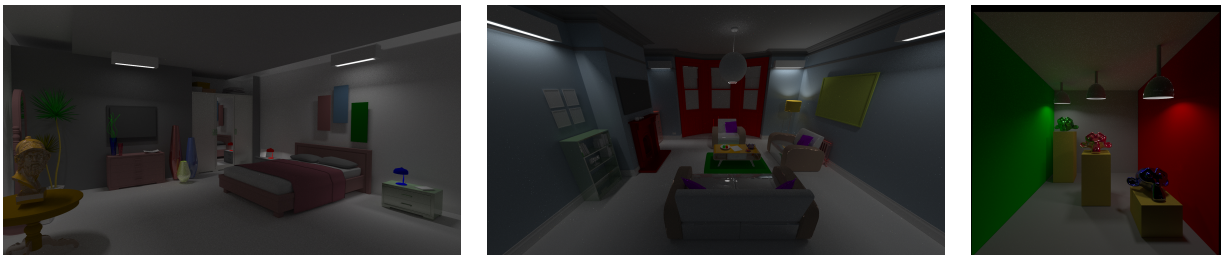
Coherent and Importance Sampled LVC BDPT on the GPU

Srinath Ravichandran*
IIIT - Hyderabad

P J Narayanan†
IIIT - Hyderabad



(a) *Bedroom (1.8M tris)* **(b)** *Whiteroom (574K tris)* **(c)** *Monkeybox (196K tris)*

Figure 1: *Scenes of varying geometric complexity with complex lighting and multi-layered materials. All the scenes were modelled with lights inside fixtures. All the scenes were rendered using CIS-LBDPT. Images 1a and 1b were rendered at 1980x1080 resolution for 256 iterations with 4 spp per iteration. Image 1c was rendered at 1024x1024 resolution for 128 iterations using 8spp per iteration.*

Abstract

Bidirectional path tracing (BDPT) can render highly realistic scenes with complicated lighting scenarios. The Light Vertex Cache (LVC) based BDPT method by Davidovic et al. [Davidovič et al. 2014] provided good performance on scenes with simple materials in a progressive rendering scenario. In this paper, we propose a new bidirectional path tracing formulation based on the LVC approach that handles scenes with complex, layered materials efficiently on the GPU. We achieve coherent material evaluation while conserving GPU memory requirements using sorting. We propose a modified method for selecting light vertices using the contribution importance which improves the image quality for a given amount of work. Progressive rendering can empower artists in the production pipeline to iterate and preview their work quickly. We hope the work presented here will enable the use of GPUs in the production pipeline with complex materials and complicated lighting scenarios.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing;

Keywords: Raytracing, Graphics Processors, Parallel Processing

1 Introduction

Light transport algorithms enable photorealistic rendering of scenes. Raytracing is the predominant method used in production rendering of movies and visual effects. Production rendering is characterized by scenes involving complex materials and heavy geometry. Bidirectional light transport algorithms are more efficient to render such scenes than path tracing. Commercial production renderers are primarily path tracers running on CPU render farms. GPUs and bidirectional path tracing on them have not made inroads there due to the higher memory requirments. Recent GPUs do provide sufficient promise for the use of BDPT for quick preview and render in production scenarios.

Bidirectional path tracing (BDPT) [Lafortune and Willems 1993] with the CPU generating the camera and light subpaths and the GPU computing combinatorial connections in parallel was developed by Pajot et al. [Pajot et al. 2011]. van Antwerpen presented a BDPT running on the GPU [van Antwerpen 2011]. These methods required large amounts of memory to store all vertices on the GPU. Davidovic et al. use a Light Vertex Cache (LVC) to store the light-path vertices, reducing memory requirements [Davidovič et al. 2014]. We modify this method to handle complex materials efficiently by employing material sorting to improve runtime performance. We also employ an importance based sampling of LVC vertices to get better quality images.

Laine et al. discussed efficiency issues associated with large monolithic kernels for path tracing on the GPU, particularly when the cost of shading was high [Laine et al. 2013]. They showed that smaller kernels utilized GPU resources better. They used queues to handle per material evaluations in separate kernels. We follow a similar approach in which separate kernels are employed for material evaluation and traversal. We use material sorting to bring more coherence to the simpler kernels as queues are inefficient on the GPUs. We present two ideas in this paper to exploit GPUs efficiently in such settings, over the Light Vertex Cache based BDPT (LVC-BDPT) [Davidovič et al. 2014] method.

1. We use a sorting-based parallel material evaluation method to bring more coherence to the evaluation of complex materials. This step is designed with GPU architecture in mind, where sorting is efficient and incoherence is costly. Complex layered materials increase execution incoherence and sorting brings greater coherence and helps reduce the material evaluation time greatly.

2. We present a new sampling method to connect a camera subpath vertex to promising light subpath vertices. Our method prefers connections that are likely to contribute more to the final image. This is achieved by sampling the light vertices

*e-mail:srinath.ravichandran@research.iiit.ac.in
†e-mail:pjn@iiit.ac.in

SA'15 Technical Briefs, November 02 – 06, 2015, Kobe, Japan.
 ISBN 978-1-4503-3930-8/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2820903.2820913

based on their contribution for the final image. This results in better image quality for a given number of visibility evaluations.

2 Bidirectional Path Tracing

The path integral formulation of light transport [Veach 1998] models how light moves within a scene and is recorded by a sensor within the scene. Each measurement recorded by the sensor can be written in the following form $I_j = \int_\Omega f_j(\overline{x})d\mu(\overline{x})$ where Ω is the set of all light transport paths of all possible lengths, μ is the measure on this space of paths, and f_j is called the measurement contribution function. Each path $\overline{x}$ of length k is composed of vertices $(x_0, x_1, ..., x_{k-1})$. Vertex x_0 sits on a light source, the vertex x_{k-1} on the camera sensor, and the intermediate vertices on other surfaces of the scene.

The measurement contribution function [Veach 1998] can be expressed as

$$f_j(\overline{x}) = L_e(x_0 \to x_1)G(x_0 \leftrightarrow x_1)W_e^j(x_{k-1} \to x_{k-2}) \prod_{i=1}^{k-2} f_s(x_{i-1} \to x_i \to x_{i+1})G(x_i \leftrightarrow x_{i+1})$$

where L_e is the radiance emitted from x_0 to x_1, G the geometry term between vertices, f the BRDF term and W the sensor importance function. In Monte Carlo light transport algorithms, the path $\overline{x}$ is created using local path sampling and can be sampled starting from a light source (light tracing), the camera (path tracing), or both (bidirectional path tracing). Unidirectional tracing methods have difficulty in finding paths that successfully transfer radiance from the light to the camera for certain lighting situations. Bidirectional path tracing methods can find more effective paths in such scenarios.

BDPT first creates samples from both the camera and the light and then generates the camera and light subpaths separately by tracing rays. When the paths terminate due to Russian roulette or fixed depth clamping, campera path vertices are are connected to light path vertices by shadow rays. Valid (i.e., visible) connections will contribute to the pixel colour by shading the materials. The first BDPT implementation on the GPU used fixed depth clamping [van Antwerpen 2011] because of the limited memory model of early generation GPUs. Davidovic et al. presented a Light Vertex Cache Bidirectional Path Tracing (LVC-BDPT) [Davidovič et al. 2014]. A coarse light prepass phase was used to determine the average path length of light subpaths. Based on this, memory was allocated to store the light subpath vertices for all the light samples. The intermediate subpath vertices were stored in memory as LVC during light traversal phase. All paths were terminated forcefully when memory was full. During the camera traversal, each camera subpath vertex sampled N vertices from the LVC randomly. Connections of the camera vertex with each light vertex were evaluated and the camera buffer was updated appropriately. They presented a single kernel variant (LVC-BDPTsk) and multikernel variant (LVC-BDPTmk) of their method.

3 Coherent and Importance Sampled LVC BDPT (CIS-LBDPT)

We propose two modifications to the LVC-BDPT scheme to improve speed and quality in the presence of complex materials. The first involves material sorting and the second involves importance sampling of LVC vertices.

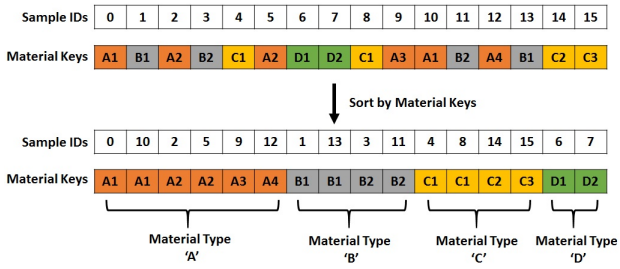


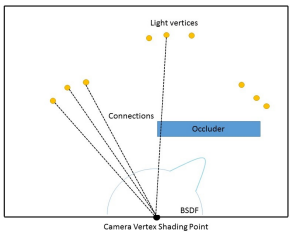
Figure 2: *Sorting of samples based on material keys. Each 32 bit key is composed of the first 8 higher order bits representing the material type(represented by alphabets A,B,C and D) and the remaining bits indicating the id of a particular material applied to a primitive(represented by the numeral).*

3.1 Sorting and Evaluation of Complex Materials

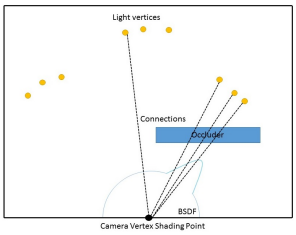
We employ a layered material system [Pharr and Humphreys 2010] to create materials of varied complexity. Materials in our system can have upto four independent layers. At each shading point, a material stack composed of the individual BSDFs is generated. We support a variety of BSDFs such as lambertian, specular reflection, specular transmission, Oren-Nayar, and microfacet models with a variety of distribution and geometric functions. Efficiency on the GPUs critically depends on coherent instruction execution and memory access. In a scene containing complex materials, rays in later bounces will hit different materials. This can result in thread divergence and degraded performance. This is more acute on layered materials with different types of BSDFs.

Laine et al. enhanced material coherence for path-tracing by placing each material-intersection request into a per-material queues [Laine et al. 2013]. We bring coherence to BDPT using material sorting in place of the use of queues. Queues have several disadvantages in a GPU setting. Queue management typically involves irregular computations that are not efficient on the GPUs. Queues also require atomic operations and wastes memory as memory needs to be preallocated to each based on the potential maximum length. Queues may additionally be overrun if their sizes are not estimated properly and are in general underutilized. Queue buffer overrun presents a problem in a bidirectional path tracing scenario where the number of rays are typically much larger than path tracing.

We use in-place sorting of the intersection points based on material ids during each step of the algorithm to enforce execution as well as data coherence. Every primitive in the scene is assigned a unique 32 bit composite key composed of the first 8 higher order bits representing the material type with the rest of bits employed for assigning a unique material id. During traversal phase each thread handling a sample creates a tuple of the composite key and sample id in a global list. This list is sorted on composite key using an efficient Thrust sort primitive, before material evaluation is performed. Only the sample id list is sorted and not the actual ray data. Sorting has several advantages. The GPUs have fast radix sorting primitives. Sorting uses memory efficiently; space for only the total number of samples is needed in any step with no wastage. Sorting brings all samples of same material together (Figure 2). We then evaluate materials using a single kernel which operates with more execution coherence since threads in a warp now handle sorted material types. Our material evaluation kernel is kept lean currently. However should the need arise to handle extremely complex materials, we can add a separate kernel to handle only those materials within the current framework. We demonstrate the computational advantage brought by sorting in the results section.



(a) *Scenario 1: Poor BSDF contribution at the camera vertex for the light vertices chosen.*



(b) *Scenario 2: Visibility test fails for the light vertices chosen thereby adding no contribution to the image*

Figure 3: *Two scenarios indicating how a particular choice of light vertices might result in poor contribution.*

3.2 Importance Sampling of LVCs

The connection term between a light path of length s and a camera path of length t is given by

$$C_{s,t} = f_s(y_{s-2} \rightarrow y_{s-1} \rightarrow y_{t-1}) f_s(z_{t-2} \rightarrow z_{t-1} \rightarrow y_{s-1}) \\ G(y_{s-1}, z_{t-1}) V(y_{s-1}, z_{t-1}),$$

where f_s terms are the BRDFs evaluated at the camera and light vertices for the connection direction, G the geometry term and V the visibility term. Depending upon the scene, estimation of these factors can be costly and bad a choice of LVC can result high variance. For example, a scene in which the light and camera subpath vertices both have high subpath contributions but have a failed visibility (Fig 3b) will result in the estimator being zero, while incurring an expensive shadow ray traversal. Even among visible LVC vertices, low product of BRDF values can also result in wasted computations (Fig 3a). The number N of samples will have to be increased to reduce the variance of the estimator in such scenarios.

We propose a different sampling scheme aimed at improving the quality of the rendered images. Our method is similar to the sampling-importance-resampling method used to reduce number of visibility tests based on the product of the BRDF and the incoming radiance terms [Burke et al. 2005]. Our scheme starts with sampling a larger number M of vertices from the LVC. We then compute a distribution over the potential contribution of the selected vertices by evaluating the product of the BSDF terms at the camera and light vertices. We then sample N light vertices using the distribution thereby choosing vertices that provide a higher contribution with a higher probability without introducing any bias into the computation. The N vertices are sent for visibility evaluation and image updation, if found visible. On complex scenes, visibility tests can dominate the render time. Preferring more promising vertices for visibility and discarding less promising ones will enhance the quality of the rendered image. We compare our new connection method against the original LVC-BDPT method by comparing the RMSE values obtained by iterative rendering of different scenes. We show that our algorithm is able to provide lower RMSE values for the same number of samples per pixel traced. Our method also provides lower RMSE values for a given total rendering time, as can be seen in the results section.

The pseudocode for the light tracing and camera tracing schemes of our Coherent and Importance-Sampled LVC BDPT (CIS LBDPT) method is given in Algorithms 1 and 2. We employ a multikernel approach for different tasks such as direct lighting computation, ray traversal, material evaluation and modified connection evaluations

Algorithm 1 Light Pass

```
1:  procedure LIGHT TRACE
2:      Create LightSamples in parallel
3:      while #LightSamples != 0 do
4:          Trace all rays in parallel
5:          Sort rays on intersected material id.
6:          Evaluate materials in parallel
7:          Store vertices in LVC in parallel
8:          Compute next bounce rays in parallel
9:          Compact rays to remove dead samples
```

Algorithm 2 Camera Pass

```
1:  procedure CAMERA TRACE
2:      Create CameraSamples in parallel
3:      while #CameraSamples != 0 do
4:          Trace all rays in parallel
5:          Sort rays on intersected material id.
6:          Stream0 - Evaluate direct lighting in parallel
7:          Stream1 - Evaluate modified connections in parallel
8:          Stream2 - Compute next bounce rays in parallel
9:          Trace connection shadow rays in parallel
10:         Compact rays to remove dead samples.
```

(Line 6, 7, 8 in Algorithm 2). We employ the multi stream launch capabilities of the latest GPU hardware to launch all these kernels in parallel and collect their results. In our experiments, we use $M = 10$ (#samples generated by our scheme) and $N = 5$ (#samples used to update the image by our and LVC-BDPT schemes) unless otherwise specified. Given the available memory on the GPU, our implementation is able to generate and trace 4 samples per pixel simultaneously on 1980×1080 images in parallel (That is, it processes 8.5 million camera rays at a time.). With increase in GPU memory we can accomodate even more sample resolutions. An iteration of our algorithm involves tracing these samples to completion. Subsequent iterations will increase the number of samples per pixel but tracing another 8.5 million rays. This can be thought of as progressively adding 4 samples for each pixel in each iteration. Every subsequent iteration performs both the light pass followed by the camera pass from the beginning thereby effectively flushing the LVC and utilizing a new set of light vertices for connection.

4 Implementation, Results and Analysis

We implemented our algorithms on a machine containing a corei7-4790K processor and a Nvidia GTX Titan GPU having 6GB of video memory. We used CUDA 7.0 and the Thrust libraries for sort and compaction. We created three different scenes with varying levels of geometric and material complexity to test the performance of our algorithm. All the scenes were designed with complicated lighting conditions by placing the light sources inside fixtures. The reference images used for computing RMSE values were progressively generated using a naive GPU-BDPT method till they had very little noise visible in them.

Sorting for Coherence: We compare the runtime performance of our algorithm against LVC BDPTmk algorithm that does not employ sorting to handle all the materials scenes having varying material and geometric complexity. Table 1 shows the running times for different scenes as the number of samples per pixel increases. These numbers are a good indicator of the performance of the CIS-LBDPT algorithm in the presence of complex materials. Good speed up is achieved across the board. The benefits obtained

by having coherent work chunks that can be handled by the GPU is very evident from the numbers themselves.

	Effective SPP	LVC-BDPTmk (ms)	CIS-LBDPT (ms)	Speedup
MonkeyBox	8	18237	14228	1.28x
	16	39534	31376	1.26x
	24	61867	46169	1.34x
	32	84471	64214	1.31x
	40	104630	78082	1.34x
Whiteroom	4	34890	24780	1.48x
	8	68011	48988	1.38x
	12	105419	76138	1.38x
	16	145022	106276	1.36x
	20	182498	126046	1.44x
Bedroom	4	31595	20374	1.48x
	8	67509	46053	1.46x
	12	103618	70717	1.46x
	16	141286	94269	1.49x
	20	179323	122053	1.46x

Table 1: *Comparison of execution times for LVC-BDPTmk and CIS-LBDPT methods. The Monkeybox scene was rendered at 1024x1024 resolution with 8spp per iteration. The Whiteroom and Bedroom scenes were rendered at 1980x1080 resolution with 4spp per iteration.*

Performance Results: Table 2 shows the quality comparison of rendered image as the rendering progresses. The plot of RMS error against iteration number shows for all the three scenes show the quality advantage of our improved sampling scheme. The error is lower for images rendered using our scheme than using LVC-BDPT which does only random sampling. The plot of RMS error against run time shows clearly that for a fixed time budget, the CIS-LBDPT achieves better rendering quality across the board. The quality gains are likely to be even more if larger values of M and N are used.

Limitations: Our method employs a new sampling scheme of choosing LVC vertices during the camera pass. It is performed with the goal of decreasing the variance by choosing good samples as well as reducing the number of shadow ray computations which would be required otherwise for sampling a larger pool of vertices for attaining the same quality. Even though method shown here handles fairly complex materials, the cost of evaluating extremely complex materials can become very high thereby increasing the computation time. This can hence reduce the performance gained in sorting the vertices for improved material coherence.

5 Conclusion

We presented a Coherent and Importance Sampled approach to perform LVC based BDPT efficiently on the GPU. Sorting provided coherence to material evaluation and an improved sampling method enhanced the rendering quality. Our work should inspire further improvements that will eventually place the GPU in the middle of a production rendering pipeline in the near future. For future work we would like to investigate out of core rendering of scenes with complex materials using our method.

Acknowledgements: We would like to thank Jay-Artist for the white room scene and other artists of the models used in the test scenes. All the models were downloaded from the blendswap website and are covered under the creative commons license.

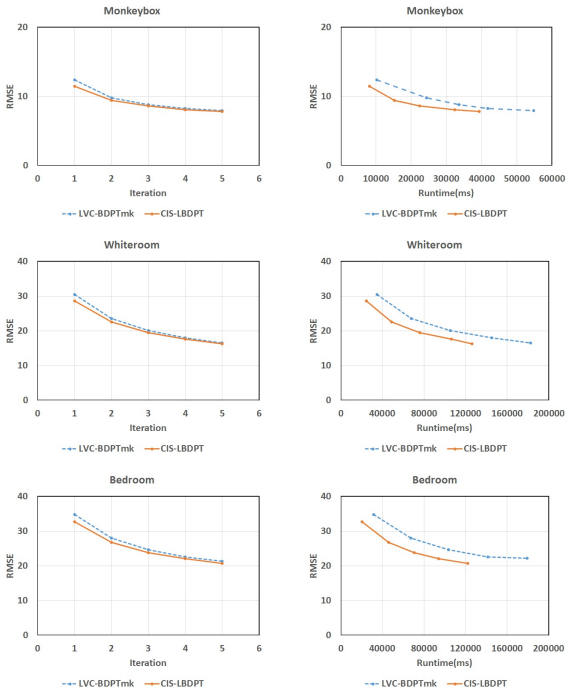


Table 2: *Comparison of performance of LVC-BDPTmk and CIS-LBDPT methods. The left and right side graphs indicate how the RMSE values for the both methods decrease with increasing iterations and time respectively.*

References

BURKE, D., GHOSH, A., AND HEIDRICH, W. 2005. Bidirectional importance sampling for direct illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, Eurographics Association, EGSR '05, 147–156.

DAVIDOVIČ, T., KŘIVÁNEK, J., HAŠAN, M., AND SLUSALLEK, P. 2014. Progressive light transport simulation on the gpu: Survey and improvements. *ACM Transactions on Graphics (TOG) 33*, 3, 29.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional path tracing. In *Proceedings of Third International Conference of Computational Graphics and Visualization Techniques (CompuGraphics' 93)*, 145–153.

LAINE, S., KARRAS, T., AND AILA, T. 2013. Megakernels considered harmful: Wavefront path tracing on gpus. In *Proceedings of the 5th High-Performance Graphics Conference*, 137–143.

PAJOT, A., BARTHE, L., PAULIN, M., AND POULIN, P. 2011. Combinatorial bidirectional path-tracing for efficient hybrid cpu/gpu rendering. *Computer Graphics Forum 30*, 2, 315–324.

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed.

VAN ANTWERPEN, D. 2011. Improving simd efficiency for parallel monte carlo light transport on the gpu. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, 41–50.

VEACH, E. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA. AAI9837162.