Interactive Monte-Carlo Ray-Tracing Upsampling

Malik Boughida Thibault Groueix Tamy Boubekeur

LTCI, CNRS, Telecom ParisTech, Paris-Saclay University

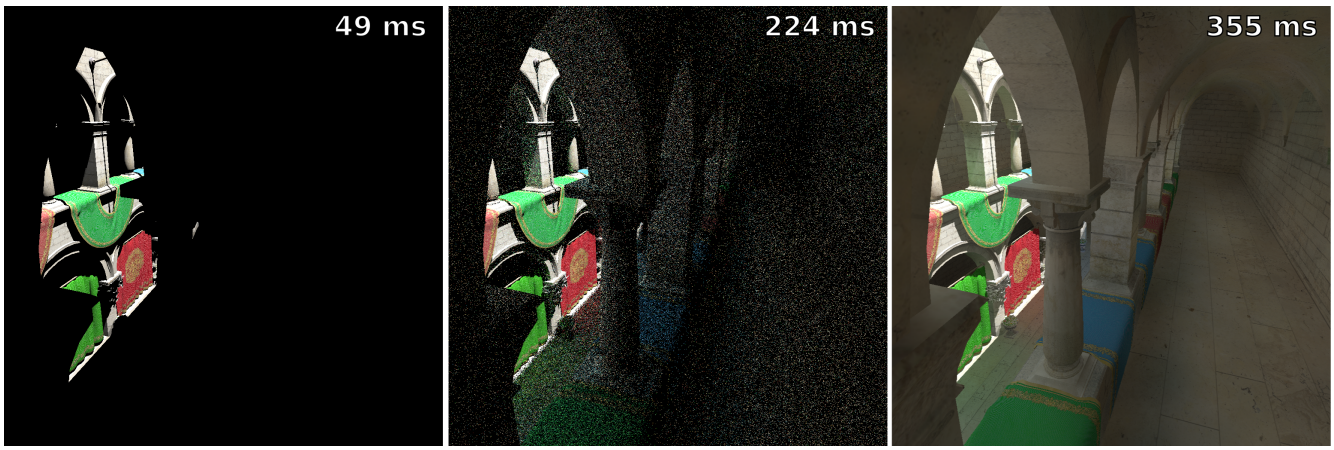

Figure 1: *Sponza scene (62267 polygons), rendered at* 1024×1024 *image resolution.* ***Left:*** *Direct lighting only.* ***Middle:*** *Path tracing 1 spp (sample per pixel).* ***Right:*** *our method, using a* 16×16 *downsampled indirect lighting solution computed at* 100 *spp. We can observe color bleeding effect, the lack of high-frequency noise, and multiple light bounces.*

Abstract

We propose a practical method to approximate global illumination at interactive framerates for dynamic scenes. We address multi-bounce, visibility-aware indirect lighting, for diffuse to moderately glossy materials, relying on GPU-accelerated ray-tracing for this purpose. While Monte-Carlo ray-tracing algorithms offer unbiased results, they produce images which are, under interactive constraints, extremely noisy, even with GPU acceleration. Unfortunately, filtering them to reach visual appeal induces a large kernel, which is not compatible with interactive framerate. We address this problem using a simple downsampling approach. First, we trace indirect paths on a uniformly distributed subset of pixels, decorrelating diffuse and specular components of lighting. Then, we perform a joint bilateral upsampling on both components, taking inspiration from deferred shading by driving this upsampling with a full-resolution G-Buffer. Our solution provides smooth results, does not require any pre-computations, and is both easy to implement and flexible, as it can be used with any generation strategy for indirect rays.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Modern parallel computing architectures such as NVidia's OptiX [PBD*10] give any developer the ability to easily design ray tracing algorithms with real-time performances. For such frameworks, progressive Monte-Carlo rendering [DKHS14] is clearly one of the major applications. Unfortunately, for dynamic scenes, convergence is usually not reached within the authorized delay, and the resulting images contain a significant amount of noise which feels very disturbing from a visual point-of-view. Indeed, in the literature, real-time global illumination is typically addressed outside the family of ray-tracing algorithms. Most of the existing methods focus on one bounce of indirect lighting, sometimes with only diffuse materials, and often without taking into account indirect visibility. Monte-Carlo ray-tracing is by far the most simple method to overcome these limitations, but suffers from heavy noise.

In this work, we propose a method based on *joint bilateral upsampling* [KCLU07] to approximate multi-bounce global illumination in real-time for completely dynamic scenes and diffuse to moderately glossy materials, without any kind of pre-computation. In such a challenging context, we do not seek for artifact-free rendering, but instead for a visually convincing insight of the indirect lighting contribution, which shall be instrumental for several computer graphics applications, including interactive lighting design, animation previewing and games. Joint bilateral upsampling [KCLU07] takes a low-resolution buffer and a joint high-resolution buffer as inputs, and outputs the high-resolution upscal-

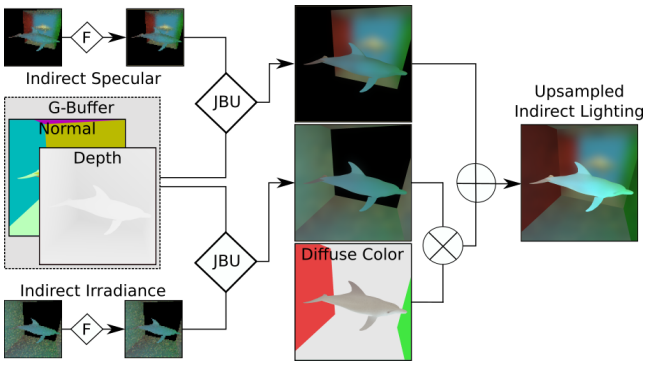

Figure 2: *Overview of our method. JBU: Joint Bilateral Upsampling; F: outlier-removal and/or bilateral Filtering.*

ing of the first one, guided by the second one. This method is quite commonly used in real-time rendering to upsample costly computations, such as ambient occlusion or subsurface scattering. To our knowledge, we propose the first attempt to combine it with interactive Monte-Carlo rendering.

2. Our approach

Our goal is to compute the best approximation of global illumination given a limited amount of time. We assume we have a way to generate the G-Buffer (per-pixel depth, normal, diffuse color and material attributes) and the direct lighting (any method can be used). With what is left of the per-frame time budget, usually we can trace only a few rays per pixel to approximate indirect lighting. Removing the heavy noise produced by this method would require a filter with a very large kernel size, leading to a prohibitive computation cost. Hence, we propose to aggressively downsample indirect lighting by computing it in a low-resolution buffer. With 16×16 downsampling, we can get, for the same delay, 256 times as many indirect rays per (low-resolution) pixel. Then we reconstruct the high-resolution result by performing a *joint bilateral upsampling* that interpolates the data through a bilateral filter. Its complexity is linear with respect to the size of the filtering kernel *in low resolution*. Thus, we basically approximate the effect of a large kernel in high resolution by the use of a small one in low resolution, which leads to much faster computations.

However, care must be taken in order to downsample only low-frequency data. At a visible point, we denote L_i the incoming radiance, L_o the outgoing radiance, and f the BRDF. L_i can be splitted into a direct and an indirect parts: $L_i = L_i^{dir} + L_i^{ind}$, leading to: $L_o = L_o^{dir} + L_o^{ind}$. We can also isolate the diffuse color k_d of the BRDF: $f = \frac{k_d}{\pi} + f^{spec}$. We then decompose: $L_o^{ind} = \frac{k_d}{\pi} E^{ind} + L_o^{ind,spec}$, where E^{ind} is the indirect irradiance. In flat textured lambertian regions, L_o^{ind} has high frequency because of the high frequency of k_d. But E^{ind} still remains low-frequency. The frequency of $L_o^{ind,spec}$, however, may be high if the material is very specular: that is why our algorithm is limited to moderately glossy materials. Thus, we filter separately E^{ind} and $L_o^{ind,spec}$, and only in the end we recompose them with k_d to compute the final filtered L_o^{ind}. With this strategy, we share a lot of indirect lighting information, across a large part of the original image, while preserving textures very well.

Our method can be summarized as follows (see also Figure 2):

1. Compute the G-buffer and the direct lighting buffer L_o^{dir} in high resolution with any method.
2. For a subset of the pixels, on a regular grid, trace N_{samples} random light paths starting from the previously computed world position, with any method. An estimate of E^{ind} and $L_o^{ind,spec}$ is associated to each path and accumulated in two separate buffers.
3. (Optional) Remove outliers. For each pixel, we consider its low-definition neighbors and compute the mean and standard deviation of luminance. If the luminance of the central pixel is distant from the mean by more than twice the standard deviation, we scale its color to reach the mean luminance.
4. (Optional) Filter these two low-resolution buffers, with a bilateral filter using depth and normal buffers.
5. Upscale E^{ind} and $L_o^{ind,spec}$ with a *joint bilateral upsampling*, using depth and normals as joint buffers.
6. Recompose the final high-resolution image by adding direct lighting, the filtered upscaled E^{ind} multiplied by $\frac{k_d}{\pi}$, and the filtered upscaled $L_o^{ind,spec}$.

3. Results, discussion and future work

We implemented our algorithm with the OptiX framework and evaluated it on an NVidia GTX 680 GPU. Figure 1 shows an example of result taken among our numerous experiments. We performed a 16×16 downsampling, and $N_{\text{samples}} = 100$ light path per downsampled pixel. Within 355 ms, we obtain a nice approximation of global illumination, with color bleeding, indirect shadows, and multiple bounces. Although we do not perform anti-aliasing, cannot handle properly highly specular materials (e.g., like mirrors) and still observe a low-frequency noise which is not temporally coherent, our method is very easy to implement, enhances significantly rendered images at little computational cost, handles completely dynamic scenes and does not require any pre-computations. Moreover, it remains orthogonal to progressive rendering: after the first rendering, we can keep tracing rays to get higher definition indirect buffers that get dynamically upsampled. We used simple path tracing for indirect ligthing, but we could of course combine our framework with more advanced Monte-Carlo ray tracing algorithms, such as bidirectional path tracing for instance. As future work, we plan to address low-level optimizations to reach higher framerates as well as seek for ways to reduce the low-frequency noise even more, recovering a form of temporal coherence.

References

[DKHS14] DAVIDOVIČ T., KŘIVÁNEK J., HAŠAN M., SLUSALLEK P.: Progressive light transport simulation on the gpu: Survey and improvements. *ACM Trans. Graph. 33*, 3 (June 2014), 29:1–29:19. 1

[KCLU07] KOPF J., COHEN M. F., LISCHINSKI D., UYTTENDAELE M.: Joint bilateral upsampling. *ACM Trans. Graph. 26*, 3 (July 2007). 1

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: Optix: A general purpose ray tracing engine. *ACM Trans. Graph. 29*, 4 (July 2010), 66:1–66:13. 1