Real-Time Importance Sampling of Dynamic Environment Maps
Heqi Lu, Romain Pacanowski, Xavier Granier

To cite this version:

HAL Id: hal-00803998
https://inria.hal.science/hal-00803998
Submitted on 4 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Real-Time Importance Sampling of Dynamic Environment Maps

H. Lu† R. Pacanowski‡ X. Granier§
Inria - Univ. Bordeaux CNRS, LP2N IOGS, LP2N
1. Inria Bordeaux Sud-Ouest - LP2N (Univ. Bordeaux, IOGS, CNRS) - LaBRI (Univ. Bordeaux, CNRS)

Abstract
We introduce a simple and effective technique for light-based importance sampling of dynamic environment maps based on the formalism of Multiple Importance Sampling (MIS). The core idea is to balance per pixel the number of samples selected on each cube map face according to a quick and conservative evaluation of the lighting contribution: this increases the number of effective samples. In order to be suitable for dynamically generated or captured HDR environment maps, everything is computed on-line for each frame without any global preprocessing. Our results illustrate that the low number of required samples combined with a full-GPU implementation lead to real-time performance with improved visual quality. Finally, we illustrate that our MIS formalism can be easily extended to other strategies such as BRDF importance sampling.

1. Motivation
Captured HDR environment maps participate, as natural light sources, to the realism of synthetic images. The resulting direct illumination at position $p$ with normal $n$ is:

$$L(p, o) = \int_\Omega s(p, n, o, \omega) d\omega$$

with

$$s(p, n, o, \omega) = \rho(o, \omega) \langle n, \omega \rangle V(p, \omega) L(\omega)$$

where $\Omega$ is the full unit sphere, $\rho(o, \omega)$ is the reflectance function (BRDF), $\langle,\rangle$ the positive clamped dot product operator (aka cosine factor), $V(p, \omega)$ is the visibility function (we will omit it in this paper) and, $L(\omega)$ represents the radiance emanating from the direction $\omega$. The size of the texture representing the environment map prohibits achieving real-time results with a brute force approach. This becomes even more obvious if $L(\omega)$ is dynamic (i.e., on-line generated or acquired [HSK*05]).

Precomputed Radiance Transfer (PRT) techniques (e.g., [WTL06]) improve rendering performance and integrate complex BRDFs and inter-reflections. However, the required memory to store the precomputed data can quickly become a bottleneck. Furthermore, the costly precomputation time prevents using PRT with dynamic environment maps [HSK*05]. Another trend is to use the general Monte-Carlo framework where the reflected radiance $L(p, o)$ is approximated by $N_s$ samples:

$$L(p, o) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{s(p, n, o, \omega_i)}{pdf(\omega_i)}$$

where $pdf(\omega_i)$ is the Probability Density Function (PDF). The closer it matches the numerator of Equation (1) the
lower the number of samples \( N_i \) will be required to converge to the solution.

In this paper, we focus on light importance sampling of dynamic environment maps (i.e., pdf\( (\omega) \)) proportional to \( L(\omega) \). More precisely, we introduce a real-time GPU importance sampling technique that recomputes for each frame a tabulated version of the Cumulative Distribution Function (CDF) of an environment map. To generate the light samples \( (\omega_i) \), the CDF is inverted through a binary search before the shading pass. Our work can be applied to any kind of environment map (spherical, hemispherical, longitude-latitude) but is particularly interesting for cube maps that are widely used for real-time applications. Furthermore, to improve rendering performance, we also introduce an unbiased Monte-Carlo estimator that:

- limits the number of useless light samples that would have been null due to the cosine factor
- reduces popping artifacts that occur when using a low number of samples (this property is particularly important when using time-dependent light sources)
- integrates easily with Multiple Importance Sampling (MIS).

In the context of light importance sampling, many techniques (e.g., [ODJ04]) for static environment maps have been introduced. However, their high computational cost make them unsuitable for dynamic lighting case. One notable exception is the work from Havran et al. [HSK\textsuperscript{*}05] which generates samples on the CPU. However, contrary to their work, we do not precompute a set of visible light sources for each shaded point \( p \). Our approach computes dynamically a set of light samples according to the light distribution of the environment map and selects some of them according to the pixel normal: it is therefore more suited for dynamic environment maps.

2. Fast and Continuous Sampling on the GPU

2.1. Monte-Carlo Estimator

When using cube maps, one needs to distribute the \( N_s \) samples between the six faces of the cube. Let \( \mu_f \) be the proportion of samples used for the face \( f \) (i.e., \( N_f = \mu_f N_s \)), and pdf\( f(\omega|f) \) the probability density function to select direction \( \omega \) on the face \( f \). Then, Equation (1) can be rewritten in a MIS formalism [Veal98]:

\[
L(p, \omega) \approx \frac{1}{N_s} \sum_{f=1}^{6} \frac{\sum_{i=1}^{N_f} s(p, n, \omega_i, \omega)}{\sum_{i=1}^{6} \mu_f \text{pdf}_f(\omega_i|f)}.
\] (2)

In this paper, we target GPU importance sampling for dynamic environment maps. For this purpose, the cumulative distribution function of each cube face need to be computed at each frame. We thus use tabulated CDFs computed using optimized versions for GPU of the prefix sum [HSO07] on each face. The supplemental material provides a complete derivation on how to compute the associated pdf.

An obvious choice is to distribute uniformly the samples across the faces (i.e., \( \mu_f = 1/6 \)). Unfortunately, depending on the orientation of \( n \), some samples may be behind the point and will be canceled by the cosine factor. A drastic example is when a very bright source (such as the sun) is behind the point \( p \): many useless samples will still be generated on the face the sun belongs to. To prevent this behavior, one solution is to integrate the cosine factor \( (\omega, \omega) \) into the PDF at the price of computing new CDFs that depends also on the normal \( n \). However, this becomes too costly in terms of memory and computation. Instead, we introduce a conservative, simple and dynamic strategy where \( \mu_f \) depends on the normal \( n \) and is computed as follow:

\[
\mu_f(n) = \frac{F_f(n)}{\sum_{i} F_f(n_i)} \text{ with } F_f(n) = \sum_{c_{i} \in f}(n, c_{i}) \] (3)

where \( c_{i} \) are normalized vertexes of the cube map face \( f \). \( F_f(n) \) can be seen as a pseudo form factor (point to face). Equation (3) balances dynamically for \( p \) the number of samples on each face according to its importance.

To prevent popping artefacts when the number of samples is low and changing between neighboring normals, we use a floating point \( N_f \) and introduce a new weight \( \beta_f \) for each sample. By combining Equation (2) and Equation (3) our estimator becomes:

\[
L(p, \omega) \approx \frac{1}{N_s} \sum_{f=1}^{6} \sum_{i=1}^{N_f} \beta_f s(p, n, \omega_i, \omega) \text{pdf}_f(\omega_i|f) \] (4)

with \( \beta_f = \begin{cases} 1 & \text{if } i \leq N_f(n) \\ \text{frac}(N_f(n)) & \text{else} \end{cases} \) (5)

where frac returns the fractional portion. Since pdf\( f(\omega|f) = 0 \) if \( \omega \notin f \), the sum in the denominator in Equation 2 disappears. Thanks to the use of the weight \( \beta_f \), the last sample is introduced progressively.

As demonstrated in the results section, compared to the classical approach, our strategy improves the frame rate because it limits the generation of useless light samples. Based on MIS formalism, we can easily integrate other strategies. We illustrate this by using BRDF importance sampling pdf\( B \). For the balance heuristic, half samples are used for light importance sampling, half for BRDF. Therefore, we use \( N_f = \mu_f(n) N_s/2 \) for each face and \( N_B = N_s/2 \) for the BRDF. This leads to the following estimator:

\[
L(p, \omega) \approx \frac{1}{N_s} \sum_{f=1}^{6} \sum_{i=1}^{N_f} \beta_f s(p, n, \omega_i, \omega) \left( \frac{1}{2} \text{pdf}_f(\omega_i|f) + \frac{1}{2} \text{pdf}_B(\omega) \right) \] (6)

and

\[
\text{pdf}_f = \frac{s(p, n, \omega_i, \omega)}{\frac{1}{2} \sum_{f} \mu_f(n) \text{pdf}_f(\omega_i|f) + \frac{1}{2} \text{pdf}_B(\omega)}. \] 2.2. GPU implementation

The rendering process for each frame is organized as presented in Algorithm 1. The whole process starts by an early
GBuffer pass (line 4) and ends by the final tone mapping (lines 15 and 16). In-between, we compute the tabulated CDF (lines 5 to 11), then we generate the light samples (line 10) and compute the shading (line 13).

We compute the 2D CDF by using the inversion method [PHO04] and store it on GPU. More precisely, a 1D CDF (CDF(\(u\))) and a 2D CDF (CDF(\(v(\mu)\))) are computed using parallel prefix sum [HSD07] and stored as floating point CDF (CDF(\(v\))) and compute the shading (line 13).

High performance in CUDA requires thread coherency, memory coalescing and branch consistency. We have therefore designed the Algorithm 2 for the shading step (cf. line 13) to reach the best performance of our different implementations. After shading, the computation of the average luminance is done using a reduction operation [RAH07] for the tone mapping step.

Finally, since our per pixel Monte-Carlo estimator is unbiased, our approach does not introduce any bias for a given pixel. However, for efficiency reason, we use the same pre-computed random sequence for all the pixels. This introduces a spatial bias between neighbor pixels but it has the advantage to limit disturbing noise in the final image. A possible extension to reduce the spatial bias would be to use interleaved sampling.

3. Results

The companion video and all results presented in this paper were computed on a 2.67 GHz PC with 6 GB of RAM and a NVIDIA GTX 680 graphics card. We implemented our system using DirectX, CUDA and the Thrust Library. The static environment map used in Figure 2 has a resolution of \(256 \times 256 \times 6\) pixels. For the dynamic environment map used in Figure 1, we use the available 67 frames from the capture made during a full day by Stumpfel et al. [STJ∗04]. Before using it, the only preprocess we apply on the captured images is a re-parametrization (from hemisphere to cube) to obtain \(256 \times 256 \times 6\) pixels images. We also fill with an average color the pixels belonging to the roof where the acquisition device was placed. For all results, we achieve real-time framerates with dragon models, 369K (resp. 100K) polygons in Figure 2 (resp. Figure 3) as well as the model (169K polygons) used in Figure 2.

Figure 1 shows a set of images with a time-varying environment map captured from daylight to night. The central images show where the light samples are located for a given pixel. Notice how our technique avoids generating useless samples on the sun for the highlighted pixel. Moreover, the companion video demonstrates our real-time performance as well as the temporal coherence of our Monte-Carlo estimator when changing the view direction or the lighting or even the BRDF parameters.

Figure 2 presents a comparison between a classical light-based importance sampling and our approach. As shown in the picture, both images have almost the same quality when compared to a reference solution, but our technique uses three times less samples and is consequently faster. Finally, Figure 3 demonstrates the convergence speed and the quality obtained with our MIS technique. As shown by the Lab difference between our results and a reference solution, even

---

Algorithm 1 Steps to render one frame. We use GC when using GPU computing and GS when using GPU shader. We indicate the for-loop that are parallelized using CUDA threads with the keyword "do in parallel".

```
1: procedure RENDER_FRAME
2:      GEO2D ⇒ Buffer for vertex positions and normals
3:      LS[1..6] ⇒ Buffers of light samples for each face f
4:      GBUFFER_PASS(GEO2D) ⇒ GS
5:      for each face f of the cube environment map do
6:         compute_luminance_per_pixel ⇒ GC
7:         compute_prefix_sum_by_inclusive_scan ⇒ GC
8:         compute,CDF,1D, u ⇒ GC
9:         compute,CDF,2D,v,knowing,u ⇒ GC
10:        LS[1|f|]=generate_light_samples(N_s/2) ⇒ GC
11:     end for
12:     for each pixel (p,n) ∈ GEO2D do in parallel
13:         shade(o,p,n,N)_s ⇒ GC - Algo. 2
14:     end for
15:     compute_average_intensity(frame) ⇒ GC
16:     tone_mappening(frame) ⇒ GS
17: end procedure
```

Algorithm 2 Shading procedure with our dynamic balancing technique. To take advantage of the CUDA architecture, we have integrated the BRDF sampling pass as a 7th pass and used a fixed maximal number of iterations (cf. Line 6).

```
1: procedure shade(o,p,n,N_s) ⇒ in parallel for each pixel p
2:      N_s[1..6] = Samples_Per_Face(N_s/2) ⇒ Equation 3
3:      N_s[7] = N_s/2 ⇒ number of BRDF samples
4:      for each step f=1..7 do
5:         for each i ∈ [0,N_s/2] do
6:            break when i ≥ N_f[1]
7:            compute β_i ⇒ Equation 5
8:            if f < 7 then
9:               sample = LS[1|f|][i] ⇒ see Algo. 1
10:          else
11:              sample = BRDF_sampling ⇒ Equation 6
12:          end if
13:      end for
14:      end for
15:     L_i(p,o) = L_i(p,o)/N_s
16: end procedure
```
with 60 samples per pixel our method achieves visual plausible results in real-time (198 fps).

4. Conclusion and Future Work

In this paper we have introduced an improved Monte-Carlo estimator for light-based importance sampling of dynamic environment maps. Our pixel-based technique increases the number of effective samples and is faster for the same quality compared to a uniform distribution of samples on each face. Furthermore, our technique handles efficiently dynamic and time-varying environment maps. Based on Multiple Importance Sampling formalism, it can be easily combined with other sampling strategies. For future work, we would like to incorporate a more robust balancing scheme to distribute light samples and also to introduce visibility and indirect lighting effects.

Acknowledgments

Heqi Lu’s PhD scholarship is funded by the Région Aquitaine. This research has been supported by the ALTA project (ANR-11-BS02-006).

References


