Efficient Point based Global Illumination on Intel MIC Architecture

Xiang Xu[1,2] Pei Wang[1,2] Beibei Wang[†][3] Lu Wang[‡][1,2] Changhe Tu[§][1,2] Xiangxu Meng[¶][1,2] Tamy Boubekeur[4]

[1]Shandong University, [2]Engineering Research Center of Digital Media Technology, Ministry of Education, China
[3]Inria Grenoble - Rhone-Alpes, LJK, CNRS, INPG, [4]LTCI, CNRS, Telecom ParisTech, Paris-Saclay University

Abstract

Point-Based Global Illumination (PBGI) is a popular rendering method in special effects and motion picture productions. The tree-cut computation is in general the most time consuming part of this algorithm, but it can be formulated for efficient parallel execution, in particular regarding wide-SIMD hardware. In this context, we propose several vectorization schemes, namely single, packet and hybrid, to maximize the utilization of modern CPU architectures. While for the single scheme, 16 nodes from the hierarchy are processed for a single receiver in parallel, the packet scheme handles one node for 16 receivers. These two schemes work well for scenes having smooth geometry and diffuse material. When the scene contains high frequency bumps maps and glossy reflections, we use a hybrid vectorization method. We conduct experiments on an Intel Many Integrated Core architecture and report preliminary results on several scenes, showing that up to a 3x speedup can be achieved when compared with non-vectorized execution.

1. Context

PBGI [Chr08] has been introduced to compute diffuse global illumination effects efficiently. At the heart of the algorithm, the scene is represented by a shaded-hierarchical point structure that acts as a multi-resolution radiance cache. A cut is gathered from this tree using a top-down traversal to shade every single receiver (e.g., pixel) using a splatting procedure which solves for visibility using a local variant of the Z-Buffer algorithm. In practice, the traversing process represents a large portion of the rendering time. In this paper we focus on the mapping of the PBGI tree traversal onto a wide-SIMD architecture, which has been previously used successfully to accelerate global illumination. In particular we address the case of the 16-wide SIMD Intel MIC architecture [Int10](MIC).

2. Related Work

PBGI. PBGI was first proposed by Christensen [Chr08] to evaluate diffuse light transport. Ritschel et al. [REG*09] and Holländer et al. [HREB11] both have provided efficient GPU implementations. A number of subsequent approaches have been proposed to improve PBGI, including tree-cut/microbuffers factorization based on spatial coherence [WHB*13], and a wavelet-based model to simulate non-diffuse light transport [WMB15]

CPU Parallelism. A number of approaches have been proposed to exploit CPU vectorization for ray tracing, including packet-based ray tracing [BEL*07], the combination of single and packet-ray [BWW*12] and a low level kernel to maximize CPU usage [WWB*14].

3. Method

We represent our spatial hierarchy as a BSH with a branching factor of two. We propose three schemes to organize the tree traversal so that it fits MIC: *single*, *packet* and *hybrid*.

Single Vectorization. For the *single* scheme, we fill 16 SIMD lanes with different tree nodes and then one receiver processes with these tree nodes in parallel. All the *available nodes* (the nodes which wait for traversal) are kept in a traversal queue, and we put 16 nodes from this queue to the SIMD lanes. During the traversal, three different masks are used for the node: a *traversing* mask, a *splatting* mask and a *discarding* mask. After this traversal operation, the nodes masked for traversing push their children node to the traversal queue and the nodes masked for *splatting* are splatted to the receiver microbuffer, while the nodes masked as *discarding* are culled. In this scheme, only one traversal queue is required to keep the available nodes. In practice, we observed that this approach can lead to a $2\times$ to $3\times$ speed up compared to the scalar one.

Packet Vectorization. For the *packet* scheme, each lane has its own receiver and microbuffer, and all the lanes share a traversal queue and a mask queue. The mask queue stores all the masks that determine whether a node will be traversed for a receiver or not. Each time, a node is picked from the traversal queue to fill the 16 lanes, while 16 masks are picked from the mask queue to fill

[†] Email:beibei.wang@inria.fr
[‡] Email:luwang_hcivr@sdu.edu.cn
[§] Email:chtu@sdu.edu.cn
[¶] Email:mxx@sdu.edu.cn

© 2016 The Author(s)
Eurographics Proceedings © 2016 The Eurographics Association.

DOI: 10.2312/egp.20161040

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

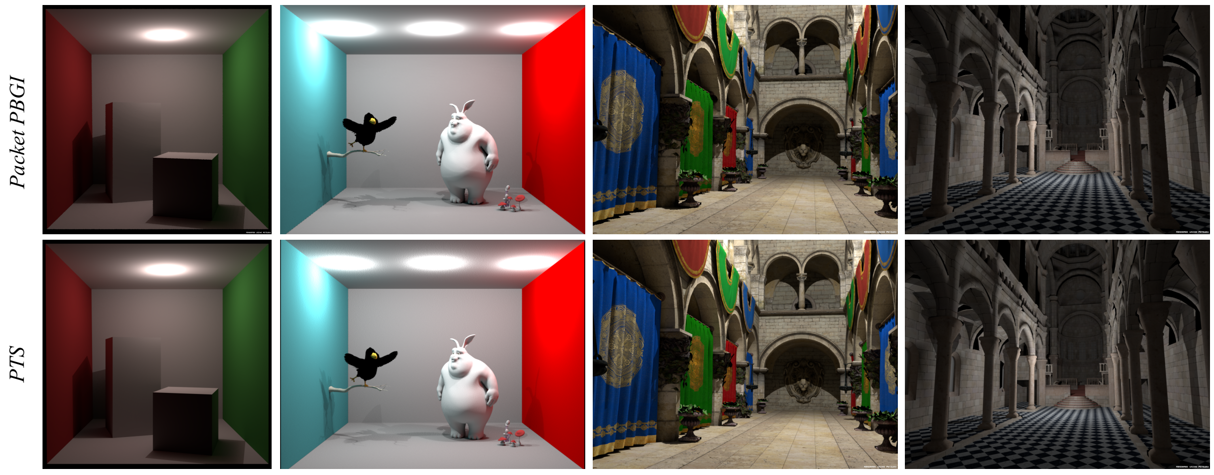

Figure 1: *Visual comparison with path tracing.*

scene	pre.	trav. time				others.
		Thread	Single	Packet	Hybrid	
CBox	2.95	50.90	24.19	17.61	18.19	3.74
Bunny	3.25	212.49	100.51	70.34	84.20	34.50
Sponza	4.28	266.60	122.28	90.22	86.73	7.53
Sib.	4.11	106.51	51.02	37.48	37.07	4.56

Table 1: *Performance measures (in seconds).*

16 lanes. After each traversal operation, both queues are updated and the nodes which satisfy the splatting threshold are projected onto the receiver's microbuffer. In the case of smooth geometry and quite diffuse reflectance, close receivers have a high probability to get similar tree-cuts and all SIMD lanes usually perform the same arithmetic operations together, achieving high SIMD utilization, with a measured $3\times$ speedup compared to the non-vectorized approach.

Hybrid Vectorization The high SIMD utilization of the *packet* scheme relies on the coherency of the tree-cuts for nearby receivers. When it comes to high frequency variations in the cut, such as with bump maps or glossy reflections, this SIMD performance diminishes drastically. To handle this problem, we propose a hybrid vectorization scheme, combining the *single* scheme and the *packet* scheme. Starting from the packet scheme, we trace the count of the *active receivers*, and when this number is less than a given threshold, we save the traversal state and switch to the *single* scheme for the last stages of the traversal. The initial traversal queue for a receiver is obtained from the traversal queue and the mask queue of the *packet* stage. Each receiver traverses one by one independently. While still at its earliest development stage, this technique is expected to bring more speedup than the packet one for such complex scenes.

4. Results

We implemented our algorithm in the Mitsuba Renderer [Jak10]. We compare (i) thread parallel execution (Thread), (ii) single vectorization (Single), (iii) packet vectorization (Packet) and (iv) path tracing solution (PTS), which we consider as ground truth for quality comparison, (shown in Fig.1). Performances (see Tab. 1) are measured on 1.1 GHz Intel Xeon Phi Coprocessor 7110P(61 core) with 8GB memory. The *others* in Tab. 1 includes the time for splatting and convolving.

Acknowledgements This work was supported by the CNNSF under Grant No.61472224, Shandong Key Research and Development Program under Grant No.2015GGX106006, the special funding of independent innovation and transformation of achievements in Shandong Province under Grant No. 2014ZZCX08201, the special funds of Taishan scholar construction project, Young Scholars Program of Shandong University under Grant No.2015WLJH41.

References

[BEL*07] Boulos S., Edwards D., Lacewell J. D., Kniss J., Kautz J., Shirley P., Wald I.: Packet-based whitted and distribution ray tracing. In *Graphics Interface 2007* (2007), pp. 177–184. 1

[BWW*12] Benthin C., Wald I., Woop S., Ernst M., Mark W. R.: Combining single and packet-ray tracing for arbitrary ray distributions on the intel mic architecture. *IEEE Transactions on Visualization and Computer Graphics 18*, 9 (Sept. 2012), 1438–1448. 1

[Chr08] Christensen P.: *Point-based approximate color bleeding*. Tech. Rep. 08-01, Pixar Technical Notes, 2008. 1

[HREB11] Holländer M., Ritschel T., Eisemann E., Boubekeur T.: Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Comp. Graph. Forum (Proc. EGSR 2011) 30*, 4 (2011), 1233–1240. 1

[Int10] Intel: Intel many integrated core architecture. http://download.-intel.com/pressroom/archive/reference/ISC 2010 Skaugen keynote.pdf, 2010. 1

[Jak10] Jakob W.: Mitsuba renderer. www.mitsuba-renderer.org, 2010. 2

[REG*09] Ritschel T., Engelhardt T., Grosch T., Seidel H.-P., Kautz J., Dachsbacher C.: Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2009) 28*, 5 (2009). 1

[WHB*13] Wang B., Huang J., Buchholz B., Meng X., Boubekeur T.: Factorized point-based global illumination. *Computer Graphics Forum 32*, 4 (2013), 117–123. 1

[WMB15] Wang B., Meng X., Boubekeur T.: Wavelet point-based global illumination. *Computer Graphics Forum (Proc. EGSR 2015), to appear* (2015). 1

[WWB*14] Wald I., Woop S., Benthin C., Johnson G. S., Ernst M.: Embree: A kernel framework for efficient cpu ray tracing. *ACM Trans. Graph. 33*, 4 (July 2014), 143:1–143:8. 1