GPU-Optimized Bi-Directional Path Tracing

Denis Bogolepov
Lobachevsky State University of Nizhni Novgorod
603950, Nizhni Novgorod, Russia
denisbogol@gmail.com

Danila Ulyanov
Lobachevsky State University of Nizhni Novgorod
603950, Nizhni Novgorod, Russia
danila-ulyanov@ya.ru

Dmitry Sopin
Lobachevsky State University of Nizhni Novgorod
603950, Nizhni Novgorod, Russia
sopindm@gmail.com

Vadim Turlapov
Lobachevsky State University of Nizhni Novgorod
603950, Nizhni Novgorod, Russia
vadim.turlapov@gmail.com

ABSTRACT

In this paper we present a light-weight modification of bi-directional path tracing algorithm that is optimized for massively parallel architectures with limited memory, like GPU. The amount of computations performed by the algorithm is still comparable to unidirectional path tracing. Though modified algorithm preserves some benefits of general bi-directional path tracing and handles indirect illumination and caustics quite efficiently.

Keywords

Realistic Image Synthesis, Interactive Global Illumination, Bi-Directional Path Tracing, GPU, GPGPU.

1. INTRODUCTION

Path tracing is an image synthesis algorithm based on the numerical solution of the rendering equation. This technique allows solving all rendering problems that assume geometric optics, such as soft shadows, indirect lighting, caustics, motion blur, and depth of field. Path tracing provides superior quality visuals compared to rasterization rendering but is also very computationally expensive. Because of the stochastic nature, the image is subject to some variance which is visible as noise. The main contributors to noise are indirect illumination and caustics.

A more general rendering algorithm is *bi-directional path tracing* (BPT) that was independently proposed by Lafortune [Laf93] and Veach [Vea94]. The basic idea is that paths are traced at the same time from a light source and from the camera aperture. All the vertices on the respective paths are then connected using shadow rays and appropriate contributions are added to the measurement of radiance through the corresponding image pixel. BPT handles caustics and indirect illumination effects far more efficiently than ordinary (unidirectional) path tracing (PT).

Despite the fact that the BPT can be implemented on the GPU, it is quite resource intensive. The memory consumption is significantly higher than for ordinary PT (more than 20x for each sample) and depends on the maximum path length [Ant11]. At the same time, effective GPU utilization is achieved for several tens of thousands of concurrent threads that require large amount of onboard memory. However, current GPUs have limited memory resources that should be used sparingly to store 3D geometric models, accelerating structure, texture maps, and other data.

Therefore, we propose a light-weight modification of BPT. The amount of computations performed by the algorithm is still comparable to ordinary PT. Though modified algorithm preserves some benefits of BPT and handles indirect illumination and caustics quite efficiently.

2. THE MEASUREMENT EQUATION

The total radiance M_j measured by the sensor (pixel) j is computed by integrating the incoming radiance L over both the film plane I and all of the surfaces of the scene M:

$$M_j = \int_{I\times M} W_j(\mathbf{x}_1 \to \mathbf{x}_0) L(\mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0) dA_{\mathbf{x}_0} dA_{\mathbf{x}_1}$$

In the equation above, W_j is *response function* that depends on a pixel filter (and/or other factors), and G is *geometry term* defined as

$$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0) = V(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0) \frac{\left(N_{\mathbf{x}_0} \cdot \overrightarrow{\mathbf{x}_0\mathbf{x}_1}\right)\left(N_{\mathbf{x}_1} \cdot \overrightarrow{\mathbf{x}_1\mathbf{x}_0}\right)}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2}$$

Here V is the *visibility function* ($V = 1$ if $\mathbf{x}_0$ and $\mathbf{x}_1$ are mutually visible, and $V = 0$ otherwise).

The total amount of outgoing radiance, $L(\mathbf{x}_1 \to \mathbf{x}_0)$, can be computed as the sum of emitted radiance L_e plus reflected radiance L_r:

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + L_r(\mathbf{x}_1 \to \mathbf{x}_0) =$$

$$L_e + \int_A f_r(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) L(\mathbf{x}_2 \to \mathbf{x}_1) G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1) dA_{\mathbf{x}_2}$$

Here f_r is the *bi-directional scattering distribution function* (BSDF), which describes the reflectance and transmittance properties of a surface.

The last equation is called the *rendering equation* and formulates the law of conservation of light energy in 3D scene. By recursively substituting L on the right side of the rendering equation by the complete right side, we get:

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \sum_{k=2}^{\infty} \int_{M^{k-1}} l(\mathbf{x}_0 \mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_k) dA_{\mathbf{x}_2} \ldots dA_{\mathbf{x}_k}$$

Here $l\colon \bigcup_{i=2}^{\infty} M^{i+1} \to \mathbb{R}$ is the *radiance flow function* defined as

$$l(\mathbf{x}_0 \ldots \mathbf{x}_k) = \prod_{i=1}^{k-1} f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \cdot L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})$$

This function describes the fraction of radiance from the light source that arrives at the sensor after all of the scattering at vertices between them.

A *light transport path* $\mathbf{X}^k$ is any sequence of surface points $\mathbf{x}_0 \ldots \mathbf{x}_k \in I \times M^k$ of which the first point $\mathbf{x}_0$ lies on the film plane I and the remaining lie on the scene surfaces M. The set of all paths of all lengths is called the *path space* and is written as Ω.

The measurement equation can also be written over unified path space Ω. The generalized measurement function, $f_j\colon \Omega \to \mathbb{R}$, is defined as

$$f_j(\mathbf{x}_0 \ldots \mathbf{x}_k) = W_j(\mathbf{x}_1 \to \mathbf{x}_0) \cdot \prod_{i=0}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \cdot \prod_{i=1}^{k-1} f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) \cdot L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})$$

Thus, the measurement of radiance through sensor (pixel) j can be expressed over unified path space as

$$M_j = \int_{\Omega} f_j(\mathbf{X}) d\Omega(\mathbf{X})$$

$$d\Omega(\mathbf{x}_0 \ldots \mathbf{x}_k) = dA_{\mathbf{x}_0} \cdot dA_{\mathbf{x}_1} \cdot \ldots \cdot dA_{\mathbf{x}_k}$$

This equation formulates the fundamental problem that a global illumination algorithm must solve. To estimate the high-dimensional integral M_j the Monte-Carlo methods are used.

3. GPU-OPTIMIZED BPT

Instant bidirectional path tracing (IBPT) is unbiased rendering algorithm which generates an image in two independent passes (can be executed in any order):

- *Path tracing pass* (PT). Tracing a path starting at the eye (camera lens). The path is extended until it is terminated with a certain probability by Russian roulette. Each vertex $\mathbf{y}_i$ is connected to a random point $\mathbf{z}$ on a light source to form the *explicit view* path $\mathbf{y}_0 \ldots \mathbf{y}_i\mathbf{z}$. If the path accidentally hits a light source at point $\mathbf{y}_i$, then the sequence $\mathbf{y}_0 \ldots \mathbf{y}_i$ forms *implicit view* path.
- *Light tracing pass* (LT). Tracing a path starting at a selected light source. Each vertex $\mathbf{z}_i$ is directly connected to a random point $\mathbf{y}$ on the camera lens to form the *explicit light* path $\mathbf{z}_0 \ldots \mathbf{z}_i\mathbf{y}$. If the path accidentally hits the camera lens at point $\mathbf{z}_i$, then the sequence $\mathbf{z}_0 \ldots \mathbf{z}_i$ forms *implicit light* path.

Thus, in IBPT algorithm each path $\mathbf{X}^k = \mathbf{x}_0 \ldots \mathbf{x}_k$ can be constructed in four different ways as either an explicit or implicit path on the LT and PT stages. It should be noted that the "implicit light" strategy can be realized only when using the camera with finite aperture (that was implemented in this study).

3.1. Compute Path Contributions

In order to compute the Monte-Carlo contribution of path $\mathbf{X}^k = \mathbf{x}_0 \ldots \mathbf{x}_k$, the probability density of sampling this path needs to be expressed with respect to unit surface area. Further the following notation is used to write the densities. Let $p_A(\mathbf{x}_i)$ be a PDF of sampling $\mathbf{x}_i$, measured with respect to unit surface area. The $p_{\omega^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i)$ and $p_{\omega^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1})$ are the PDFs of sampling $\mathbf{x}_i$ from points $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$, respectively, measured with respect to projected solid angle. These PDFs relate according to

$$p_A(\mathbf{x}_i) = G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) p_{\omega^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) = G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) p_{\omega^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i)$$

The probability density of generating a light transport path can be expressed as the product of the sampling densities for the individual path vertices.

PDF of *implicit view* path $\mathbf{X}^k = \mathbf{x}_0 \ldots \mathbf{x}_k$ equals

$$p_{V_I}(\mathbf{X}^k) = p_I(\mathbf{x}_0) \cdot \prod_{i=1}^{k} p_A(\mathbf{x}_i) = p_I(\mathbf{x}_0) p_A(\mathbf{x}_1) \cdot \prod_{i=1}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) p_{\omega^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})$$

PDF of *explicit view* path $\mathbf{X}^k = \mathbf{x}_0 \ldots \mathbf{x}_k$ equals

$$p_{V_E}(\mathbf{X}^k) = p_I(\mathbf{x}_0) \cdot \prod_{i=1}^{k} p_A(\mathbf{x}_i) = p_I(\mathbf{x}_0) p_A(\mathbf{x}_1) \cdot \prod_{i=1}^{k-2} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) p_{\omega^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1}) \cdot p_A(\mathbf{x}_k)$$

Here $p_I(\mathbf{x}_0)$ and $p_A(\mathbf{x}_1)$ are the probability densities of sampling the vertices $\mathbf{x}_0$ and $\mathbf{x}_1$, respectively. The specific choice of such PDFs is defined by camera model used. For example, finite aperture lens camera is described in [Url01].

PDF of *implicit light* path $\mathbf{X}^k = \mathbf{x}_0 \ldots \mathbf{x}_k$ equals

$$p_{L_I}(\mathbf{X}^k) = \prod_{i=0}^{k} p_A(\mathbf{x}_i)$$
$$= \prod_{i=0}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) p_{\omega^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) \cdot p_A(\mathbf{x}_k)$$

PDF of *explicit light* path $\mathbf{X}^k = \mathbf{x}_0 \dots \mathbf{x}_k$ equals

$$p_{L_E}(\mathbf{X}^k) = p_I(\mathbf{x}_0) \cdot \prod_{i=1}^{k} p_A(\mathbf{x}_i) = p_I(\mathbf{x}_0)$$
$$\cdot \prod_{i=1}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) p_{\omega^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) \cdot p_A(\mathbf{x}_k)$$

For compactness, let $\widehat{W}_j$ denote the *modified sensor response function*:

$$\widehat{W}_j(\mathbf{x}_1 \to \mathbf{x}_0) = \frac{W_j(\mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0)}{p_I(\mathbf{x}_0) p_A(\mathbf{x}_1)}$$

When points on the camera lens and view plane are sampled uniformly, $\widehat{W}_j(\mathbf{x}_1 \to \mathbf{x}_0) = C$ for the simple finite aperture camera model (constant C determines the total sensor sensitivity) [Url01].

After canceling out the common factors, the Monte-Carlo contribution of *implicit view* path $\mathbf{X}^k = \mathbf{x}_0 \dots \mathbf{x}_k$ equals

$$\frac{f_j(\mathbf{X}^k)}{p(\mathbf{X}^k)} = \widehat{W}_j(\mathbf{x}_1 \to \mathbf{x}_0) \prod_{i=1}^{k-1} \frac{f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})}{p_{\omega^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})}$$
$$\cdot L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})$$

The Monte-Carlo contribution of *explicit view* path $\mathbf{X}^k = \mathbf{x}_0 \dots \mathbf{x}_k$ equals

$$\frac{f_j(\mathbf{X}^k)}{p(\mathbf{X}^k)} = \widehat{W}_j(\mathbf{x}_1 \to \mathbf{x}_0) \prod_{i=1}^{k-2} \frac{f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})}{p_{\omega^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})}$$
$$\cdot \frac{f_r(\mathbf{x}_k \to \mathbf{x}_{k-1} \to \mathbf{x}_{k-2}) G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1})}{p_A(\mathbf{x}_k)} \cdot L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})$$

The Monte-Carlo contribution of *implicit light* path $\mathbf{X}^k = \mathbf{x}_0 \dots \mathbf{x}_k$ equals

$$\frac{f_j(\mathbf{X}^k)}{p(\mathbf{X}^k)} = W_j(\mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0)$$
$$\cdot \prod_{i=1}^{k-1} \frac{f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})}{p_{\omega^\perp}(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i)} \cdot \frac{L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})}{p_A(\mathbf{x}_k) p_{\omega^\perp}(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)}$$

The Monte-Carlo contribution of *explicit light* path $\mathbf{X}^k = \mathbf{x}_0 \dots \mathbf{x}_k$ equals

$$\frac{f_j(\mathbf{X}^k)}{p(\mathbf{X}^k)} = \frac{W_j(\mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0)}{p_I(\mathbf{x}_0)} f_r(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0)$$
$$\cdot \prod_{i=2}^{k-1} \frac{f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})}{p_{\omega^\perp}(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i)} \cdot \frac{L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})}{p_A(\mathbf{x}_k) p_{\omega^\perp}(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)}$$

3.2. Multiple Importance Sampling (MIS)

If there are multiple sampling strategies to generate light transport paths, the samples can be combined in a single unbiased estimator by using MIS.

We propose an efficient procedure for computing the power heuristic weights. Let us consider the inverse weight for *explicit view* path $\mathbf{X}^k = \mathbf{x}_0 \dots \mathbf{x}_k$:

$$\frac{1}{w_{V_E}(\mathbf{X}^k)} = 1 + \frac{p_{V_I}(\mathbf{X}^k)^\beta}{p_{V_E}(\mathbf{X}^k)^\beta} + \frac{p_{L_E}(\mathbf{X}^k)^\beta}{p_{V_E}(\mathbf{X}^k)^\beta} + \frac{p_{L_I}(\mathbf{X}^k)^\beta}{p_{V_E}(\mathbf{X}^k)^\beta}$$
$$= 1 + \frac{p_{V_I}(\mathbf{X}^k)^\beta}{p_{V_E}(\mathbf{X}^k)^\beta} + \frac{p_{L_E}(\mathbf{X}^k)^\beta}{p_{V_E}(\mathbf{X}^k)^\beta} \left(1 + \frac{p_{L_I}(\mathbf{X}^k)^\beta}{p_{L_E}(\mathbf{X}^k)^\beta}\right)$$

Both explicit and implicit view paths are sampled in the same way except for the last vertex $\mathbf{x}_k$. Thus, the common factors cancel out:

$$\frac{p_{V_I}(\mathbf{X}^k)}{p_{V_E}(\mathbf{X}^k)} = \frac{p_{\omega^\perp}(\mathbf{x}_{k-1} \to \mathbf{x}_k) G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)}{p_A(\mathbf{x}_k)}$$

Similar, explicit and implicit light paths differ in how the first vertex $\mathbf{x}_0$ is generated:

$$\frac{p_{L_I}(\mathbf{X}^k)}{p_{L_E}(\mathbf{X}^k)} = \frac{p_{\omega^\perp}(\mathbf{x}_0 \leftarrow \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)}{p_I(\mathbf{x}_0)}$$

The ratio of the densities of generating explicit light and view paths can be written in the following way:

$$\frac{p_{L_E}(\mathbf{X}^k)}{p_{V_E}(\mathbf{X}^k)} = \frac{1}{p_A(\mathbf{x}_1)} \cdot \frac{1}{p_{\omega^\perp}(\mathbf{x}_1 \to \mathbf{x}_2)}$$
$$\cdot \frac{p_{\omega^\perp}(\mathbf{x}_1 \leftarrow \mathbf{x}_2)}{p_{\omega^\perp}(\mathbf{x}_2 \to \mathbf{x}_3)} \cdot \dots \cdot \frac{p_{\omega^\perp}(\mathbf{x}_{k-3} \leftarrow \mathbf{x}_{k-2})}{p_{\omega^\perp}(\mathbf{x}_{k-2} \to \mathbf{x}_{k-1})}$$
$$\cdot p_{\omega^\perp}(\mathbf{x}_{k-2} \leftarrow \mathbf{x}_{k-1})$$
$$\cdot p_{\omega^\perp}(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k) G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1}) = \prod_{i=0}^{k} s_i$$

Obviously, this expression can be computed during the incremental path construction. At the i-th vertex ($i = 0 \dots k$) the following scalar variable needs to be accumulated:

$$s_0 = p_A^{-1}(\mathbf{x}_1)$$
$$s_1 = p_{\omega^\perp}^{-1}(\mathbf{x}_1 \to \mathbf{x}_2)$$
$$s_i = \frac{p_{\omega^\perp}(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i)}{p_{\omega^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})}$$
$$s_{k-1} = p_{\omega^\perp}(\mathbf{x}_{k-2} \leftarrow \mathbf{x}_{k-1})$$
$$s_k = G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1}) p_{\omega^\perp}(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k)$$

Thus, to compute the inverse MIS weight of explicit view path we only need the accumulated value s for the current path vertex. MIS weights for other light transport paths are computed similarly.

3.3. GPU-Specific Features

IBPT can be effectively implemented on massively parallel GPUs due to the following features:

1. Both render passes generate intermediate images independently. Thus, computations can be easily parallelized. In particular, LT and PT passes may be executed on different GPUs (to get the final result two images must be simply summed up).
2. To process paths of any length a constant amount of memory is needed. Therefore, the GPU can be

utilized efficiently by running a large number of light-weight threads. Furthermore, this eliminates restrictions on the unbiased image synthesis.

3. The path connection stage is eliminated (requires information on all vertices of each path).
4. The optimal MIS weights can be computed using only one floating variable per path regardless of its length.

4. RESULTS

The IBPT allows improving the convergence rate (as compared to "ordinary" PT) for regions with indirect illumination problems. Figure 1 shows an example of scene with a significant amount of caustics, which would have taken a long time to render using PT.

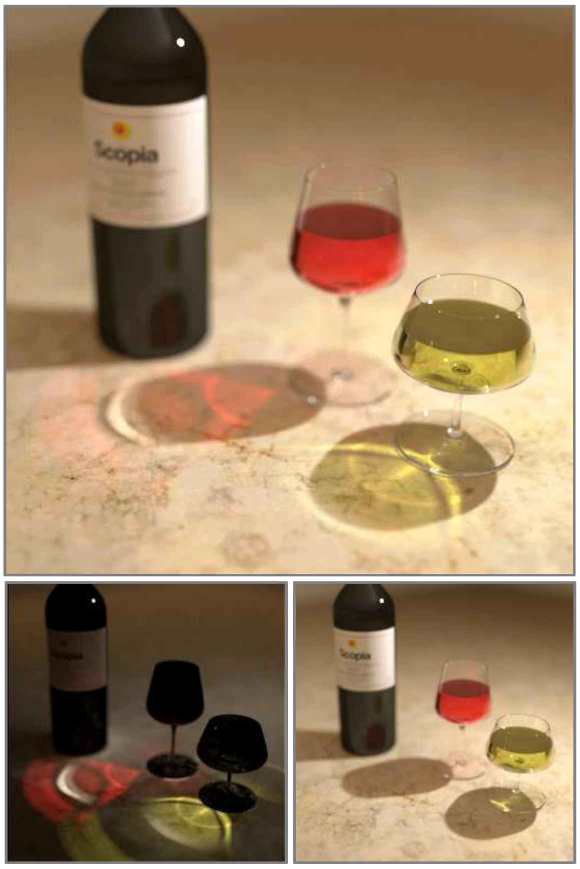

Figure 1. Reference image and weighted contributions of LT and PT sampling techniques

The IBPT was compared with the unidirectional MIS PT (combines explicit and implicit view paths). For both rendering algorithms the speed of convergence to the reference image was measured (the difference between images was computed using L2 norm). This comparison is appropriate, since the computational cost of the IBPT is much closer to the ordinary PT than to the general BPT. Strictly speaking, one IBPT sample corresponds to two PT samples. For the test scene, in the region of caustics the IBPT allows to increase the convergence rate by more than an order of magnitude (see Figure 2).

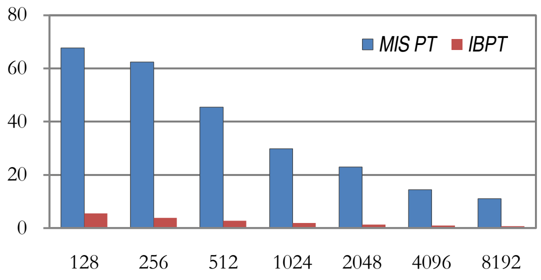

Figure 2. The absolute error depending on the number of samples per pixel (SPP)

Intermediate images generated with a small number of samples per pixel are shown in Figure 3. Bright pixels (fireflies) on the stems of wine glasses are the main factor that limits the convergence speed. These pixels are generated by complicated SDS paths (in Heckbert's notation) that can be processed only by implicit sampling strategies.

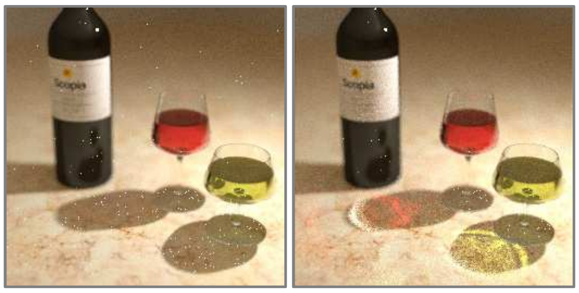

Figure 3. PT 128 SPP and IBPT 64 SPP

The general BPT does not solve the problem because the whole set of additional sampling strategies will have zero contribution. For efficient handling of such paths Metropolis Light Transport (MLT) can be used. An alternative approach is the Vertex Merging and Connection algorithm (VCM).

5. CONCLUSION

In this paper we presented light-weight modification of BPT, which is specially optimized for massively parallel architectures with limited memory resources like GPU. We showed that IBPT performs a lot better than simple MIS PT, especially in scenes containing strong indirect illumination and caustics.

6. REFERENCES

[Ant11] D. Antwerpen. *Improving SIMD Efficiency for Parallel Monte Carlo Light Transport on the GPU*. In Proceedings of High Performance Graphics 2011, pp. 41–50, 2011.

[Laf93] E. Lafortune, and Y. Willems. *Bi-Directional Path Tracing*. In Proceedings of Compugraphics '93, pp. 145–153, 1993.

[Vea94] E. Veach, and L. Guibas. *Bidirectional Estimators for Light Transport*. In Fifth Eurographics Workshop on Rendering, pp. 147–162, 1994.

[Url01] D. Antwerpen. *A Survey of Importance Sampling Applications in Unbiased Physically Based Rendering*: http://graphics.tudelft.nl/~dietger/survey.pdf