Vis Comput (2012) 28:613–623
DOI 10.1007/s00371-012-0709-9

ORIGINAL ARTICLE

KD-tree based parallel adaptive rendering

Xiao-Dan Liu · Jia-Ze Wu · Chang-Wen Zheng

Published online: 1 May 2012
© Springer-Verlag 2012

Abstract Multidimensional adaptive sampling technique is crucial for generating high quality images with effects such as motion blur, depth-of-field and soft shadows, but it costs a lot of memory and computation time. We propose a novel kd-tree based parallel adaptive rendering approach. First, a two-level framework for adaptive sampling in parallel is introduced to reduce the computation time and control the memory cost: in the prepare stage, we coarsely sample the entire multidimensional space and use kd-tree structure to separate it into several multidimensional subspaces; in the main stage, each subspace is refined by a sub kd-tree and rendered in parallel. Second, novel kd-tree based strategies are introduced to measure space's error value and generate anisotropic Poisson disk samples. The experimental results show that our algorithm produces better quality images than previous ones.

Keywords Multidimensional space · Adaptive technique · Parallelization · Poisson disk · Anisotropic sampling

X.-D. Liu (✉) · J.-Z. Wu · C.-W. Zheng
National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences, Beijing, China
e-mail: lxdfigo@163.com

J.-Z. Wu
e-mail: wujiaze05@gmail.com

C.-W. Zheng
e-mail: cwzheng@ieee.org

X.-D. Liu
Graduate University of Chinese Academy of Sciences, Beijing, China

1 Introduction

Sampling technique concerns how to transform a continuous signal into a discrete representation with as much information as possible and reconstruction technique is its reverse process. Both of them are applied frequently in imaging, rendering, geometry processing. In photorealistic rendering, sampling and reconstruction techniques are used to do anti-aliasing, remove noises and generate high quality images with effects such as motion blur, depth-of-field and soft shadows. This kind of images always has different frequencies in different regions. Depending on the changing frequencies, the sampling method decreases the samples in the smooth regions and increases the samples in the rapid changing regions, which is known as adaptive sampling.

The adaptive sampling methods give better quality results than the images obtained by using stochastic sampling methods [1]. However, most of these methods only adaptively sample the image plane dimensions, but randomly sample the other dimensions. This kind of methods are hard to generate good images having effects of motion blur, depth-of-field or soft shadows. Recent advanced approaches consider non-image dimensions such as time [2] or area light's surface [3]. Hachisuka et al. propose a multidimensional adaptive sampling method [4], but it costs large computation time and a lot of memory.

Motivated by this observation, a kd-tree based parallel adaptive rendering algorithm is presented in this paper. Our algorithm includes the following contributions.

Kd-tree based parallelization Our algorithm uses kd-tree structure to separate the entire render space and considers each node of the kd-tree as a subspace which will be rendered in parallel. Owing to this novel parallelization strategy, the computation time decreases and the memory cost is under control.

Feature-focus error measurement According to the drawbacks of most local variance measurement methods, a new method is presented to measure the space variance which focuses on both local and global features. In addition, our method can also consider the optional depth and velocity features of the scene.

Anisotropic radius-varying Poisson disk sampling To render high quality image, a novel method is described to generate anisotropic radius-varying Poisson disk samples in multidimensional space. This variable Poisson disk radius depends on the kd-tree node's size and the anisotropic property depends on the node splitting process.

Noise removal Based on our kd-tree structure, a noise removal strategy is easily applied in our algorithm to remove the outlier samples, whose values are much larger or smaller than their neighbor samples.

2 Related work

Adaptive sampling Since Whitted first proposed to use an adaptive sampling method in Monte Carlo ray tracing [5], there are two main categories of adaptive sampling methods. Most researchers prefer to the first one which only adaptively samples the image plane. Early methods [1, 6, 7] generate adaptive samples relying on the measurement of local variance or attempt to use an density map of the image [8]. Overbeck et al. [9] use wavelet to analyze the frequency of the image and adaptively sample the area having high variance value in different scale resolutions. Greedy error minimization method given by Rousselle et al. [10] shows good results for adaptive sampling and reconstruction of the image. However, these methods only adaptively sample the image plane and ignore the non-image dimension information. They are hard to use for generating efficient effects such as motion blur, depth-of-field and soft shadows.

The second category methods are multidimensional adaptive sampling. Egan et al. [2] use a Fourier transform method to adaptively sample the multidimensional information and provide an excellent motion blur effect. Another method proposed by Soler et al. [11] uses Fourier transform to analyze the lens dimensions and improves the depth-of-field effect. Chen et al. [12] use depth map and variance value to sample and reconstruct the image of depth-of-field effect. Most of these methods are dimensional limited. They consider only one or two non-image dimensions. The temporal light field reconstruction [13] reconstructs multidimensional samples into high quality results, but it must have velocity and depth information. The multidimensional adaptive sampling method presented by Hachisuka et al. [4] uses kd-tree to split the space and generate good results. However, it costs a lot of memory and it is hard to generate high resolution images.

Poisson disk sampling Poisson disk distributions were used by early researchers [1, 14] to turn regular aliasing patterns of the image into noises. Yellot found that the photoreceptors in the retina of human eyes are distributed according to a Poisson disk distribution, which implies that Poisson disk distribution is effective for rendering [15]. To generate the Poisson disk distribution better and faster, several methods were proposed, such as dart throwing [14], tile Poisson disk sampling [8], multidimensional Poisson disk [16, 17], multi-class Poisson disk [18] and parallel Poisson disk sampling [19]. However, there is no research on generating Poisson disk distribution which is suitable for adaptive multidimensional sampling.

Anisotropic sampling In recent years the anisotropic property has been used to assist sampling method. McCool [20] gives an anisotropic diffusion algorithm to reduce the Monte Carlo noises. Li et al. [21] provide a method to generate anisotropic blue noise samples on the plane surface and the object surface. To judge anisotropic sample distribution, a differential domain analysis method [22] is proposed. However, using anisotropic distribution techniques to sample multidimensional space is still a challenge.

3 Overview

Adaptive sampling process needs the feedback of the current sampled samples in global render space to calculate the next optimal efficient sample in this space. Therefore, the adaptive sampling process is viewed as a serialized process and difficult to execute in parallel.

This paper proposes a kd-tree based parallel adaptive render algorithm. A kd-tree (short for k-dimensional tree) is a space-partitioning data structure for organizing samples in a k-dimensional space. The kd-tree is used to split the global space into subspaces in our algorithm. Each subspace is assigned an appropriate sample budget and is adaptively rendered in parallel. Our algorithm tends to use the optimal samples in the local subspace to approach the optimal samples in the global space of the original adaptive sampling process. Depending on this core idea, a two-level framework is proposed: the *multidimensional space initialization* stage and the *parallel adaptive rendering* stage.

In the *multidimensional space initialization* stage, the multidimensional render space of the scene is coarsely sampled and is split into several multidimensional subspaces. In the coarse sampling step, our algorithm calculates the number of initial samples which decides how many samples will be used in this stage. Random sample distribution is used to coarsely sample the scene. During the initial splitting step, a kd-tree is built on the entire multidimensional space. This

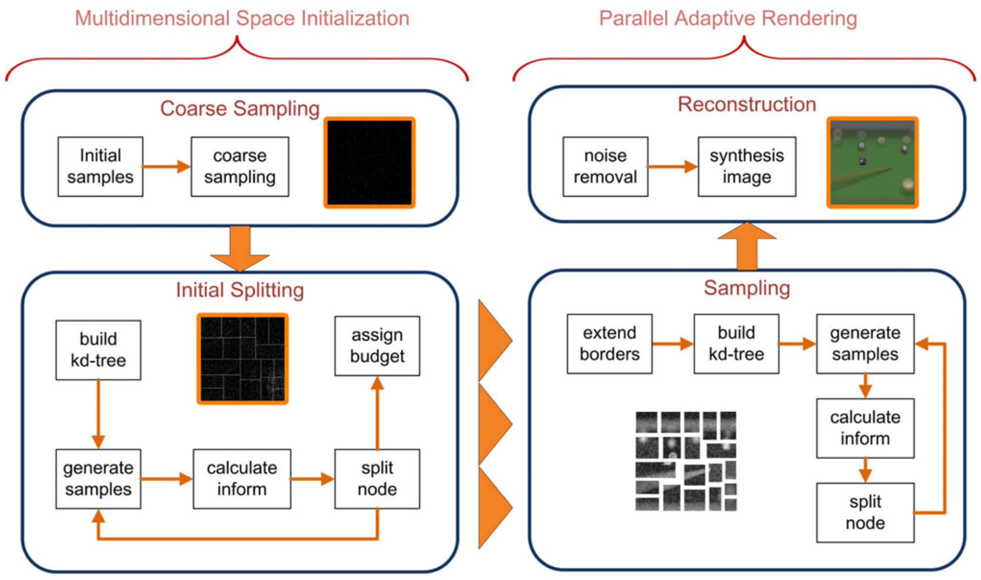


Fig. 1 A two-level framework: multidimensional space initialization; parallel adaptive rendering

kd-tree is repeatedly sampled and split until the termination criterion is fulfilled. After the node splitting is finished, each kd-tree node contains a multidimensional subspace. A balanced strategy is adopted to assign each subspace an appropriate sample budget for the following refined adaptive sampling.

In the *parallel adaptive rendering* stage, each subspace is sampled and reconstructed in parallel. In the sampling step, each subspace's borders are extended to avoid the aliasing of the entire sample distribution. After extending the borders, a sub kd-tree is built on each subspace. An anisotropic Poisson disk distribution is used to generate the samples. Each sub kd-tree is repeatedly sampled until the number of samples in this sub kd-tree arrives at its sample budget. In the reconstruction step, a noise removal strategy based on kd-tree structure is applied and the series of subimages are reconstructed from each sub kd-tree. The final image is combined by these subimages. Our algorithm is illustrated in Fig. 1 and each step will be explained in the following sections.

4 Multidimensional space initialization

In most parallel non-adaptive sampling methods [19], tile technique is used to generate samples in parallel, which separates the image plane into equal size pieces and renders them with same samples in parallel. If this technique is used in the parallel adaptive sampling method, it causes a terrible block effect in the final image. In this paper, we propose to adaptively split the entire multidimensional space into multidimensional subspaces of different size and assign each subspace an appropriate sample budget.

At the beginning of the multidimensional space initialization, the question of how many samples used in this stage need to be determined first, which are called initial samples. They are used to coarsely sample and separate the entire space. The number of initial samples is calculated by

Table 1 The major notations used in this paper

Notation	Description
Δ	multidimensional subspace
Ω	kd-tree node
ε_i	error value of node Ω_i
vol_i	volume of node Ω_i
r	radius of the Poisson disk
f	contribution function
D	dimension of the render space
N_Δ	count of subspaces
ω	heuristic parameter which controls the N_Δ
ρ	heuristic parameter which controls the r

$$N_{\mathrm{I}} = N_\Delta * U_\Omega = \left(\omega * \frac{N_{\mathrm{T}} * D}{M_{\mathrm{c}}} + 1\right) * U_\Omega \qquad (1)$$

Table 1 shows the description of the major notations. The number of initial samples N_{I} should be large enough to get necessary information of this scene for the following splitting. It is the product of the count of subspaces N_Δ and the user defined number U_Ω. N_Δ is the possible count of the subspaces which is determined by the sampling dimensions D and the total count of samples N_{T}. U_Ω is the user defined maximum sample number of each kd-tree node which should be big enough to get the node's information, we typically use 512 in this stage. The notation M_{c} represents the maximum number of samples that can be saved in the rendering computer, it depends on the computer memory. Coefficient $\omega \in [0, 1]$ controls the number of subspaces. Larger

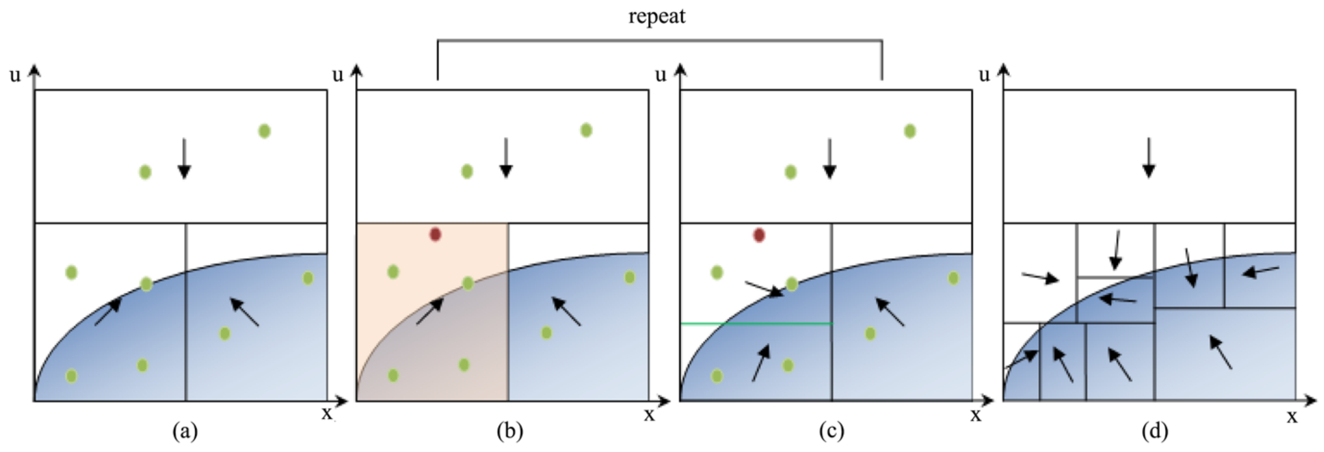


Fig. 2 (**a**) To adaptively sample the scene, a sub kd-tree is built on the subspace. (**b**) First, our algorithm lays a sample (the *red dot*) in the node which has the maximum error value. (**c**) If the node's samples beyond the maximum number, it is split along its longest dimension (the *green line*). During the repeatedly splitting, we calculate the anisotropic vectors. (**d**) After splitting, this subspace is combined by different size leaf nodes. The leaf nodes are assembled around the high error value area. The meaning of arrows will be explained in Sect. 5.3

ω causes more subspaces of small size and smaller ω causes less subspaces of large size.

After the initial samples are decided, random sample distribution is used to coarsely sample the entire multidimensional space and a kd-tree is built based on this space using these coarse samples. The kd-tree used in our algorithm has D dimensions as same as the entire render space dimensions which may contain image plane dimensions, time dimension or lens aperture dimensions in implementation. This kd-tree only saves samples in its leaf nodes. Each leaf node calculates its error value and anisotropic information. Only the node having the maximum error value can be sampled. After sampling a node, we recalculate its error value. When a leaf node contains more samples than U_Ω, it will be split into two child nodes along its longest dimension. An anisotropic Poisson disk sampling method (described in Sect. 5) is used to repeatedly sample the leaf node until all the initial samples have been used. Because each node will not contain more than U_Ω samples, the number of initial samples N_I determines the number of subspaces. After the adaptive splitting, each leaf node contains a multidimensional subspace and our algorithm assigns each subspace an appropriate sample budget (described in Sect. 5). In implementation, the initialization stage costs little time of the entire render time, because the initial sample number N_I is 1–5 percent of the total sample number.

5 Parallel adaptive sampling

The parallel adaptive sampling stage is as similar as the initial splitting stage. At the beginning, our algorithm extends each subspace's borders and builds sub kd-trees using samples which are already located in their domain during the initialization. Then, an anisotropic Poisson disk distribution is used to repeatedly sample the leaf node having the maximum error value in each subspace. Once a node has been sampled, its error value is recalculated. If the samples in a leaf node are more than the maximum number U_Ω, which is 4–8 here, this node is median-split along its longest dimension. Once a sub kd-tree generates a new leaf node, the error value and anisotropic vector of this node are calculated. This sampling step will not stop until the subspace's budget samples are all used. This process is shown in Fig. 2.

5.1 Poisson disk distribution

Poisson disk distribution is effective for sampling images. Our algorithm chooses to use the traditional dart-throwing method [14] to improve the image quality. The dart-throwing algorithm generates uniform random samples in the domain. If the distances of a sample from other samples are all greater than r, this sample is accepted. Otherwise, it is rejected. So each sample has a minimum distance r away from others, just like having a disk around it.

However, the adaptive principle makes it hard to decide a fixed minimum distance r in multidimensional space. We tend to use a Poisson disk technique with an variable minimum distance $r(vol)$. In the smooth regions of the scene, $r(vol)$ should be large. In the rapid changing regions, $r(vol)$ should be small. A novel method is proposed to calculate this adaptive disk radius in multidimensional space:

$$r_{\max}(vol) = 2 * \sqrt[D]{\frac{vol * \Gamma(\frac{D}{2}+1)}{\pi^{\frac{D}{2}} * U_\Omega}}, \tag{2}$$

$$\Gamma(x) = \begin{cases} 1 & x = 1 \\ \sqrt{\pi} & x = 0.5 \\ (x-1) * \Gamma(x-1) & x > 1, \end{cases} \quad (3)$$

$$r(vol) = \rho * r_{\max}(vol) \quad (4)$$

First, a maximum disk radius $r_{\max}(vol)$ of a multidimensional space is defined. This space is assumed to be fulfilled with several equal size multidimensional spheres. These spheres do not cross each other and they are as many as the samples which will be sampled in this space. $r_{\max}$ is the possible longest radius of these spheres. It is determined by the space's volume vol, samples in this space and the space dimension D. $\Gamma(x)$ function is easy to be calculated because D is an integer. Second, our algorithm multiplies $r_{\max}$ by a coefficient $\rho \in [0.5, 0.75]$ to find the suitable minimum distance $r(vol)$. In implementation, the input vol is the volume of a leaf node and the samples in this space is the user defined maximum sample number U_Ω of a leaf node. $r(vol)$ is the suitable Poisson disk radius for sampling kd-tree leaf nodes.

5.2 Error measurement

The error measurement is crucial for the adaptive sampling method. There are some classical error methods to measure the image's variance [1, 7], but they only focus on the local features and ignore the global ones. Overbeck et al. [9] use wavelet analysis to measure both the global and local variances, but they only consider the image plane. We present a novel error measurement strategy which makes full advantage of the kd-tree structure and focuses on both local and global features in multidimensional space. This strategy also takes into account the optional depth and velocity features of the scene.

$$\epsilon_j \approx \frac{vol_j}{N_j} \sum_{s \in \Omega_j} \frac{|f(s) - \tilde{f}_j|}{\tilde{f}_j} * F_x, \quad (5)$$

$$\tilde{f}_j = \alpha_f * \bar{f}_j + (1 - \alpha_f) * \tilde{f}_i \quad \alpha_f \in [0, 1] \quad (6)$$

The error value ϵ_j represents the variance of the space in leaf node Ω_j. It is calculated by combining the differences between the contribution f of each sample in Ω_j and the ideal value $\tilde{f}_j$ of Ω_j. It is also multiplied by the volume vol_j of the node Ω_j which results in putting more samples in the large nodes. The ideal value $\tilde{f}_j$ is a quantity evaluated by iteration. At each iteration, it is calculated by $\bar{f}_j$ and $\tilde{f}_i$. $\bar{f}_j$ is the mean contribution of node Ω_j and $\tilde{f}_i$ is the ideal value of node Ω_j's parent Ω_i. The coefficient α_f controls the measurement of error value. If α_f is smaller, $\tilde{f}_i$ is closer to its parents' mean contributions and the error value is more sensitive to the global feature. Otherwise, it is more sensitive to the local feature.

In implementation, most scenes details always have similar size features and we define α_f according to a Gauss distribution $\exp(-|vol - featuresize|^2/\delta^2)$. At the beginning, the node's volume vol is large and its error value is more sensitive to its global feature. During the splitting, the node's volume becomes smaller and its error value becomes more sensitive to its local feature. At last, some node's volume becomes too small and their error values become more sensitive to its global feature again.

The F_x is an optional factor. If there are more information about the samples such as speed or depth, our error measurement method use this optional factor to control the sampling distribution to pay more attention on the depth-of-field or the motion blur effects of the scene. In the depth-of-field effect case, the defocused area is much more blurring than the focused area. The pixels of the defocused area share more samples than the focused area. Our algorithm uses the following equation as F_x to put more samples near the focus space:

$$F_{\text{dep}} = \frac{1}{\left|\frac{C_1}{focaldis} - \frac{C_1}{dis}\right| * lensradius} \quad (7)$$

F_{dep} depends on the camera's coefficient, the focus distance $focaldis$ and the radius of lens aperture $lensradius$. The input value dis is the node's current distance from the camera. To improve motion blur effect, our algorithm uses Eq. (8) to put more samples on the moving object.

$$F_{\text{mot}} = x * t + y * t + \left|F_{\text{dep}}(dis) - F_{\text{dep}}(dis + z * t)\right| \quad (8)$$

F_{mot} depends on the depth-of-field factor and the speed of the current node. x, y, z are the node's speed on different axes which are calculated by summing its samples' speed value. t is the time dimension length of this node.

5.3 Anisotropic technique

Most previous work have focused on isotropic sampling and there are few research in anisotropic sampling, specially in multidimensional anisotropic sampling. Anisotropic sampling means putting samples according to the changing contribution of their domain. In this paper, the anisotropic information is obtained by the order of node splitting. It is used to help locating samples. Each leaf node contains an anisotropic vector which represents the changing of the contribution around it. When a leaf node needs to be split, it means that the light contribution in this node changes rapidly. The anisotropic vectors of the new leaf nodes are generated by adding their parent's anisotropic vector with an additional vector. If the error value of the child node is larger than its parent's, the additional vector points from the parent to its child. Otherwise, it points from the child to its parent. The length of this additional vector is determined by the difference of the error value between the parent node and its child. This flow is displayed in Fig. 2.

In the sampling step, the anisotropic information is used to assist the sampling (in Fig. 3). An earlier research [21]

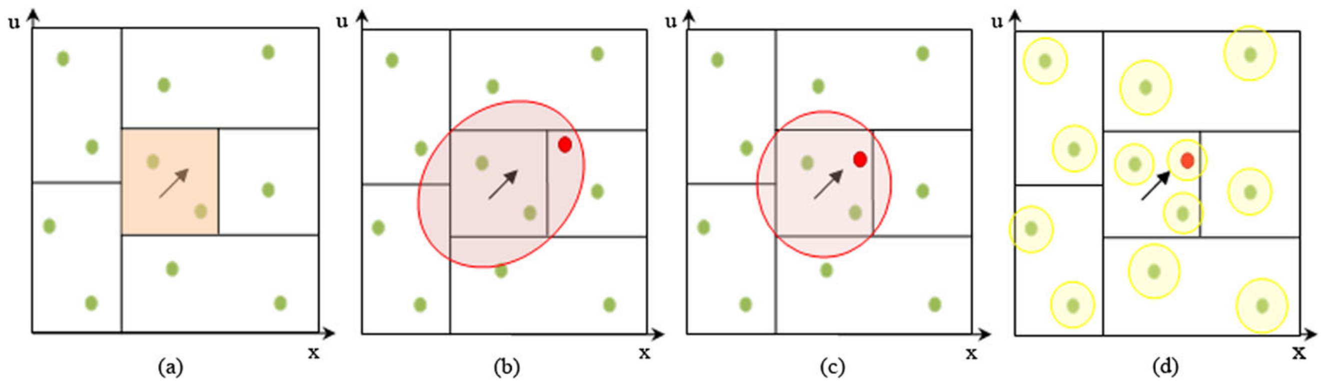


Fig. 3 (**a**) The maximum error value node is chosen (the *red region*). (**b**) Our algorithm generates the multidimensional ellipsoid using the anisotropic vector and locates a randomized sample in it (the *red dot*). (**c**) The ellipsoid is wrapped to a sphere around the node. (**d**) The sample is accepted, if its new location satisfies the Poisson disk condition. The minimum radius $r(vol)$ of this node is decided by Eq. (4)

illumines us a way to generate anisotropic samples. Before sampling the leaf node, our algorithm uses this node's multidimensional anisotropic vector to construct a multidimensional ellipsoid around it. After locating a uniform random sample in this ellipsoid, the ellipsoid is warped to an sphere around this leaf node. This sample's new location is checked by the Poisson disk condition.

5.4 Parallelization

To sample and reconstruct in parallel, each subspace needs a reasonable sample budget before *parallel adaptive rendering* stage. After initial splitting step, all leaf nodes have similar error values. Each subspace is assumed to be split B_Δ / U_Ω times in the following stage. The sample budget number B_Δ of each subspace is evaluated by

$$B_\Delta = \frac{\epsilon_i}{E_{\text{total}}} * (N_{\text{T}} + N_\Delta * U_\Omega) - U_\Omega \tag{9}$$

N_{T} is the total count of samples. ϵ_i is the error value of the subspace which is the leaf node of the kd-tree after initial splitting step. $E_{\text{total}} = \sum \epsilon_i$ is the sum of all the error values. N_Δ is the count of subspaces and the user defined U_Ω means the maximum number of the subspace.

Another problem caused by parallelization is the distribution of samples. When each subspace is rendered in parallel, it is hard to maintain Poisson disk distribution near the border area of each subspace. To advance the distribution near the border area, our algorithm slightly extends each subspace's borders. The extent distance d is given by Eq. (4), in which uses the sample budget B_Δ instead of U_Ω. The extent area is sampled by fake samples and will not be reconstructed (in Fig. 4). The fake sample only has the location information and does not need to get the light contribution.

In implementation, we use fixed threads to implement our algorithm. Each thread renders a subspace. If the threads rendered in parallel are fixed, the subspace's size can be changed to control the memory cost which is influenced by parameter ω. Larger size subspaces cause higher memory cost. Because the memory cost is under control, our algorithm can render higher resolution image which previous multidimensional adaptive method cannot.

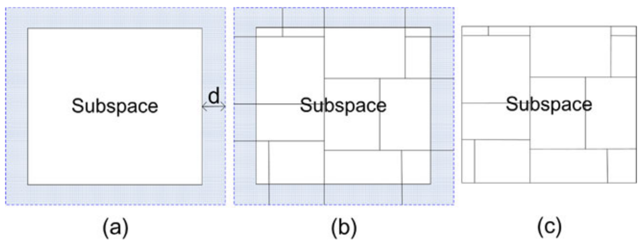


Fig. 4 (**a**) Each subspace's borders are extended by a distance d. (**b**) The extent area (the *blue area*) is part of the subspace and is split as same as the inner space. (**c**) The extent area will not be reconstructed

6 Reconstruction

Before the reconstruction, each leaf node is split until each node contains only one sample. This splitting only separate the space of each node without any calculation. It copies each node's error value and anisotropic information to its child nodes.

6.1 Noise removal

Based on the kd-tree structure, we can easily apply a noise removal step in our algorithm. This removal method is implied by DeCoro's approach [23]. The difference between the contributions of samples is used to find out the noise. First, each sample needs to be organized with its k-nearest neighbors as a group and this sample is considered as a

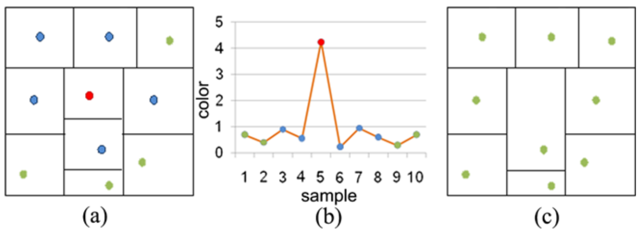


Fig. 5 (**a**) The leader sample (in *red*) and its k-nearest neighbors (in *blue*) are organized to a group. (**b**) Our algorithm calculates the variance of this group and the difference between the leader and its neighbors. (**c**) If this difference is much larger than the variance, this leader is considered as a noise sample and is removed

leader. Second, we calculate the color variance of each group and the difference between the leader and its neighbors. The variance of each group is the mean value of all members' differences. The difference is the contribution between a sample and the mean contribution of its group. If this leader sample's difference is much larger or smaller than its group's variance, this leader is considered as a noise and is removed (Fig. 5). After removing the noise sample, the empty leaf node which contained this sample is combined with its brother.

6.2 Image synthesis

The final step uses the contribution of each sample to evaluate the real image function. Our algorithm reconstructs the image by integrating the contribution of every node in each subspace. The core idea is shown in Eq. (10):

$$L(x,y) = \sum_{\Omega \in P(x,y)} \int_{\Omega} f(x,y,u_1,\ldots,u_n)\, du_1 \cdots u_n$$
$$\approx \sum_{\Omega \in P(x,y)} V_\Omega L_\Omega = \sum_{\Delta \in P(x,y)} \sum_{\Omega \in \Delta} V_\Omega L_\Omega \qquad (10)$$

$L(x,y)$ is the pixel value in image coordination (x,y) combined by the contribution of every part of node Ω in multidimensional space $P(x,y)$. f is the light contribution in multidimensional coordination $(x,y,u_1,\ldots,u_n)$. L_Ω and V_Ω are the contribution and volume of node Ω in subspace Δ.

The entire space is combined by all the sub kd-trees' leaf nodes. Depending on the samples in each subspace, their contributions are integrated into subimages. Each sample's node has a volume and it is a hyper-rectangle which may across several dimensions. The sample's contribution is timed by its node's volume. The result is considered as the integrated contribution of this node and the subimage is combined by all its subspace's node contribution. In additional, if there are more information of samples such as speed or depth, the time and lens aperture dimensions can be integrated by an advanced method which introduced by Lehtinen et al. [13]. In time dimension, each sample adds its contribution to every pixels it passes along the time dimension length of its node. In lens aperture dimensions, each sample is dispersed by the lens. The contribution of this sample is added to its dispersed pixels. After reconstructing all the subimages, we combine them into the final image.

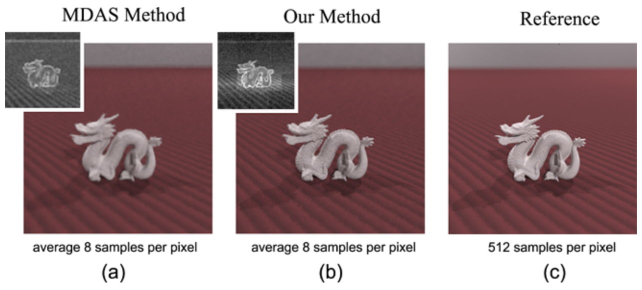


Fig. 6 The 300 × 300 resolution dragon image with four dimensions is rendered by our algorithm and MDAS method. (**a**) MDAS method costs 221 seconds and 965 MB memory. (**b**) Our approach costs 37 seconds and 312 MB memory. (**c**) Compared to the reference image, MDAS method is blurrier on the small sharp features

7 Results

Our algorithm and previous approaches are implemented on top of LuxRender [24]. The results were generated on a Intel Xeon CPU X5450 system at 3.0 GHz, with 2 GB of RAM, using eight threads. In this section, our technique is compared with previous adaptive methods in rendering effects such as motion blur, depth-of-field and soft shadows. The memory usage and time consumption are also compared between our algorithm and MDAS, which is the state-of-the-art method in multidimensional sampling domain. In addition, we give the effect of the noise removal technique and explain the parameters used in our approach.

7.1 Algorithmic improvement

Because MDAS method [4] costs a lot of memory, it cannot generate a whole high resolution image. MDAS method and our algorithm are compared by rendering a 300 × 300 resolution scene with depth-of-field effect. Our algorithm is faster than the MDAS method and costs less memory (Fig. 6). According to the sample distribution, our algorithm puts more samples near the focus distance region and is more sensitive to the edge.

Figure 7a compares the sample distribution between our algorithm and MDAS. Our algorithm puts more samples on the edges of the image. In Fig. 7b, we illustrate the problem of groove effects caused by sampling in parallel. It is improved by extending each subspace's borders. Figure 7c

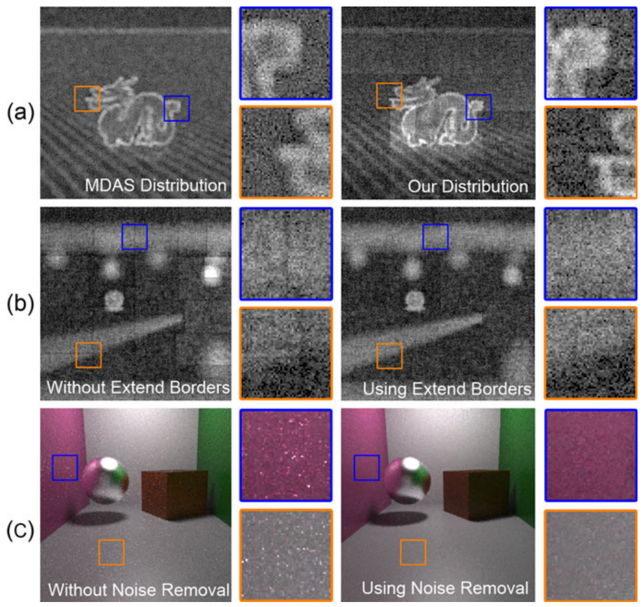


Fig. 7 (**a**) Our sample distribution puts more samples on the edges than the MDAS method. (**b**) After using the extending border technique, the grooves in the sample distribution are extinguished. (**c**) The *right image* has no high light pixel after using the noise removal

shows the effect of noise removal in our algorithm. Compared with the original image, our algorithm removes the high light samples clearly.

7.2 Examples

The results synthesized by our algorithm and previous methods such as low discrepancy approach, Mitchell adaptive sampling [1] and MDAS technique [4] are shown in Fig. 8. Mitchell and MDAS techniques are not rendered in parallel, because the traditional adaptive sampling process is a serialized process.

Scenes The experiments of three scenes use a wide range of effects. The first scene (Fig. 8a) of motion blur effect contains three moving balls. All its images were rendered at the resolution 400 $\times$ 400 and eight samples per pixel. The second scene (Fig. 8b) contains a billiard pool of depth-of-field and soft shadows effects. The camera focuses on the purple ball in the center. All its images were rendered at the resolution 1024 $\times$ 1024 and four samples per pixel. The third scene (Fig. 8c) is an outdoor scene of motion blur and depth-of-field effects. The car in the center is fixed and the background moves. All the images of this scene are rendered at the resolution 1024 $\times$ 1024 and 16 samples per pixel. Because MDAS method costs too much memory, the first scene is rendered at low resolution and the other two scenes are rendered by MDAS's tile technique [4] which renders only part of the whole image. The cost time, visual image quality and the mean square error (MSE) are compared in each image.

Comparisons The first scene (Fig. 8a) shows that our algorithm is able to find and sample the motive regions. In strong motive regions, our method is better than low discrepancy method and Mitchell's adaptive method which generate more noises. In this motion blur scene, the result of MDAS method is as good as ours, but our algorithm costs less time and memory because of our two-level framework.

In the second scene (Fig. 8b), our algorithm generates good effects of depth-of-field and soft shadows. In contrast with low discrepancy method, Mitchell adaptive sampling gains little benefit in these effects. MDAS technique reconstructs a better soft shadows effect, but it causes aliasing in the high light dispersion circles. Because MDAS does not remove the outlier samples and they disturb its anisotropic reconstruction. The reference image is rendered by random sampler with 1024 samples per pixel. Compared with the reference image, our algorithm is able to sample and reconstruct the image with smoother results and less noises.

In the car scene (Fig. 8c), the experiments compare our approach with others in a strong motion blur region and a depth-of-field region. Because of the parallel and sampling strategies in this article, our method synthesizes less noise and smoother image than other methods with the equal samples per pixel. Different from MDAS's tile technique which separates the image plane into pieces and renders them one by one, our algorithm renders the entire image in parallel which is easy to implement and has no block effect results.

7.3 Discussion

The parameters ω (in Eq. (1)) and ρ (in Eq. (4)) are analyzed by rendering the dragon scene with different values of them (Fig. 9). The parameter ω influences the number of subspaces. If our algorithm uses certain samples and threads to render a image, such as four samples per pixel and eight threads, there is an inverse relationship between subspaces and memory cost. The parameter ρ helps us to find the suitable Poisson disk of each node's space. If ρ is too large, it causes lots of refused samples during dart-throwing sampling. If ρ is too small, the distribution does not have blue noise property which affects the mean square error.

Limitations To sample each subspaces in parallel, we present a strategy to assign each subspace a sample budget. But this strategy may cause block effect in sampling distribution (Fig. 7a). Like most adaptive algorithm, our algorithm coarsely samples the scene at first. It may miss thin or small features in the scene.

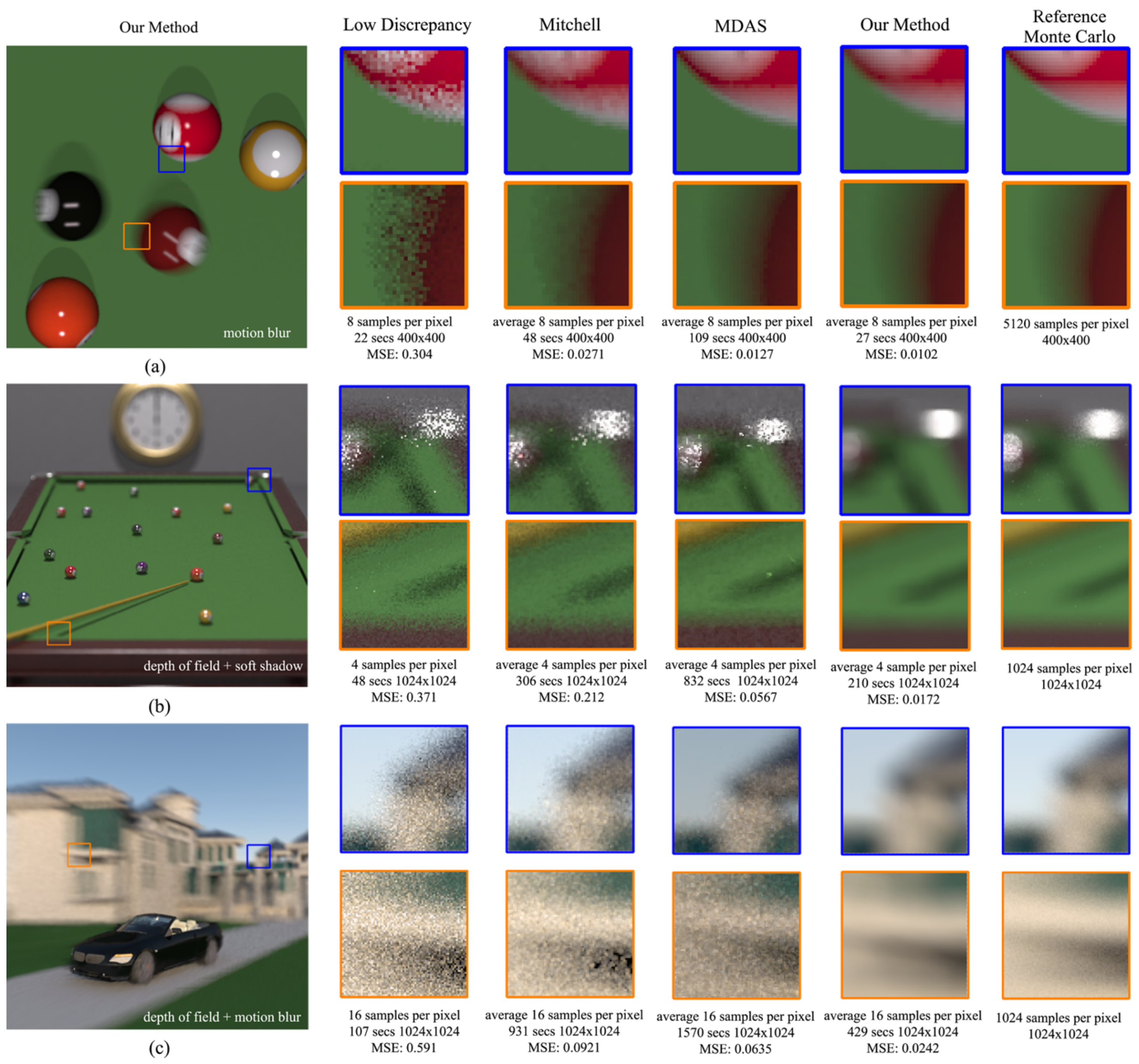

Our Method
Low Discrepancy
Mitchell
MDAS
Our Method
Reference Monte Carlo
motion blur
8 samples per pixel 22 secs 400x400 MSE: 0.304
average 8 samples per pixel 48 secs 400x400 MSE: 0.0271
average 8 samples per pixel 109 secs 400x400 MSE: 0.0127
average 8 samples per pixel 27 secs 400x400 MSE: 0.0102
5120 samples per pixel 400x400
(a)
depth of field + soft shadow
4 samples per pixel 48 secs 1024x1024 MSE: 0.371
average 4 samples per pixel 306 secs 1024x1024 MSE: 0.212
average 4 samples per pixel 832 secs 1024x1024 MSE: 0.0567
average 4 sample per pixel 210 secs 1024x1024 MSE: 0.0172
1024 samples per pixel 1024x1024
(b)
depth of field + motion blur
16 samples per pixel 107 secs 1024x1024 MSE: 0.591
average 16 samples per pixel 931 secs 1024x1024 MSE: 0.0921
average 16 samples per pixel 1570 secs 1024x1024 MSE: 0.0635
average 16 samples per pixel 429 secs 1024x1024 MSE: 0.0242
1024 samples per pixel 1024x1024
(c)

Fig. 8 (**a**) The first scene is a simple motion blur scene. The results show our algorithm is able to find and sample the motion regions better and faster than the other approaches. (**b**) The billiard pool scene shows depth-of-field and soft shadows effects. With equal samples per pixel, our algorithm generates an image with significantly less noises than the other methods. (**c**) The last scene is a complex scene of motion blur and depth-of-field effects. Our technique is able to sample and reconstruct the regions that are out of focus and strongly motive while other samplers are more noisy in these regions

8 Conclusion and future work

This paper presents a kd-tree based parallel adaptive rendering algorithm. A two-level framework is proposed to control the memory cost and reduce the time computation by parallel rendering. Based on the kd-tree structure, an anisotropic Poisson disk sampling method and a novel error measurement strategy are proposed to improve the sampling. Our algorithm renders high quality images of motion blur, depth-of-field and soft shadows effects.

According to the current render engine, our parallel algorithm is based on the multithreaded CPU technique. Once a GPU-based render engine has been developed, we plan to accelerate our algorithm using GPU to achieve read-time rendering. We also plan to develop a more efficient adaptive algorithm to accelerate the generation of bokeh effects [25],

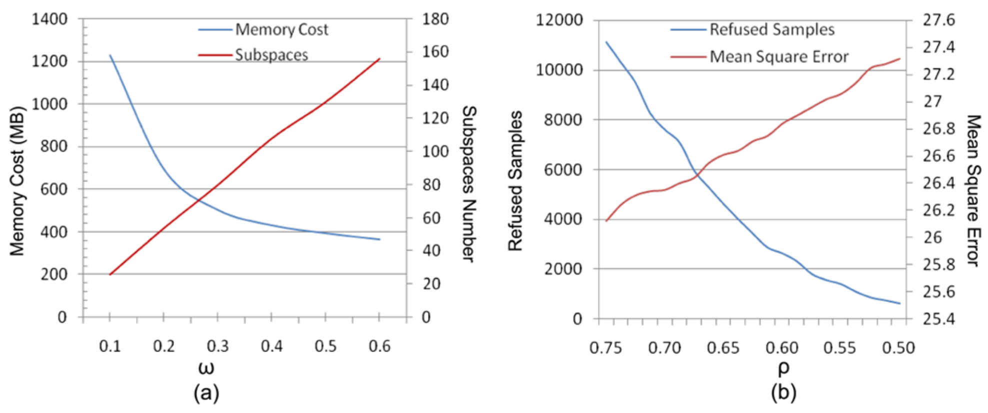


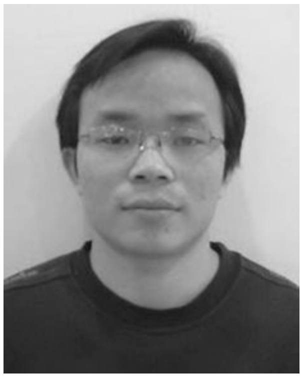
Fig. 9 (**a**) The memory cost decreases and the number of subspaces increases, when the parameter ω increases. (**b**) The parameter ρ influences the refused samples and the sampling distribution

with the anisotropic characteristic of sample distribution relevant with the bokeh pattern.

References

1. Mitchell, D.P.: Generating antialiased images at low sampling densities. In: Computer Graphics Proceedings. Annual Conference Series, ACM SIGGRAPH, vol. 21, pp. 65–72. ACM Press, New York (1987)
2. Egan, K., Tseng, Y.-T., Holzschuch, N., Durand, F., Ramamoorthi, R.: Frequency analysis and sheared reconstruction for rendering motion blur. ACM Trans. Graph. **28**(3), 1–13 (2009) (Proceedings of the SIGGRAPH Conference)
3. Egan, K., Hecht, F., Durand, F., Ramamoorthi, R.: Frequency analysis and sheared filtering for shadow light fields of complex occluders. ACM Trans. Graph. **30** (2001)
4. Hachisuka, T., Jarosz, W., Weistroffer, R.P., Dale, K.: Multidimensional adaptive sampling and reconstruction for ray tracing. ACM Trans. Graph. **27**, 1–10 (2008) (Proceedings of the SIGGRAPH Conference)
5. Whitted, T.: An improved illumination model for shaded display. Commun. ACM **23**, 343–349 (1980)
6. Mitchell, D.P.: Spectrally optimal sampling for distribution ray tracing. In: Computer Graphics Proceedings. Annual Conference Series, ACM SIGGRAPH, vol. 25, pp. 157–164. ACM Press, New York (1991)
7. Rigau, J., Feixas, M., Sbert, M.: Refinement criteria based on f-divergences. In: Eurographics Workshop on Rendering, pp. 260–269 (2003)
8. Ostromoukhov, V., Charles, D., Pierre-Marc, J.: Fast hierarchical importance sampling with blue noise properties. ACM Trans. Graph. **23**, 488–495 (2004)
9. Overbeck, R.S., Donner, C., Ramamoorthi, R.: Adaptive wavelet rendering. ACM Trans. Graph. **28**(5), 1–12 (2009) (Proceedings of the ACM SIGGRAPH Asia Conference)
10. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive sampling and reconstruction using greedy error minimization. In: ACM SIGGRAH Asia (2011)
11. Soler, C., Subr, K., Durand, F., Holzschuch, N., Sillion, F.: Fourier depth of field. ACM Trans. Graph. **28**(2), 1–12 (2009) (Proceedings of the SIGGRAPH Conference)
12. Chen, J., Wang, B., Wang, Y., Overbeck, R.S., Yong, J.-H., Wang, W.: Efficient depth-of-field rendering with adaptive sampling and multiscale reconstruction. Comput. Graph. Forum (2011). doi:10.1111/j.1467-8659.2011.01854.x
13. Lehtinen, J., Alia, T., Chen, J., Laine, S., Durand, F.: Temporal light field reconstruction for rendering distribution effects. ACM Trans. Graph. **40**, 10 (2011)
14. Cook, R.L.: Stochastic sampling in computer graphics. ACM Trans. Graph. **5**(1), 51–72 (1986) (Proceedings of the SIGGRAPH Conference)
15. Yellot, J.I.: Spectral consequences of photoreceptor sampling in the rhesus retina. Science **221**, 382–385 (1983)
16. Gamito, M.N., Maddock, S.C.: Accurate multidimensional Poisson-disk sampling. ACM Trans. Graph. **29** (2009). doi: 10.1145/1640443.1640451
17. Ebeida, M.S., Mitchell, S.A., Patney, A., Davidson, A.A., Owens, J.D.: A simple algorithm for maximal Poisson-disk sampling in high dimensions. Comput. Graph. Forum **31**(2) (2012)
18. Wei, L.-Y.: Multi-class blue noise sampling. ACM Trans. Graph. **29** (2010) (Proceedings of the SIGGRAPH conference). doi: 10.1145/1778765.1778816
19. Wei, L.-Y.: Parallel Poisson disk sampling. ACM Trans. Graph. **20**, 1–9 (2008) (Proceedings of the SIGGRAPH Conference)
20. McCool, M.D.: Anisotropic diffusion for Monte Carlo noise reduction. ACM Trans. Graph. **18**(2), 171–194 (1999)
21. Li, H., Wei, L.-Y., Sander, P.V., Fu, C.-W.: Anisotropic blue noise sampling. In: ACM SIGGRAPH Asia (2010)
22. Wei, L.-Y., Wang, R.: Differential domain analysis for nonuniform sampling. ACM Trans. Graph. (2010) (Proceedings of the SIGGRAPH Conference). doi:10.1145/2010324.1964945
23. DeCoro, C., Weyrich, T., Rusinkiewicz, S.: Density-based outlier rejection in Monte Carlo rendering. In: Pacific Graphics, vol. 29, p. 7 (2010)
24. LuxRender, http://src.luxrender.net (2008)
25. Wu, J.Z., Zheng, C.W., Hu, X.H.: Realistic rendering of bokeh effect based on optical aberrations. Vis. Comput. **26**, 555–563 (2010)

Xiao-Dan Liu is a Ph.D. candidate in the National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences. He received his B.Sc. degree from Xiamen University in 2009. His research interest includes computer graphics, realistic rendering, game developing and computer simulation.



Jia-Ze Wu is a Ph.D. in the National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences. He received his B.Sc. degree from Nankai University in 2005. His research interest includes real-time rendering, realistic rendering, virtual reality and optical simulation.

Chang-Wen Zheng is a Professor in the National Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences. He received his Ph.D. degree from Huazhong University of Science and Technology. His research interest includes computer graphics, virtual reality, computer simulation and artificial intelligence.