer science of the Biskra University in Algeria, where he received a Ph.D. in 2006. His current research interests are real-time rendering, 3D reconstruction, point-based rendering and data mining. He has co-authored many papers in these fields.

Kadi Bouatouch is working on global illumination, lighting simulation for complex environments, GPU based rendering and computer vision. He is currently Professor at the university of Rennes 1 (France) and researcher at IRISA Rennes (Institut de Recherche en Informatique et Systèmes Aléatoires). He is the head of the FRVSense team at IRISA. He was/is member of the program committee of several conferences and workshops and referee for several Computer Graphics journals. He also acted as a referee for many conferences and workshops. He is an associate editor for the Visual Computer Journal.