Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

Volume 34 (2015), Number 4

Supplemental Material for Probabilistic Connections for Bidirectional Path Tracing

Stefan Popov, Fredo Durand, Ravi Ramamoorthi, George Drettakis

1. Introduction

In this supplemental material we show additional comparisons to Metropolis methods and we provide the implementation details and exact formulas of the PMF interpolation step.

2. Additional comparisons to Metropolis methods

We compare to more recent Metropolis global illumination methods, since they use non-local information to explore path space, and in particular Kelemen et al. [KSKAC02] and Jakob and Marschner [JM12], as implemented in Mitsuba [Jak10]. For the scenes presented, the results of these methods contain more noise and artifacts than ours in most places. A notable exception is the glass pane in the Apartment image in Fig. 2: this is the case where Manifold Exploration [JM12] works well, and their method contains less noise for these pixels. The rest of the scene however is noisier.

To calibarate the computational times between our system and Mistsuba, we determine the paths/second cost for BDPT in both systems. We then use the ratio as a scaling factor to estimate same computation time: Mitsuba is 2.6 times faster than our system.

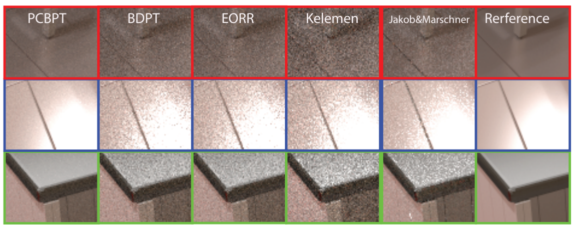

Figure 1: *Same time comparison for 8.4 min for Kitchen scene, insets correspond to Fig. 1 in the main paper. Left to right: PCBPT (97 iterations), BDPT (92 iterations), EORR (102 iterations), [Kelemen et al. 2002], [Jakob and Marschner 2012] and reference. Wall clock times for Metropolis methods 3.2min.*

3. PMF Interpolation Details

At a given cache record at eye vertex z^c, we store $\mathrm{PMF}(z^c, v)$ and $\mathrm{CMF}(z^c, v), v = 0 \dots V-1$, where V is the total number of vertices in all M light paths. For a new eye vertex z during PCPT, we interpolate from the 6 closest cache records $z_i^c, i = 0 \dots 5$ and blend with a uniform distribution. If $\mathcal{C} = \{z_0^c, \dots z_5^c\}$ is the set of selected cache points, n and n_i^c are the normals, and ω_n and ω_i^c the incoming directions at z and z_i^c respectively (please also refer to Fig. 3), weight $w(z_i^c)$ is given as follows:

$$w(z_i^c) = \cos(n, n_i^c)\cos(\omega, \omega_i^c)\frac{d(z, z_i^c)}{\max_{i \in \mathcal{C}} d_i} \tag{1}$$

with d being the distance between two points and $d_i = d(z, z_i^c)$. Conceptually, the value of one element of the interpolated CMF is given as follows:

$$\mathrm{CMF}(z, v) = \frac{\sum_{i=0}^{5} w(z_i^c)\mathrm{CMF}(z_i^c, v) + 0.5\mathcal{U}(v)}{\sum_{i=0}^{5} w(z_i^c) + 0.5} \tag{2}$$

where $\mathcal{U}(v)$ is a uniform CMF; the same interpolation is performed for PMF values. In our tests, we found 0.5 to be the best compromise for blending with $\mathcal{U}$ since lower than 50% blending, gave high intensity speckles while higher than 50% deteriorated results. In practice, Eq. (2) is evaluated lazily during the binary search used for CMF inversion sampling, and the equivalent interpolation is peformed for the PMF once the corresponding light source index is found.

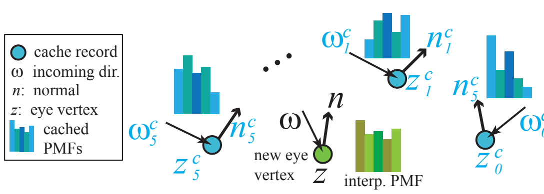

Figure 3: *PMF interpolation: The PMF at the new eye vertex z (green) is interpolated from the cached records (cyan), weighted by the cosines between incoming directions ω and normals n.*

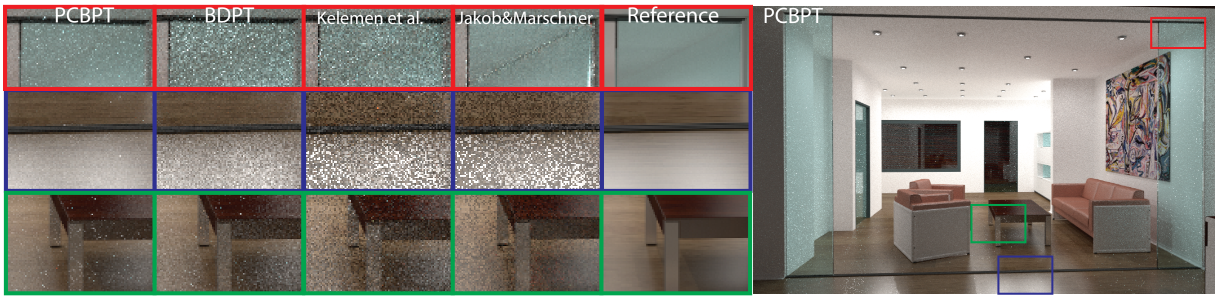

Figure 2: *Same time comparisons for Apartment scene. LeftRight: PCBPT image. Inset results for PCBPT, BDPT, [Kelemen et al. 2002], [Jakob and Marschner 2012] and reference. Rendering time is 12min; wall-clock (unscaled) time for Metropolis methods is 4.6min. PCBPT and BDPT performed 200 iterations each.*

References

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 1

[JM12] JAKOB W., MARSCHNER S.: Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. on Graph. 31*, 4 (2012). 1

[KSKAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the metropolis light transport algorithm. In *Comp. Graph. Forum* (2002), vol. 21. 1