Kang CM, Wang L, Wang P *et al.* Coherent photon mapping on the Intel MIC architecture. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(3): 519–527 May 2015. DOI 10.1007/s11390-015-1542-1

Coherent Photon Mapping on the Intel MIC Architecture

Chun-Meng Kang [1,2] (康春萌), Lu Wang [1,2,*] (王 璐), Pei Wang [2,3] (王 佩), Yan-Ning Xu [1,2] (徐延宁), and Xiang-Xu Meng [1,2] (孟祥旭), *Member, CCF*

[1] *School of Computer Science and Technology, Shandong University, Jinan 250101, China*

[2] *Engineering Research Center of Digital Media Technology, Ministry of Education, Jinan 250101, China*

[3] *Software College, Shandong University, Jinan 250101, China*

E-mail: kcm89kimi@163.com; luwang_hcivr@sdu.edu.cn; jobwangpei@gmail.com; {xyn, mxx}@sdu.edu.cn

Received November 30, 2014; revised March 9, 2015.

Abstract Photon mapping is a global illumination algorithm which is composed of two steps: photon tracing and photon searching. During photon searching step, each shading point needs to search the photon-tree to find k-neighbouring photons for reflected radiance estimation. As the number of shading points and the size of photon-tree are dramatically large, the photon searching step is time consuming. We propose a parallel photon searching algorithm by using radiance estimation approach for coherent shading points on the Intel® Many Integrated Core (MIC) Architecture. In order to efficiently use single instruction multiple data (SIMD) units, shading points are clustered by similarity first (every cluster contains 16 shading-points), and an initial neighbouring scope is searched from the photon-tree for each cluster. Then we use 16-wide SIMD units by performing k-NN searching from the initial neighbouring scope for those 16 shading-points in a cluster in parallel. We use the method to simulate some global illumination scenes on Intel® Xeon® processors and Intel® Xeon® Phi™ coprocessors. The comparison results with existing photon mapping techniques indicate that our method gives significant improvement in speed with the same accuracy.

Keywords photon mapping, parallel processing, SIMD

1 Introduction

Photon mapping is an extension of ray tracing method that makes it able to efficiently compute global illumination effects, such as caustics, ambient occlusion, color bleeding, soft shadows and soft indirect illumination in participating media. The visual impact of global illumination is essential for photo-realistic rendering. Fast and high quality global illumination has been the central goal of photo-realistic image synthesis for a long time.

On modern programmable architectures such as CPUs, GPUs and MICs, the key to reaching the goal of speeding up is to efficiently use those architectures' SIMD units. Intel® Many Integrated Core (MIC) Architecture Xeon® Phi™ coprocessors have the same fundamentals of vectorization or bandwidth with main processors. Therefore, a system using Intel® Xeon® Phi™ coprocessors will have broader applicability than a system using GPUs. In this paper, we propose a parallel photon mapping algorithm using Intel® Xeon® processors and Intel® Xeon® Phi™ coprocessors to explore the acceleration of global illumination through SIMD execution. In our parallel algorithm, we achieve photon tracing and photon searching on Intel® Xeon® Phi™ coprocessor, and SIMD instructions are both used in the photon tracing and the photon searching steps.

Regular Paper

Special Section on Computational Visual Media

This work was partly supported by the National Natural Science Foundation of China under Grant Nos. 61472224, 61472225, the National High Technology Research and Development 863 Program of China under Grant No. 2012AA01A306, the National Key Technology Research and Development Program of China under Grant No. 2013BAH39F02, the Special Funding of Independent Innovation and Transformation of Achievements in Shandong Province of China under Grant No. 2014ZZCX08201, and the Special Funds of Taishan Scholar Construction Project of China.

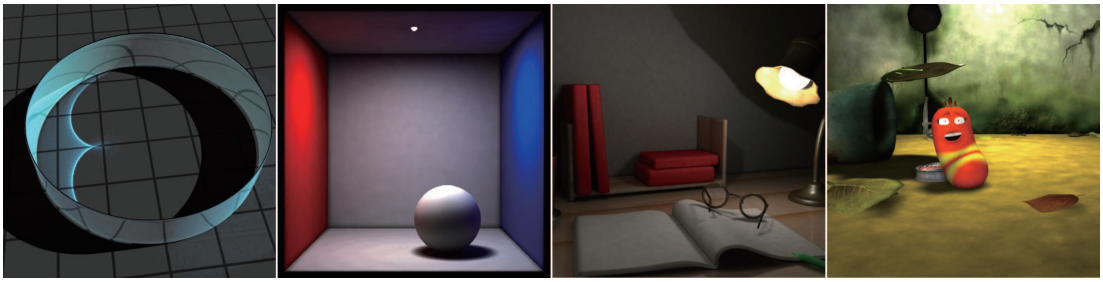
*Corresponding Author

The standard photon mapping method[1] in the simplest way can be summarized in two main steps: photon tracing and photon searching.

In the photon tracing step, photons are emitted from the light sources, traced through the scene, and stored in the photon maps. During this step, we follow the ray tracing method proposed by Benthin *et al.*[2] to trace photons. By using a bounding volume hierarchy (BVH) tree with a branching factor to store objects in the scene, each photon can do intersection test with four nodes of the BVH tree in parallel by using 16-wide SIMD units of Intel® Xeon® Phi™ coprocessors. After the photon-tracing step, a photon-tree (organized as a K-D tree) can be constructed.

During the photon searching step, the radiance information needs to be estimated for each shading point by searching the k-neighbouring photons in the photon-tree. Considering the spatial coherence property of shading points, a proper neighbouring scope of photons can be shared by nearby shading points, and each shading point can find its explicit k-neighbouring photons from this public photon scope. As the nearby shading points and the photons in the public scope are both with spatial coherence, the SIMD units of MIC can be efficiently used when we search k-neighbouring photons.

The main contribution of this paper is a parallel photon searching algorithm by a novel factorized radiance estimation method to improve SIMD utilization. By using a similarity-based merging clustering step for shading points, 16 shading points can be organized in a cluster. Then a central point can be selected for each cluster to find an initial neighbouring scope of the photon-tree for each cluster. Finally, a 16-wide k-NN searching for shading points in every cluster can be easily implemented by SIMD. Thus the radiance information of those 16 shading points in a cluster can be calculated in parallel. In Fig.1, we show the rendering results for different scenes.



(a) (b) (c) (d)

Fig.1. Results rendered using our algorithm. (a) Metal ring scene is an experiment for caustic effect rendering, and (b) cornell box scene, (c) desk model, and (d) small insect scene are examples for global photon mapping test.

2 Background and Previous work

The idea of speeding up global illumination has been explored by several researchers in the last few years. In order to make better use of graphics hardware computing power, different calculation methods and data structures are applied to the global illumination algorithms, such as ray tracing, point-based global illumination (PBGI), and photon mapping (PM).

Ma and McCool[3] presented a neighbourhood-preserving hashing algorithm that is low-latency and has sub-linear access time on GPU in 2002. In 2003, Purcell *et al.*[4] also presented a modified photon mapping algorithm in which the photons are stored in a grid-based photon map. In 2009, Fabianowski and Dingliana[5] proposed a highly parallel photon mapping algorithm that utilizes CUDA architecture, by handling diffuse reflections using photon differentials. Gupte[6] presented a hybrid photon-mapping approach for global illumination using the spatial hashing method to store and retrieve a photon map.

Wang *et al.*[7] used k-means to sample receiving points and interpolate irradiance. Their approach improves the final gather and photon mapping method, by selecting a representative point to perform the gather process and the others to perform the interpolation process. In 2013, Wang *et al.*[8] similarly applied the idea of receiving points coherence clustering to eliminate redundant computations in PBGI.

Using efficient GPU ray shooting and K-D tree building, Zhou *et al.*[9] implemented efficient photon mapping based on K-D tree querying. Their approach is similar to the original photon mapping idea, but using modern graphics hardware.

Recently, the first GPU method of progressive photon mapping[10] has prevented constructing K-D tree and standard spatial hashing data structures. Photon mapping method can be used in volume rendering[11], and Zhang *et al.*[12] proposed a real-time volume rendering by using precomputed photon mapping.

After years of development in parallel rendering technology, many researchers increasingly focus on SIMD units with both CPUs and GPUs using ever wider SIMD units: 8-wide AVX on Intel CPUs①, and 16-wide (or greater) SIMD units on GPUs[13]. Wald *et al.*[14] proposed the first SIMD ray tracing that defines the concept of packet tracing and works by traversing N different rays of a packet in parallel through sharing one traversal stack. Singh and Faloutsos[15] presented a novel photon mapping framework that uses SIMD parallelism to accelerate the final gathering phase of photo mapping. To implement SIMD photon gathering, they used Intel SSE[16] instructions. Intel issued the Xeon® Phi™ coprocessor which is based on MIC Architecture② with the KNC instructions. The flexibility of an Intel® Xeon® Phi™ coprocessor provides its suitability for complex parallel structure.

To improve SIMD utilization, Benthin *et al.*[2] used a bounding-volume hierarchy with four branches as the acceleration structure to efficiently perform intersection tests in parallel when the rays are incoherent. They implemented their method on the Intel® MIC Architecture which is designed for highly parallel applications with the highest demands for compute power and memory bandwidth.

3 Algorithm Overview

By analyzing the performance of photon mapping algorithm, we find that the photon tracing and the photon searching steps are most time consuming. Thus we apply the idea of SIMD to accelerate these two steps respectively. Fig.2 shows the working flow of our rendering system. The photon tracing and radiance estimation modules are implemented on the coprocessor. The other modules are implemented on the host.

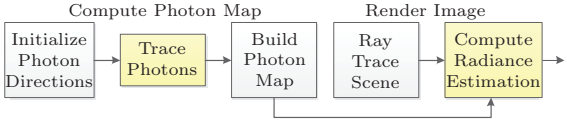


Fig.2. Working flow of our rendering system.

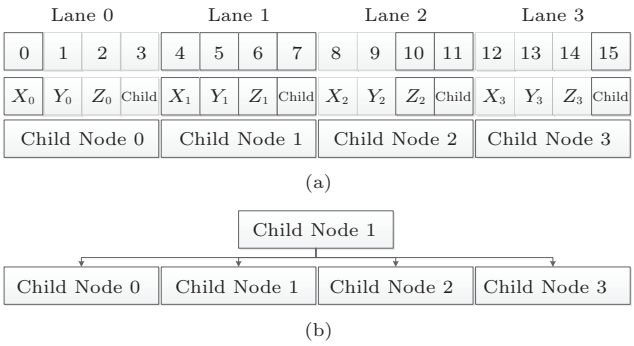


Fig.3. (a) Take a 16-wide SIMD as four lanes and each lane is composed of four elements. (b) Organize a QBVH tree.

Firstly, in the photon-tracing step, photons are emitted from the light sources, and directions are randomly assigned according to properties of the light sources. Then, these photons are traced through the scene, which means tracing N independent rays. There are many performance challenges associated with the parallel N independent rays algorithm. The main one is that it needs local storage for N independent rays and temporary variables. Especially when rays are totally incoherent, the performance of the parallel N independent rays algorithm degrades significantly.

We implement the parallel N independent photons tracing algorithm on MIC based on the work of Benthin *et al.*[2] which takes the 16-wide SIMD hardware of MIC as four lanes of four elements, and uses this to process four nodes intersection test in parallel. To realize this efficiently, a four-wide BVH (also called Quad-BVH or QBVH) should be organized for single ray traversal (see Fig.3). In this step, we launch a beam of photons each time, and load these photon rays onto MIC for the intersection in parallel. As different kinds of behavior may happen on the surfaces after the first bounce, the intersection results should be returned and we reorganize these bundles of photon rays after the first bounce behavior. Then, data will be loaded onto MIC to intersect until reaching the maximum depth.

Meanwhile, in the photon searching step, each shading point needs to search the photon-tree to find k-neighbouring photons for reflected radiance estimation. It is true that k-NN searching in a K-D tree can be simply implemented in parallel. However, K-D tree searching requires random read and write frequently. Therefore, parallel K-D tree algorithms do not have good performance on graphics hardware generally. In

①http://software.intel.com/en-us/avx, Mar. 2015.

②http://download.intel.com/pressroom/archive/reference/ISC_2010_Skaugen_keynote.pdf, Mar. 2015.

this case, we propose a two-step K-D tree searching approach, and make it run in parallel on the Intel MIC architecture efficiently by organizing shading points into clusters. In order to better use wide-SIMD capability, shading points are clustered by similarity. We should make sure that each cluster contains 16 shading-points, and we search an initial neighbouring scope from the photon-tree for each cluster. Then we use 16-wide SIMD by performing k-NN searching from the initial neighbouring scope for those 16 shading-points in a cluster in parallel. A flow of this step is illustrated in Fig.4.

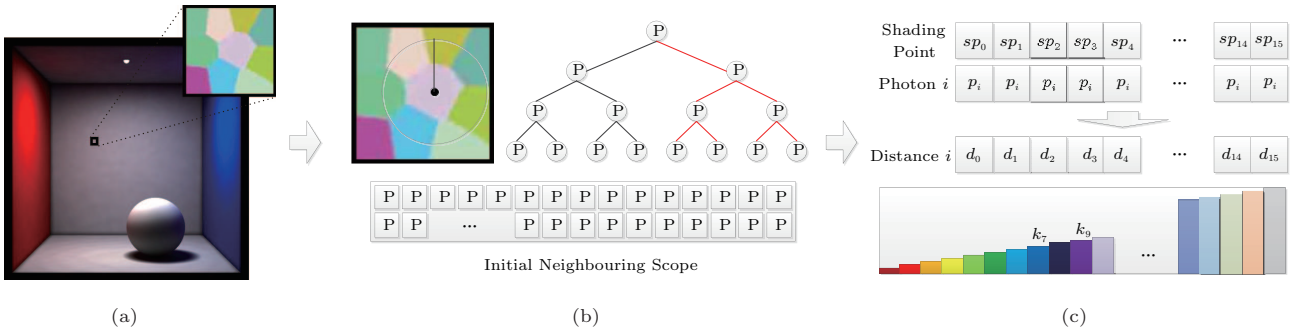


Fig. 4. Flow of photon searching step. (a) Cluster shading points. (b) Find an initial neighbouring scope from the photon-tree for each cluster. (c) Find k-NN for 16 shading-points in a cluster in parallel. P: photon.

4 Parallel Radiance Estimation

Our basic assumption is that shading points with spatial coherence have similar k-neighbouring photons. We propose a clustering strategy to collect coherent shading points into a cluster, and points in a cluster can use their spatial coherence to efficiently use SIMD units when doing radiance estimation. Each cluster can work independently and run in parallel.

4.1 Shading-Points Clustering

Shading points are those intersection points of rays and the scene in ray tracing pass. As rays are traced by bundles with similar directions, shading points located on the same object are with spatial coherence. Thus we divide the shading points into different groups according to the objects they located on. Then, we divide each group to clusters according to the similarity of shading points. We define the similarity of shading points with their positions and normals.

$$D(x_1, x_2) = \|p_{x_1} - p_{x_2}\|^2 + \alpha \times \|n_{x_1} - n_{x_2}\|^2,$$

with x_1 and x_2 being two points, p being their position, and n being their normal in the 3D geometric space. The weight α trades cluster flatness for spatial extent. We typically set it to 1.

Then, we cluster shading points based on this similarity by a hierarchical aggregation method. And each cluster contains 16 shading-points.

1) We initialize a random cluster list $\{C_1, C_2, \ldots, C_m\}$ with m shading points and assign one point for each cluster.

2) We merge the first unclassified cluster with its most similar cluster and calculate their center position and normal for the next level clustering.

3) We repeat 2) until we get 16 shading-points in each cluster.

In step 2, the center position p_c and the normal n_c of a cluster C with N_c points are updated as follows respectively:

$$p_c = \frac{1}{N_c} \sum_{i=1}^{N_c} p_i,$$

$$n_c = \frac{\sum_{i=1}^{N_c} n_i}{\| \sum_{i=1}^{N_c} n_i \|},$$

where N_c is the number of shading points contained in a cluster. When a cluster contains more than one point, we use its center position and normal to measure the cluster's similarity with others.

We perform this classification algorithm before K-D tree searching when we receive a set of shading points. Then we search k-NN photons for every point in a cluster using SIMD.

4.2 Factorized k-Neighbor Photons Searching

To achieve high SIMD utilization, we propose a factorized (two-step) K-D tree searching method. Firstly,

we search an initial neighbouring scope from the photon-tree for each cluster. We call this step range searching by using an initial search radius R_0. The radius R_0 should be conservative such that k nearest photons for every point in the cluster are within this radius and the radius remains as small as possible. Fig.5(a) shows the R_0 we use, and the formula is:

$$R_0 = r_0 + distance, \tag{1}$$

with r_0 being the initial search range for one shading point, and *distance* being the farthest distance from the center point to shading points in the cluster. Different clusters are with different search radius values R_0. The initial neighbouring scope of photons can be searched for each center point of a cluster with searching radius R_0.

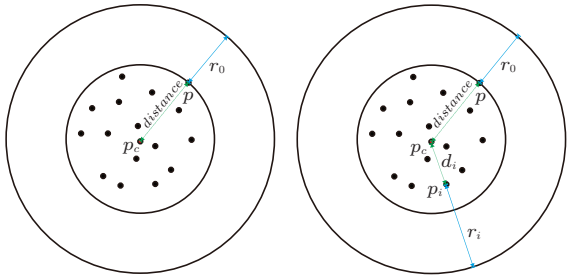


Fig.5. (a) Components of the initial searching radius R_0 for a cluster in (1). (b) Searching radius r_i for one point in a cluster in (2).

Secondly, we implement our parallel k-NN searching algorithm from the initial neighbouring scope for those 16 shading-points in a cluster following Zhou *et al.*[9] The key idea is that instead of using a priority queue, we try to find every shading point's k-NN photons within radius r_i through the range histogram. r_i is set as R_0 in areas with dense photons. During the iteration, we construct a two-dimensional (2D) histogram for 16 different shading points synchronously. By splitting the searching radius r_i for each shading point to different radius ranges, the number of photons is counted in different radius ranges. The horizontal axis of the histogram shows the radius ranges and the longitudinal axis indicates the number of photons located in each range. The k-NN photons can be searched iteratively according to the histogram, and the 16 shading-points can be processed in parallel. The following is the searching process which starts from the initial neighboring scope.

1) Calculate every shading point's searching radius r_i. Divide r_i into multi-ranges to construct the shading point's histogram, and initialize the number of photons in each radius range as 0.

2) For every photon in the initial neighbouring scope, calculate its distance to the 16 shading points in a cluster in parallel, and this photon can be classified into proper range in each shading point's histogram.

3) For each shading point, find the first range which contains the k-th photon based on its histogram, and then divide this range into multi-ranges again. Then 2) is repeated until the specified maximum number of iterations are reached.

4) Search k-neighbor photons in the range containing the k-th photon with a priority queue.

We achieve SIMD through organizing the data of 16 points in a cluster into several vectors to use MIC's vector calculation unit. Fig.6 illustrates the organization of position vectors, with $\boldsymbol{sp}_x$, $\boldsymbol{sp}_y$, $\boldsymbol{sp}_z$ being the position vectors of a cluster and $\boldsymbol{px}_i$, $\boldsymbol{py}_i$, $\boldsymbol{pz}_i$ being the position vectors of the i-th photon. In $\boldsymbol{sp}_x$, the respective components from 0 to 15 equal the value of the x-coordinate from the 0th to the 15th points in a cluster. However, in $\boldsymbol{px}_i$, the values of the respective components from 0 to 15 are the same, and equal the i-th photon's x-coordinate value. The organization is similar for another two position vectors of y-coordinate and z-coordinate and the normal vectors. Then, we can compute the distance of the 16 shading points with a photon through one distance calculation. A 2D histogram also contains several 16-wide vectors for counting and saving range radius of different points. Thus, we achieve computing 16 shading points by using the 16-wide SIMD units.

4.3 Hybrid Scheme in Sparse Area

The factorized radiance estimation scheme is more suitable for SMID than k-NN searching with a priority queue, but excess calculations make the results lack of accuracy in the place with sparse photon distribution. In these areas, we determine the query radius r_i directly for every shading point in the cluster by (2). With the same formula, we compute k-NN photons from the initial neighbouring scope using the 16-wide SIMD units to compute distance in parallel.

$$r_i = r_0 + distance - d_i, \tag{2}$$

where r_0 is the initial search range for the center point, *distance* is the farthest distance from the center point to shading points in the cluster, and d_i is the distance from the center point to the i-th shading point. Fig.5(b) illustrates the value of r_i.

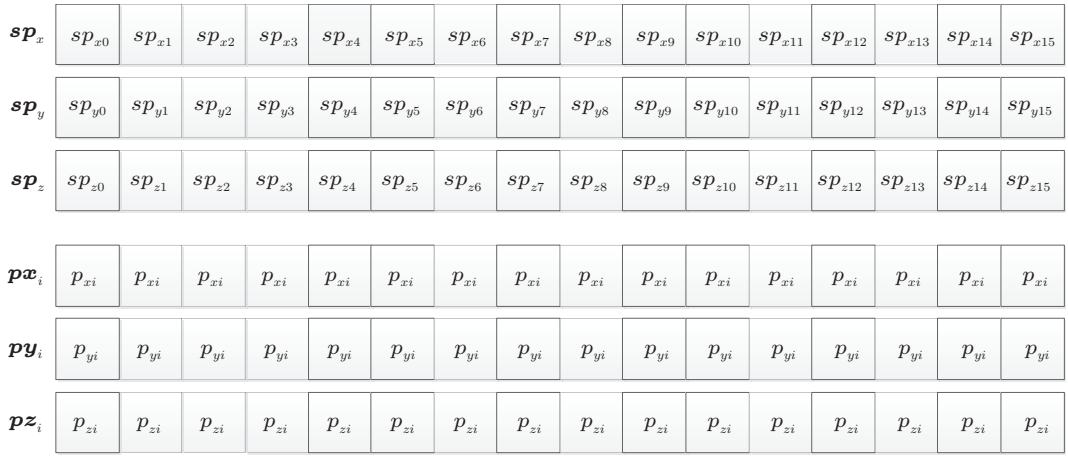


Fig.6. We use a 16-wide SIMD unit in our method. It shows that the organization of shading-point's position vector and photon's position vector.

When the number of photons in the initial neighbouring scope is less than the searching number k, we start out with the simple calculation process for query radius r_i by (2). Fortunately, we can easily derive a threshold through the distance and the number. The basic assumption is that photon density is a constant within a small area, and it is inversely proportional to the query radius r_i.

$$K_c = b \times k,$$
$$b = (R_0/r_0)^2,$$

where K_c is the threshold and k the number for k-NN searching.

5 Results

We implement our algorithm in the Pixie Renderer. Performances are measured on a server equipped with an Intel® Xeon® CPU E5-2609 at 2.50 GHz with 16 GB of main memory and an Intel® MIC Architecture Xeon® Phi™ coprocessor SC7110P at 1.10 GHz with 8 GB of shared memory. Images are rendered with our method at a 960×720 pixels resolution (except for the cornell box, at 720×720).

In all experiments, we test our method's computational efficiency using 120 threads in a process of calculation, which contains a set of shading points and has the maximum capacity of 2 000. Then, we classify the shading points, and each thread is allocated to process a cluster with SIMD.

In Fig.1, we show the rendering results for different scenes, both caustic photon map and global photon map have been tested. In Fig.1(a), there is a metal ring, and the scene generates 8 MB caustic photon map which takes about 150 thousand photons in the map. In addition, we measure radiance estimation with the photon's searching number being 50. Fig.1(b), Fig.1(c) and Fig.1(d) show global illumination in a cornell box scene, a desk model, and an insect scene respectively. Global photon map contains the number of enormous photons with respect to the caustic photon map. It takes more than 10 million photons in the map, and 2 000 is used as the search value, K.

In Table 1, we report the total time (T time) and the searching time (S time) for the four different examples shown in Fig.1, where MES is the mean squared error. Here, we can assess the benefit of our approach for caustic effect rendering, with a speed-up ratio for the total rendering time ranging from 1.2 to 1.3 compared with the original photon mapping algorithm. Meanwhile, in the specific portion of the algorithm that we target (radiance-estimation), the speed-up ratio ranges from 8.5 to 8.8. In the experiments using global photon map, accelerating effect is more prominent. The speed-up ratios range from 3.7 to 10.1 and from 8.4 to 25.6, corresponding to total rendering time and searching time respectively. Additionally, in the photon tracing step, the acceleration rate is about four times.

Table 1. Time Comparison and Error Analysis for the Four Different Experiments

Scene	CPU T Time (s)	CPU S Time (s)	MIC T Time (s)	MIC S Time (s)	MES
Metal ring	121	62	103	7	0.584
Cornell box	448	352	117	32	0.268
Desk	574	536	153	64	0.669
Insect	3 930	3 767	389	147	0.078

We compare the difference between our coherent radiance estimation and the original photon mapping algorithm. Overall, we observe a negligible error, both from the visual detail comparison of Fig.7 and the numerical error with MES in Table 1. The values of MES prove that our classification and factorized search are effective. We also check the coherence for shading points in a cluster. For each shading point p_i, the minimum distance between p_i and other shading points in a cluster is signed as d_{p_i}. If $d_{p_i} < \epsilon$, where ϵ is the mean *distance* of all clusters, we consider the points in a cluster are with spatial coherence. If all points in a cluster are coherent, we consider the cluster with spatial coherence. By using the checking standard, the percentage of coherence clusters is about 99% in our experiments.

We also analyze the influence of the number of photons for our method, both the number of photons collected and the number of photons emitted. By comparing the experimental results of the global photon map and the caustic photon map, it can be clearly found that acceleration is more efficient when global photon map is used. The reason is that in global photon map, there is a great-high tree with a large number of photons and almost all of the shading points need to search for enough photons for radiance estimation, which takes a long time to traverse. In our method, a cluster only needs to traverse once, thus the method harvests a high acceleration efficiency. The search number k is also a very important parameter, which impacts the accuracy of the rendering results. In Fig.8, we plot the speed-up evolution under the variation of the number k, both the total time and the searching time. We can see from the figure that the larger the k, the better the acceleration. Choosing a larger k value will improve acceleration effect, but on the other hand, the larger k is, the more bias appears.

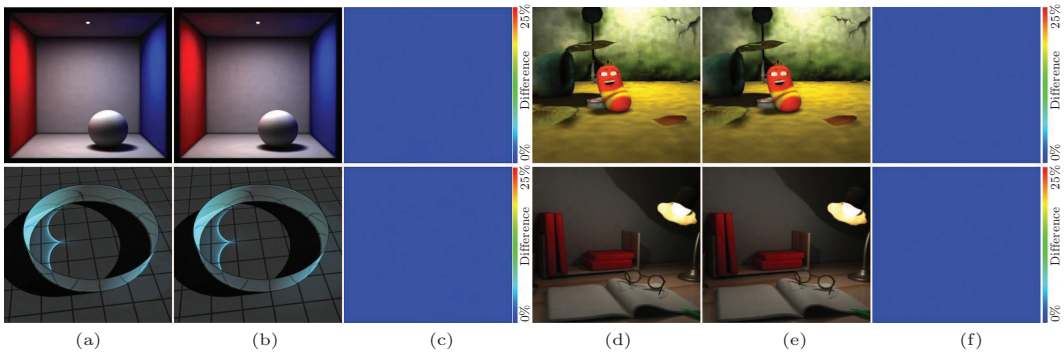


Fig. 7. Differences between our method and the traditional photon mapping. (a) (d) Results of our method. (b) (e) Results of traditional method. (c) (f) Differences between the two methods.

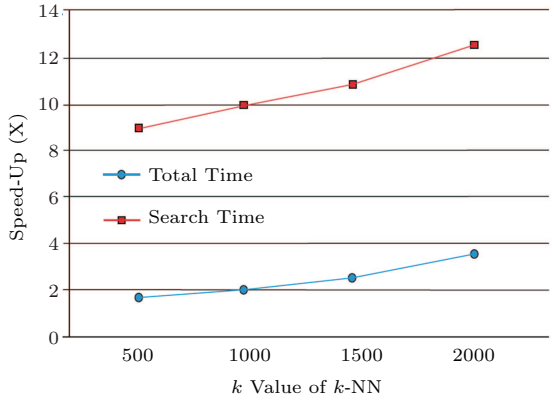


Fig.8. Performance of the cornell box with different k values.

6 Summary and Discussion

In summary, we presented a parallel method for radiance estimation on MIC. The method divides shading points into coherent clusters using a hierarchical aggregation method, and then searches the K-D tree for an initial neighbouring scope in radius R_0 at each cluster center. Compared to standard photon searching, we searched k-NN photons in the initial scope for all of the 16 shading points in a cluster, thereby allowing us to perform SIMD instructions in parallel on the MIC. We also presented a photon tracing approach on the MIC based on a QBVH tree by applying a 16-wide SIMD hardware as four lanes of four elements. The approach uses the KNC instructions of MIC to implement SIMD calculation. Although some exiting algorithms can utilize the parallelism of CPU with SSE instructions, we take advantage of the many-core computing power on MIC and our algorithm for the photon searching can also be implemented on CPU with SIMD instructions.

Our algorithm presents a balance achieved between rendering accuracy and efficiency as we use a classification and factorization search algorithm for k-NN searching. The classification step can complete the

search tree in one time, and the factorized algorithm applies the 16-wide SIMD units to maximize the performance of the parallel program. Meanwhile, the approach chooses a conservative initial radius for each cluster to make sure that k-NN photons of every shading point are included in the initial neighbouring scope.

Our approach has two main limitations. First, the cluster step introduces redundant computation. We used a hierarchical aggregation method to cluster 16 points in a set. Although using a four-layer iteration can get the results, the method has to calculate the distance between any two points. Therefore, the initial bundle should not be too large, and we set the maximum value to be 2 000 as test using 120 threads in parallel. Second, when rays are totally incoherent, it is hard to get a bundle with a number of shading points, which makes our factorized method burdensome. Fortunately, ray tracing pipeline will generally return appropriate bundles in the first bounce. Our current solution is that when we get a bundle with a small number of points, we achieve radiance estimation according to the original method in parallel without SIMD. Of course, this will reduce the acceleration efficiency. It would then be interesting to determine how to re-manage incoherent rays.

Finally, in this work, successfully searching k-NN photons in a K-D tree using SIMD units onto the MIC has many engineering challenges, although the device is more flexible compared with GPU. We still believe that this device has certain advantages in dealing with parallel speedup global illumination. Due to the different architectures and methods, direct comparison with GPU methods is difficult. However, the experimental data in [4, 6-7] shows that the acceleration data of our approach is not less than the data in GPU methods. Our classification and factorization search algorithm is a general algorithm, as it can be applied to progressive photon mapping in the same way. In progressive photon mapping, photon tracing and photon detection are processed in the same process, which makes all of the operators on the photon map can be run entirely on MIC. Our future job will be concentrated on these operations' optimization and parallelization on MIC.

References

[1] Jensen H W. Global illumination using photon maps. In *Proc. the Eurographics Workshop on Rendering Techniques*, June 1996, pp.21-30.

[2] Benthin C, Wald I, Woop S *et al.* Combining single and packet-ray tracing for arbitrary ray distributions on the Intel MIC architecture. *IEEE Transactions on Visualization and Computer Graphics*, 2002, 18(9): 1438-1448.

[3] Ma V C H, McCool M D. Low latency photon mapping using block hashing. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, June 2002, pp.89-99.

[4] Purcell T J, Donner C, Cammarano M *et al.* Photon mapping on programmable graphics hardware. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, June 2003, pp.41-50.

[5] Fabianowski B, Dingliana J. Interactive global photon mapping. *Computer Graphics Forum*, 2009, 28(4): 1151-1159.

[6] Gupte S. Real-time photon mapping on GPU. University of Maryland Baltimore County, 2011. http://www.cs-ee.umbc.edu/olano/635s11/sgupte1.pdf, Mar. 2015.

[7] Wang R, Wang R, Zhou K *et al.* An efficient GPU-based approach for interactive global illumination. *ACM Transactions on Graphics*, 2009, 28(3): Article No. 91.

[8] Wang B, Huang J, Buchholz B *et al.* Factorized point based global illumination. *Computer Graphics Forum*, 2013, 32(4): 117-123.

[9] Zhou K, Hou Q, Wang R *et al.* Real-time kd-tree construction on graphics hardware. *ACM Transactions on Graphics*, 2008, 27(5): Article No. 126.

[10] Hachisuka T, Jensen H W. Parallel progressive photon mapping on GPUs. *ACM SIGGRAPH ASIA 2010 Sketches*, 2010, Article No. 54.

[11] Collin C, Ribardiére M, Gruson A *et al.* Visibility-driven progressive volume photon tracing. *The Visual Computer*, 2013, 29(9): 849-859.

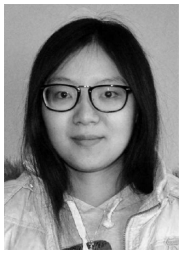
[12] Zhang Y, Dong Z, Ma K L. Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 2013, 19(8): 1317-1330.

[13] Aila T, Laine S. Understanding the efficiency of ray traversal on GPUs. In *Proc. Conference on High Performance Graphics*, Aug. 2009, pp.145-149.

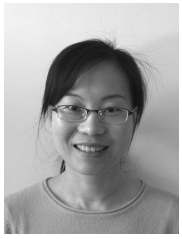
[14] Wald I, Slusallek P, Benthin C *et al.* Interactive rendering with coherent ray tracing. *Computer Graphics Forum*, 2001, 20(3): 153-165.

[15] Singh S, Faloutsos P. SIMD packet techniques for photon mapping. In *Proc. IEEE Symposium on Interactive Ray Tracing*, Sept. 2007, pp.87-94.

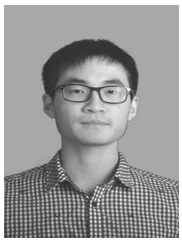
[16] Thakkur S, Huff T. Internet streaming SIMD extensions. *Computer*, 1999, 32(12): 26-34.



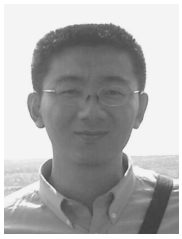
Chun-Meng Kang is a Ph.D. candidate in the School of Computer Science and Technology, Shandong University, Jinan. She received her B.S. degree in software engineering from Shandong University in 2011. Her research interests include photorealistic rendering and high performance computing.



Lu Wang is an associate professor in the School of Computer Science and Technology of Shandong University, Jinan. She received her Ph.D. degree in computer science and technology from Shandong University in 2009. Her research interests include photorealistic rendering, non-photorealistic rendering, geometric modeling, interactive design, etc.



Pei Wang is a master candidate in Software College, Shandong University, Jinan. He received his Bachelor's degree in software engineering from Chongqing University in 2012. His research interests include photorealistic rendering and high performance computing.



Yan-Ning Xu received his Ph.D. degree in computer science and technology from Shandong University in 2006, and is now an associate professor of the School of Computer Science and Technology, Shandong University. His research interests include virtual reality and human computer interaction, 3D modeling and rendering, etc.

Xiang-Xu Meng is a professor in the School of Computer Science and Technology of Shandong University, Jinan. He obtained his Ph.D. degree in computer science and technology from Institute of Computing Technology, Chinese Academy of Science, Beijing, in 1998. His researches focus on human-computer interaction and computer graphics theory and methods, virtual reality and virtual prototyping, grid computing and service computing, and manufacturing of information technology.