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 \( M \):

\[
M_j = \int_{x \in I} W_j(x_1 \rightarrow x_0) L(x_1 \rightarrow x_0) G(x_1 \leftrightarrow x_0) dA_{x_0} dA_{x_1}
\]

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

\[
G(x_1 \leftrightarrow x_0) = V(x_1 \leftrightarrow x_0) \frac{(x_{0x} \cdot x_{1x}) (x_{0x} \cdot x_{0x})}{||x_1 - x_0||^2}
\]

Here \( V \) is the visibility function (\( V = 1 \) if \( x_0 \) and \( x_1 \) are mutually visible, and \( V = 0 \) otherwise).

The total amount of outgoing radiance, \( L(x_1 \rightarrow x_0) \), can be computed as the sum of emitted radiance \( L_e \) plus reflected radiance \( L_r \):

\[
L(x_1 \rightarrow x_0) = L_e(x_1 \rightarrow x_0) + L_r(x_1 \rightarrow x_0) =
\]
\[ L_x + \sum_{\mathcal{A}} f_t(x_2 \rightarrow x_1 \rightarrow x_0) L(x_2 \rightarrow x_1) G(x_2 \leftrightarrow x_1) dA_{x_2} \]

Here \( f_t \) 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:

\[ L(x_1 \rightarrow x_0) = L_x(x_1 \rightarrow x_0) \]

Here \( l: \bigcup_{\mathcal{A}} M^{i+1} \rightarrow \mathbb{R} \) is the radiance flow function defined as

\[ l(x_0 \ldots x_k) = \sum_{i=1}^{k-1} f_t(x_{i+1} \rightarrow x_i \rightarrow x_{i-1}) G(x_i \leftrightarrow x_{i+1}) \]

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

A light transport path \( \mathbf{X}^k \) is any sequence of surface points \( x_0 \ldots x_k \in l \times M^k \) of which the first point \( x_0 \) lies on the film plane \( l \) 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 \( \Omega \).

The measurement equation can also be written over unified path space \( \Omega \). The generalized measurement function, \( f_j: \Omega \rightarrow \mathbb{R} \), is defined as

\[ f_j(x_0 \ldots x_k) = W_j(x_1 \rightarrow x_0) \cdot \prod_{i=0}^{k-1} G(x_i \leftrightarrow x_{i+1}) \]

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

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

This equation formulates the fundamental problem of which the first point \( x_0 \) equals \( dA_{x_0} \cdot dA_{x_1} \cdot \ldots \cdot dA_{x_k} \). This equation 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:

**3. GPU-OPTIMIZED BPT**

**Instant bidirectional path tracing (IBPT)** is an 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 \( y_i \) is connected to a random point \( z \) on a light source to form the explicit view path \( y_0 \ldots y_i \). If the path accidentally hits a light source at point \( y_i \), then the sequence \( y_0 \ldots y_i \) forms implicit view path.

- **Light tracing pass (LT).** Tracing a path starting at a selected light source. Each vertex \( z_i \) is directly connected to a random point \( y \) on the camera lens to form the explicit light path \( z_0 \ldots z_i \). If the path accidentally hits the camera lens at point \( z_i \), then the sequence \( z_0 \ldots z_i \) forms implicit light path.

Thus, in IBPT algorithm each path \( X^k = x_0 \ldots x_k \) 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 \( X^k = x_0 \ldots x_k \), 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 \( p_t(x_i) \) be a PDF of sampling \( x_i \), measured with respect to unit surface area. The \( p_{wt}(x_{i-1} \rightarrow x_i) \) and \( p_{wt}(x_i \leftarrow x_{i+1}) \) are the PDFs of sampling \( x_i \) from points \( x_{i-1} \) and \( x_{i+1} \), respectively, measured with respect to projected solid angle. These PDFs relate according to

\[ p_t(x_i) = G(x_i \leftrightarrow x_{i+1})p_{wt}(x_i \leftarrow x_{i+1}) = G(x_i \leftrightarrow x_{i-1})p_{wt}(x_{i-1} \rightarrow x_i) \]

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 \( X^k = x_0 \ldots x_k \) equals**

\[ p_t(X^k) = p_t(x_0) \cdot \prod_{i=1}^{k} p_t(x_i) = p_t(x_0)p_t(x_1) \cdot \prod_{i=1}^{k-1} G(x_i \leftrightarrow x_{i+1})p_{wt}(x_i \leftarrow x_{i+1}) \]

**PDF of explicit view path \( X^k = x_0 \ldots x_k \) equals**

\[ p_t(X^k) = p_t(x_0) \cdot \prod_{i=1}^{k} p_t(x_i) = p_t(x_0)p_t(x_1) \cdot \prod_{i=1}^{k-1} G(x_i \leftrightarrow x_{i+1})p_{wt}(x_i \leftarrow x_{i+1}) \cdot p_t(x_k) \]

Here \( p_t(x_0) \) and \( p_t(x_1) \) are the probability densities of sampling the vertices \( x_0 \) and \( x_1 \), 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 \( X^k = x_0 \ldots x_k \) equals**
We propose an efficient procedure for computing the power heuristic weights. Let us consider the inverse weight for explicit view path \( X^k = x_0 \ldots x_k \):

\[
\frac{1}{w_{\text{inv}}(X^k)} = 1 + \frac{p_{x_0}(X^k)^\beta}{p_{x_0}(X^k)^\beta} + \frac{p_{x_1}(X^k)^\beta}{p_{x_1}(X^k)^\beta} + \cdots + \frac{p_{x_k}(X^k)^\beta}{p_{x_k}(X^k)^\beta}
\]

Both explicit and implicit view paths are sampled in the same way except for the last vertex \( x_k \). Thus, the common factors cancel out:

\[
\frac{p_{x_k}(X^k)}{p_{x_k}(X^k)} = \frac{p_{x_0}(X_k-1 \rightarrow x_k)G(x_k-1 \leftrightarrow x_k)}{p_i(x_k)}
\]

Similarly, explicit and implicit light paths differ in how the first vertex \( x_0 \) is generated:

\[
\frac{p_{x_0}(X^k)}{p_{x_0}(X^k)} = \frac{1}{p_{x_0}(X^k)}
\]

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

\[
\frac{p_{x_k}(X^k)}{p_{x_0}(X^k)} = \frac{1}{p_{x_0}(X^k)}
\]

Obviously, this expression can be computed during the incremental path construction. At the \( i \)-th vertex (for \( i = 0 \ldots k \)) the following scalar variables need to be accumulated:

\[
s_0 = p_{x_0}^{-1}(x_1) \\
\vdots \\
s_k = p_{x_k}^{-1}(x_{k+1})
\]

\[
s_0 = p_{x_0}^{-1}(x_1) \\
\vdots \\
s_k = p_{x_k}^{-1}(x_{k+1})
\]

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](image)

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)](image)

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](image)

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


