The Backward Photon Mapping for the Realistic Image Rendering[*]

Dmitry Zhdanov [0000-0001-7346-8155], Andrey Zhdanov [0000-0002-2569-1982]

ITMO University, 49 Kronverksky Pr., St. Petersburg, 197101, Russia

ddzhdanov@mail.ru, adzhdanov@itmo.ru

Abstract. The current paper is devoted to the methods of the realistic rendering methods based on the bidirectional stochastic ray tracing with photon maps. The research of the backward photon mapping method to account for both caustics and indirect illumination is presented. By using the backward photon maps authors reduced the amount of data that should be stored in the photon maps that allowed to speed up the process of the indirect luminance calculation. Methods used for constructing a tree of backward photon maps and methods of efficient parallelization used in algorithms of accumulation and forming the backward photon maps along with tracing forward and backward rays in the rendering process are considered. Methods to estimate the attained luminance error both for single image pixels and for the entire image with the designed rendering method are presented. The rendering results obtained with the use of the developed methods and algorithms are presented.

Keywords: Realistic Rendering, Photon Maps, Bidirectional Ray Tracing, Accuracy Estimation, Parallel Computing, Forward Ray Tracing, Backward Ray Tracing, Voxelization, Backward Photon Maps.

1 Introduction

Realistic rendering is the research area of modern realistic visualization, virtual prototyping, and virtual reality systems. It is used while solving a wide range of applied problems, including the realistic images forming, optical effects simulation, virtual prototyping of complex optical systems, etc.

The main algorithms that are used when implementing the realistic rendering algorithms are based on the stochastic ray tracing methods. In general, the task of the realistic rendering is to calculate the luminance of the observed scene at each point of the image. In its turn, it requires that the realistic rendering algorithms are be based on the physically correct laws of the light propagation and transformation. In 1985 James

[*] This work was funded by the Russian Science Foundation (project No. 18-79-10190).

Kajia proposed to use the following equation for calculating the visible luminance [1], which is also called the rendering equation:

$$L(\vec{p},\vec{v}_r) = \tau(t)\left(L_0(\vec{p},\vec{v}_r) + \iint^{2\pi} L_i(\vec{p},\vec{v}_i) f(\vec{p},\vec{v}_i,\vec{v}_r)\cos(\vec{n},\vec{v}_i)\, d\omega\right) \tag{1}$$

In formula (1) $\vec{p}$ is the observation point, $\vec{v}_r$ is the direction to the observer from the observation point, $\tau(t)$ is the transmittance between the observer and the observation point, $L_0(\vec{p},\vec{v}_r)$ is the luminance of the object self-luminance at the point of observation in direction to the observer, $L_i(\vec{p},\vec{v}_i)$ is the luminance of the ambient light in the solid angle $d\omega$, in the direction $\vec{v}_i$ to the observation point $\vec{p}$, $f(\vec{p},\vec{v}_i,\vec{v}_r)$ is the Bidirectional Scattering Distribution Function (BSDF) at the point $\vec{p}$ illuminated from direction $\vec{v}_i$, to the observer in direction $\vec{v}_r$, $\vec{n}$ is the normal to the surface in the observation point.

Since the rendering equation has the analytical solution only for a limited number of combinations of optical material properties and geometric shapes, a large number of numerical methods were developed to solve this equation. The most appropriate approaches are solutions based on the Monte Carlo ray tracing. These are forward, backward, and bidirectional ray tracing. These approaches use different solutions for calculation of the direct visibility luminance component ($L_0(\vec{p},\vec{v}_r)$), and direct, indirect and caustic luminance components ($\iint^{2\pi} L_i(\vec{p},\vec{v}_i) f(\vec{p},\vec{v}_i,\vec{v}_r)\cos(\vec{n},\vec{v}_i)\, d\omega$). Both direct and caustic luminances are formed by the first component of this infinite recursion and the main difference between them is that caustic luminance is formed through surfaces (or gradient media) that do not have diffuse surface properties. The light source luminance is simply scaled with the specular transmission coefficient to the observed surface ($\tau(t_r)L_0(\vec{p},\vec{v}_r)$). The indirect luminance component is defined by the rest of components of the infinite sum ($\iint^{2\pi} L_i(\vec{p},\vec{v}_i) f(\vec{p},\vec{v}_i,\vec{v}_r)\cos(\vec{n},\vec{v}_i)\, d\omega$).

The main methods which are used for the luminance calculation are forward, backward, and bidirectional ray tracing. These methods allow getting the unbiased solution of the rendering equation. However, these methods [2] might be not effective for scenes with complex indirect and caustic illumination. Traditional bidirectional ray tracing method [3] might be quite effective for scenes when several diffuse events occur on the way from the light source to the observed surfaces, however, it has very low effectivity in the case of caustic illumination, especially if light sources are rather small. Therefore, the most universal method for calculating the physically correct luminance of indirect and caustic illumination are the methods based on the photon maps [4].

In 1996 Henrik Wann Jensen proposed [5] to use the photon maps for calculation of the global illumination. The rendering algorithm that uses photon maps consists of the following three main steps: in the first step, rays are emitted from the light sources, propagated across the scene and form the distribution of photons on the surfaces of the scene; in the second step, photon maps are formed from the photon distribution; and at the last step, the camera reads the luminance distribution from the photons observed from the corresponding image pixels. The rendering process with the photon mapping method is shown on Fig. 1.

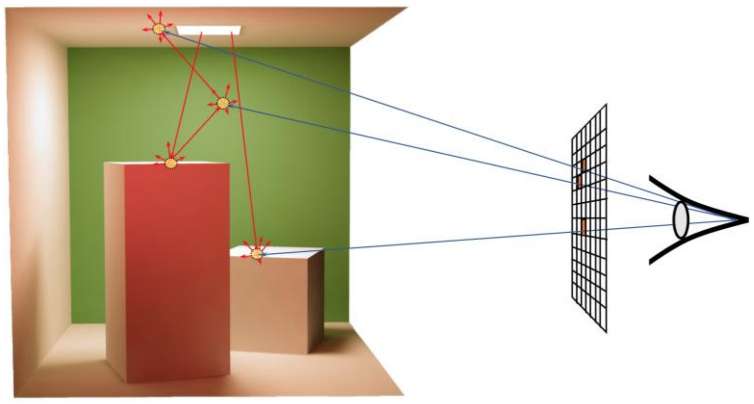

Fig. 1. The photon mapping method.

The observed luminance is the sum of the luminances of all photons observed on the backward ray paths:

$$L_{idc}(\vec{p},\vec{v}_r) \approx \frac{1}{\pi r^2}\sum_{i=1}^{K} f(\vec{p},\vec{v}_i,\vec{v}_r)\Delta\Phi(\vec{v}_i) \quad (2)$$

In formula (2) K is the number of observed photons, $\Delta\Phi(\vec{v}_i)$ is the photon flux in direction $\vec{v}_i$. This solution is biased since the luminance is averaged over the integration sphere of the radius r. This is the main disadvantage of the photon mapping method. There are two mainly used solutions to select the radius of the integration sphere. At first, the radius can be set during the forward ray tracing. In this case, it has a high probability to be either overestimated or underestimated, since at that moment there is no information on how this sphere will be observed. This can slow down the convergence of the rendering or cause the image to blur. Secondly, the radius of the sphere can be calculated when tracing backward rays. When a ray passes near a photon, which is determined using the corresponding voxel structure, the radius of the integration sphere is calculated based on the given solid angle of integration of the indirect or caustic luminance. The disadvantage of this approach is the complexity of the algorithm for finding the intersection of the ray with photons and, as a result, the slower rendering. Also, the forward photon mapping method generally requires storing the history of the ray trace path, i.e. photons on all diffuse surfaces of the scene. For "bounded and bright" scenes, this can lead to a significant increase in memory required to store photon maps that would slow down the rendering process as quite a large amount of data should be processed.

In 2005 Vlastimil Havran proposed [6] to use the backward photon mapping to accelerate the final gathering for calculation of the indirect illumination. In our opinion, this method deserves further research and development. So, in this paper, we present our results of the additional research of the backward photon mapping method and the corresponding methods for the efficient calculation of the indirect and caustic luminance components with backward photon mapping in a parallel computing environment.

2 Backward photon mapping

If we assess the physical correctness of the rendering process, then the method of backward photon maps is similar to the method of the forward photon maps. The luminance of the indirect and caustic illumination is calculated similarly to the method of forward photon maps with the formula (2). The fundamental difference is that instead of forming the illumination photon maps, the maps of how the illumination is be observed, i.e. visibility maps, are formed. Fig. 2. illustrates how backward photon maps or visibility maps work.

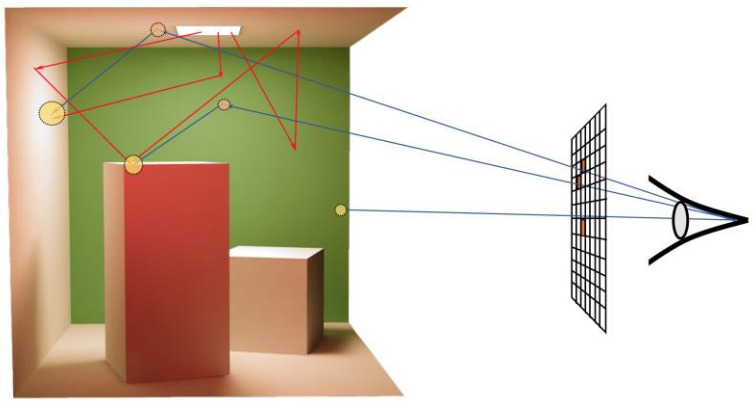

Fig. 2. The backward photon mapping method.

There are two fundamental differences between the methods of forward and backward photon maps in the physics of the rays propagation process and the accumulation of photon maps. At first, in the case of forward photon maps, the propagation of the light rays (or photons) transfers the luminous flux $\Delta\Phi(\vec{v}_i)$. In the case of backward photon maps, the ray emitted from the camera does not have the luminous flux. The backward ray transfers the transmittance $\tau(t)$ from the camera to the point where the luminance is calculated. Despite the different physical meanings of $\Delta\Phi(\vec{v}_i)$ and $\tau(t)$ for RGB and simple spectral modeling, there are no fundamental differences between them as both are vectors. However, if the polarization effects are taken into account then the structure of the $\Delta\Phi(\vec{v}_i)$ and $\tau(t)$ parameters change. If the polarization effect is represented by Stokes vectors, then each color component $\Delta\Phi(\vec{v}_i)$ becomes a Stokes vector of 4 components. And each of the color components of $\tau(t)$ turns into a 4x4 Muller matrix. As a result, if the rendering is performed with polarization effects in mind, then each color component of the backward photon must store a Mueller matrix that significantly increases the memory load for backward photon maps storage. However, as the polarization effect is not important in most of the scenes, it can be neglected.

The second important difference is in the method for calculating $BSDF(\vec{p}, \vec{v}_i, \vec{v}_r)$. According to the Helmholtz reciprocity principle, the following condition must be satisfied:

$$BSDF(\vec{p}, \vec{v}_i, \vec{v}_r) = BSDF(\vec{p}, \vec{v}_r, \vec{v}_i) \tag{3}$$

This condition is valid only for a limited number of surface optical properties models, for example, Lambert or Blinn-Phong. For most of the other models, including models obtained by measuring properties of the real-world objects, this condition is not satisfied. This means that the rendering result obtained with forward ray tracing will differ from the result obtained with backward ray tracing. Typically, the result obtained by the forward ray tracing is chosen as ground truth, so the backward ray tracing result must be matched with the forward ray tracing one.

To ensure the Helmholtz reciprocity principle is satisfied when implementing the ray scattering on a diffuse surface, we used the importance sampling corresponding to the BSDF function of the forward ray path, followed by compensation of $\tau(t)$ by the BSDF value in the backward path of the same ray. Since the variation of the BSDF in the forward and backward ray paths mainly does not exceed two times, the variation of the obtained $\tau(t)$ values also do not exceed two times.

The main advantage of the backward photon mapping method over the forward photon mapping method is the ability to limit the number of diffuse events occurred with ray before creating the backward photon in the map. This limitation of the diffuse ray tracing depth can significantly reduce the amount of data required to store backward photon maps, especially for “bounded and bright” scenes with a large number of diffuse reflections.

The research has shown that the optimal diffuse ray tracing depth varies between zero and one for most of the scenes. A further increase of the diffuse ray tracing depth leads to a slowdown of the rendering process without a noticeable improvement of the generated image quality. Even though from a formal point of view (based on the results of the image accuracy evaluation) the zero diffuse rays tracing depth seems to be preferable, for most of the scenes the optimal (based on the visual assessment of the image quality) diffuse ray tracing depth is equal to one. This difference is explained by the fact that at the zero diffuse ray path depth, the image is more biased, i.e. the sampling error caused by the limited radius of the integration sphere is more noticeable. Visually, this appears as the high graininess of the image and shadowing around the sharp edges of the scene objects. It can be noted that for each point in the scene, its own "optimal" depth of the diffuse ray tracing can be chosen, and this depth is determined by the variation of luminance at the image point (for example, due to the proximity of the observation point to the light source or the excess "sharpness" of the BRDF at the observed point). Moreover, the diffuse ray tracing depth can vary during the luminance calculation.

The second important advantage of the backward photon mapping method is the ability to determine the radius of the integration sphere when forming the backward photon maps. The radius of the integration sphere is based on the value of the acceptable luminance discretization error, i.e. in which solid angle the luminance would be averaged. For convenience, this solid angle can be defined based on the solid angle of one screen pixel as it is seen by the virtual camera. Sampling error is directly related to the kind of luminance that would be accumulated at the observation point. In the case of zero diffuse depth of the backward ray, all diffuse surfaces on the ray path accumulate only direct and caustic illumination before the first diffuse event. This means that the radius of the integration sphere should be kept minimal, about the size of the image pixel. After the first diffuse event occurs, only the indirect luminance is accumulated (the surface illumination also can be either direct or caustic) and the radius of the integration sphere can be expanded to a size of several pixels.

Since the radius of the integration sphere is already known at the moment of forming the backward photons map, the voxel accelerating structure (kd-tree) can be built not just for backward photon centers but for the whole integrating spheres. Studies have shown that building a voxel acceleration structure for spheres requires from one and a half to two times more time comparing to building a similar structure for their centers only, however, the search for the intersection of the forward ray with the photon maps is executed much faster if the voxel acceleration structure stores integration spheres comparing to the one storing centers only. Taking into account a large number of traced forward rays, the overall acceleration gain by using a voxel acceleration structure with integration spheres can reach one and a half times.

Also, a natural solution is to split the backward photon maps by scene objects, i.e. each scene object has its backward photon map, in which photons are accumulated when the ray hits this object. This separation does not increase the memory load when storing maps however allows us to speed up both the process of building the voxel acceleration structure and the intersection search of the ray with the integration spheres. Edge effects caused by the fact that illumination of the edge of one object cannot be transformed into the luminance at the edge of a neighboring object (even though they can be covered by the same sphere of integration), do not make a significant contribution to the variation of luminance, since the radius of the sphere of integration is rather small. Also, in some cases, the separation of backward photon maps by scene objects helps to avoid light leaking into unlit areas.

3 Parallelizing solutions for the backward photon mapping method

In the scope of the current research, the presented realistic rendering algorithm based on backward photon maps was implemented for multicore computers without GPU. The general scheme of the implemented algorithm is shown in Fig. 3.

This algorithm resembles the progressive photon mapping rendering algorithm. The main difference is that the backward rays are traced first with the light visibility and direct luminance calculation. Then the backward photon maps are voxelized and the accelerating structures are built. Subsequent forward ray tracing "throws" photons into backward photon maps and forms the indirect and caustic illumination maps. These maps are used to calculate the luminance of the caustic and secondary illumination. The size of caustic and indirect illumination maps is not large enough, therefore the process of "throwing" photons and calculating the luminance is repeated several times until the number of backward photons hits by direct rays becomes close to the number of backward photons. After completing the calculation of all the luminance components, the accuracy of the generated image is evaluated and, if the required accuracy was not achieved, then the whole process is repeated from the beginning.

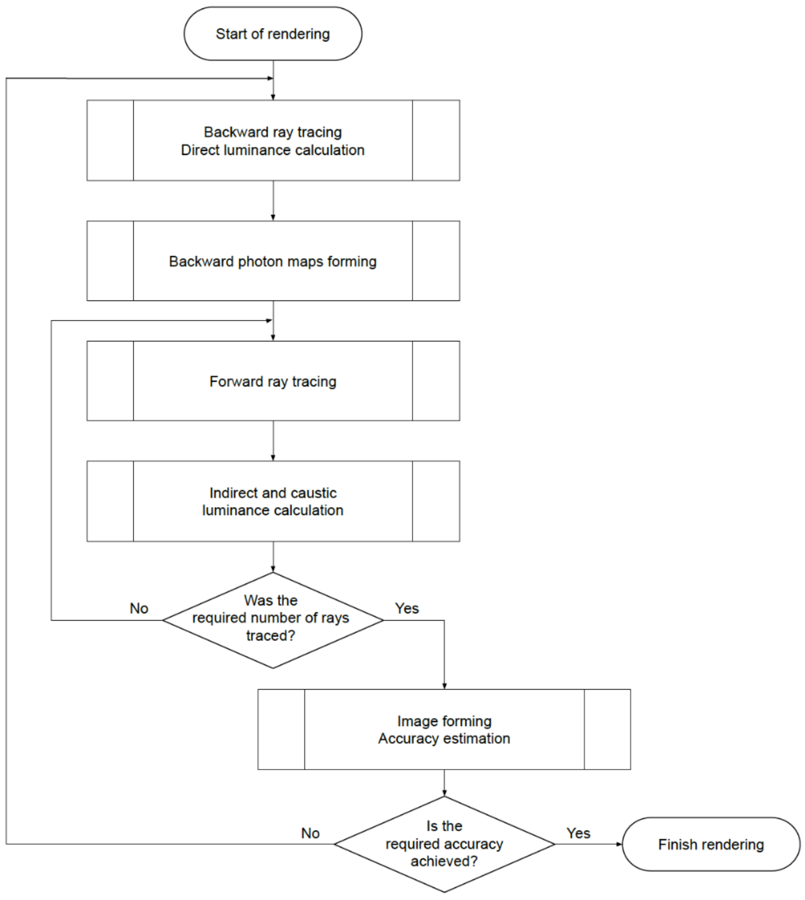
Start of rendering
Backward ray tracing
Direct luminance calculation
Backward photon maps forming
Forward ray tracing
Indirect and caustic luminance calculation
Was the required number of rays traced?
No
Yes
Image forming
Accuracy estimation
Is the required accuracy achieved?
No
Yes
Finish rendering

Fig. 3. Bidirectional ray tracing with backward photon maps algorithm.

For the efficient implementation of the rendering process on multicore computers, special algorithms were developed for more efficient parallelization of backward ray tracing, including the calculation luminance of the direct light visibility and direct illumination, construction of the accelerating structures, forward ray tracing, and calculation of the caustic and indirect illumination components.

The backward ray tracing is performed in blocks of 32x32, 64x64, or 128x128 pixels (depending on the number of threads used for parallel computing). Each thread renders its part of the image to avoid conflicts when writing to the shared image memory. Also, each thread forms its local photon map, which also avoids conflicts when writing photons into the map of the same scene object. In addition to the listed "classical" parallel methods, a method of parallelization by mask within one block was proposed. The idea of this method is illustrated in Fig. 4.

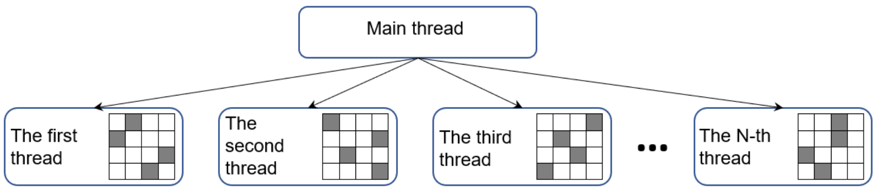

Fig. 4. The method of thread jobs distribution with a mask.

All threads perform the rendering of each block. However, within the range of one block, each thread traces rays that pass only through its part of the block pixels determined by the mask. This approach allows organizing a more uniform calculations load across all threads. This is useful for rendering optically complex scenes on multi-core computers. If a small part of the image contains objects that create long ray paths, for example, light-guiding devices, then the calculation of this part of the image will not "hang" and will not lead to a general underload of the CPU, since each area of the image is processed by all threads.

To create the accelerating structure for the backward photon map, kd-trees are used. As each scene object has its backward photon map and corresponding accelerating structure, the parallel voxelization of photon maps was implemented. The maps are sorted in descending order by the number of photons they contain and are queued for voxelization. Then first *K* (where *K* is the number of threads used in the simulation) of the largest maps are voxelized in parallel. As soon as one thread finishes voxelizing a map, it takes the next map from the queue and starts its voxelization. Since the number of objects in a scene is usually orders of magnitude greater than the number of threads, there is no noticeable decrease in the CPU load, and the acceleration of the voxelization process almost linearly depends on the number of CPU cores.

The next component of the parallel compute rendering system is forward ray tracing and calculation of the caustic and indirect luminance. The direct Monte Carlo ray tracing algorithm is simple enough for efficient parallelization. The main problem is the time required to write image data to the shared memory. To avoid memory access conflicts and synchronization, an algorithm was developed that splits the forward ray tracing and the calculation of the indirect and caustic luminance into two parts. The first part traces the forward light rays and searches for the intersection of these rays with the backward photon maps. To avoid memory conflicts when writing the result of the intersection of the ray with the backward photon map (sphere of integration), an additional map of forward photons is formed, containing a list of all spheres of integration that the given ray hit. This map is built for each separate thread. After a portion of the light rays has been traced (as a rule, the portion size is from 100,000 to 1,000,000 rays), this map is processed, and caustic and secondary luminance components are calculated. The calculation of these luminance components is performed in parallel when each thread chooses its part of the image (according to the masks that were formed for backward ray tracing) and calculate corresponding pixels luminance. Since the generated map of forward photons contains a reference to the backward photons and, as result, the indices of the screen pixels from which they were emitted, each thread can select only pixels that correspond to the screen area they process and calculate corresponding caustic and indirect luminance components.

It should be noted that after performing a series of rendering phases, from backward ray tracing to the accuracy estimation and the intermediate image forming, masks are rebuilt, re-distributing the backward rays across the image area.

4 Accuracy estimation

One of the important points of the realistic rendering, aimed for virtual prototyping of optically complex scenes and optical devices, is the correct estimation of the calculation accuracy. The estimation of the luminance calculation accuracy can be performed both for an individual pixel of the image or for the entire image.

Various approaches can be used to estimate accuracy. One of the simplest solutions is based on the progressive nature of luminance calculations. Rendering (from backward ray tracing to intermediate image forming) must be repeated many times to achieve a good quality image. Therefore, it is possible to create additional M independent intermediate images instead of one during the rendering process, then it is possible to estimate the accuracy of the total image by calculating the standard deviation between these images. This method was inherited from the algorithm used for evaluating the accuracy of calculated illumination maps in the Lumicept rendering system [7]. At the *i*-th rendering step, the rendering result is written to the intermediate image with the index (*i mod M*). In this case, the final image is the average of all images, and the accuracy of the pixel luminance RMS_i can be estimated as the standard deviation between the luminance of the pixels in *M* images. Then the accuracy δ over the entire image with size $w \times h$ can be estimated as follows:

$$\delta = \sqrt{\frac{\sum_{i=1}^{w\cdot h} {RMS_i}^2}{\sum_{i=1}^{w\cdot h} {L_i}^2}} \qquad (4)$$

In case if *M* is a small value, for example, 2, this method correctly estimates the average accuracy of the entire image. However, the pixel luminance estimation might be highly inaccurate. For a correct estimation of the separate pixel accuracy, the *M* value must be more than 10, which leads to a significant increase in the amount of memory required to store the intermediate images. Therefore, this algorithm was replaced by a more "economical" one which is based on the following formula:

$$RMS_i^2 = \frac{1}{N}\left(\frac{\sum_{j=1}^{N} {L_{i,j}}^2}{N} - \left(\frac{\sum_{j=1}^{N} L_{i,j}}{N}\right)^2\right) \qquad (5)$$

In formula (5) N is the total number of luminance calculation phases, $L_{i,j}$ is the luminance of the i-th pixel, calculated at the j-th calculation phase. As can be seen, to estimate the pixel error it is enough to accumulate only one additional value: the square of the pixel luminance over the calculation phases.

During the rendering process, the estimation of the current calculation accuracy is used not only to interrupt calculations when a required accuracy is achieved but also this estimation can be used as the density probability function when emitting backward rays through the image pixel. A larger error resulting in a higher probability of ray emission should lead to a reduction of the error in the corresponding pixel.

5 Results

The developed rendering method, based on the backward photon maps, was integrated into the Lumicept physically correct rendering package. Comparison with the prototype that was implemented basing on the stochastic ray tracing with classic forward photon maps showed that the backward photon maps method can significantly decrease the number of photons created by one ray and the total size of the photon maps while keeping the same number of traced rays. As a result, it also decreases the memory load of the rendering methods.

The backward and forward photons forming were tested on three scenes, these are the Cornell Box scene, the Room2 scene, and the Light Guides scene. Rendered images of test scenes acquired with the backward photon mapping method are shown in Fig. 5-7. Table 1 shows the average number of photons created by a single forward ray in the photon mapping method comparing to the number of backward photons created by one backward ray in the backward photon mapping method for the same scene.

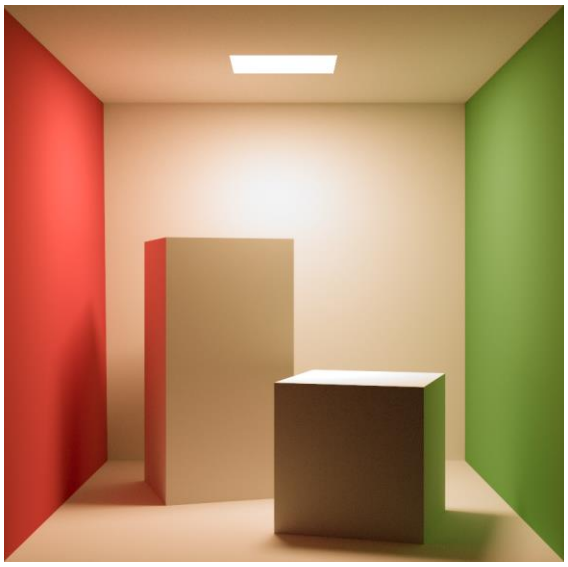

Fig. 5. Rendered images of the Cornell Box test scene.

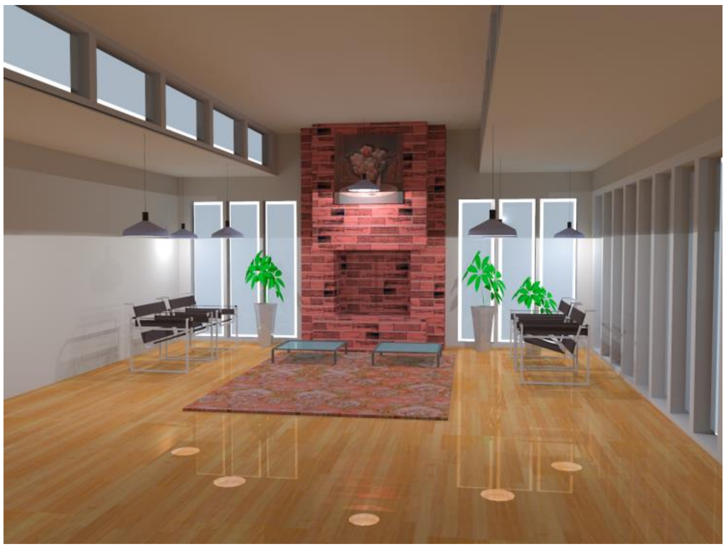

Fig. 6. Rendered images of the Room2 test scene.

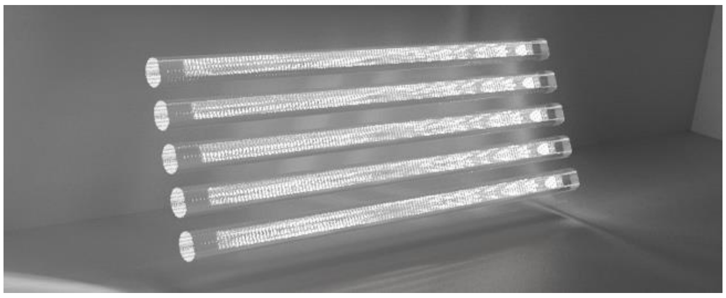

Fig. 7. Rendered images of the Light Guides test scene.

Table 1. The number of photons formed by one ray in the forward photon mapping and the backward photon mapping methods.

Scene	Photons per forward ray	Photons per backward ray	Photon map size reduction (times)
Cornell Box	3.45	1.59	2.17
Room2	7.6	1.93	3.94
Light Guides (light sources are inside light guides)	7.6	3.1	2.45
Light Guides (light sources are directed into light guides but are positioned outside of light guides)	5.6	3.1	1.81
Light Guides (light sources are inside light guides; the camera is directed into light guide side face)	7.6	5.9	1.29

As can be seen, the number of photons per ray depends on the position of the light sources and camera. The Light Guides scene with correct camera positioning shows an example of the extreme case for the backward photon mapping method, however, in real use cases the backward photon maps size is about two to three times smaller than the forward photon maps formed in the same scene with the same number of traced rays.

6 Conclusion

The developed method of the bidirectional stochastic ray tracing with backward photon maps provides physically correct rendering and high computational efficiency. It can be used not only for realistic rendering of the optically complex scenes but also for virtual prototyping of various kinds of optical systems, including observation systems, lighting systems, and virtual reality systems. This method can be effectively parallelized to be used on multicore computers or in distributed computing systems. The efficiency of the CPU implementation of the presented methods can be additionally increased if ray tracing, luminance calculation, photon maps forming, and processing methods are implemented with SSE, AVX, AVX2, and AVX-512 instruction sets. Currently, authors are working on rendering algorithms that would calculate the caustic and indirect illumination by forming the independent forward and backward photon maps for each of the illumination components. These algorithmic solutions might significantly improve the performance of realistic rendering.

References

1. Kajiya, J. T.: The rendering equation. In: SIGGRAPH '86 Proceedings, vol. 20, pp. 143-150, ACM, New York, USA (1986).
2. Veach, E.: Robust monte carlo methods for light transport simulation. Ph. D. thesis. Stanford, CA, USA (1998).
3. Georgiev, I., Krivanek, J., Slusallek, P.: Bidirectional light transport with vertex merging. In: SIGGRAPH Asia 2011, pp. 27:1-27:2, ACM, New York, NY, USA (2011).
4. Kang, C. et al:. A survey of photon mapping state-of-the-art research and future challenges. In: Frontiers Inf Technol Electronic Vol 17, pp. 185–199. (2016).
5. Jensen, H. W.:Global illumination using photon maps. In: Proceedings of the eurographics workshop on Rendering techniques '96, pp. 21–30. Springer-Verlag, London, UK (1996).
6. Havran, V., Herzog, G., Seidel, H.-P.: Fast Final Gathering via Reverse Photon Mapping. In: Proceedings of the Eurographics 2005, Vol. 24, Nr. 5, pp. 323-333. Blackwell Publishing, Oxford, UK (2005).
7. Lumicept – Hybrid Light Simulation Software, http://www.integra.jp/en.