Multidimensional Adaptive Sampling and Reconstruction for Realistic Image Based on BP Neural Network

Yu Liu(✉), Changwen Zheng, and Fukun Wu

Science and Technology on Integrated Information System Laboratory,
Institute of Software, Chinese Academy of Sciences, Beijing, China
15600602128@163.com, changwen@iscas.ac.cn,
1024057620@qq.com

Abstract. A novel adaptive sampling and reconstruction algorithm is presented to address the noise artifacts of Monte Carlo rendering. BP neural network is adopted to estimate per pixel error that guides sampling rate in the multidimensional spaces. In sampling stage, coarse samples are firstly generated to train BP network. Then per pixel error is estimated by the BP predicted value. Additional samples are distributed to the slices of pixels which have large errors. These slices are extracted through a heuristic distance. A warping procedure is then carried out to remove individual light paths that result in significant spikes of noise. In reconstruction stage, the final image is reconstructed by filtering each pixel with appropriate anisotropic filter. Filters with small scales are used to keep clarity while the large ones smooth out noise. Compared to the state-of-the-art methods, experimental results demonstrated that our algorithm achieves better results in numerical error and visual image quality.

Keywords: Adaptive sampling and reconstruction · BP neural network · Anisotropic filter · Multidimensional analysis

1 Introduction

Monte Carlo technique computes pixel colors by analyzing the multidimensional integral at every pixel. This method distributes samples into the multidimensional domain and then integrates them into the final pixel value. Though its simplicity, Monte Carlo technique suffers serious noise because of limited sampling rate. In order to reduce noise with sparse samples, kinds of adaptive sampling and reconstruction methods are employed.

Adaptive sampling methods distribute samples with an optimal fashion that sampling rate is determined by per pixel error. Therefore regions with large errors acquire more samples. The pioneering work of Mitchell [1] laid the foundation for this method. There are two strategies to address this: image space sampling and multidimensional spaces sampling. The first strategy relies only on a local measure of variance, which fails to simulate effects such as motion blur and soft shadow. For instance, Rigau [2] and Xu [3] construct criterions by f-divergence and fuzzy uncertainty respectively. Bala et al. [4] perform an edge-aware interpolation to reduce noise. However, their quality is

Y.-J. Zhang (Ed.): ICIG 2015, Part II, LNCS 9218, pp. 510–523, 2015.
DOI: 10.1007/978-3-319-21963-9_47

restricted by the accuracy of edge detection. The second strategy takes lens, time and other non-image spaces into account to render a wide range of realistic effects. Hachisuka et al. [5] sample the multidimensional spaces to simulate special effects, but it is limited by the curse of dimensionality. Recently, algorithms which operate in the transform domain generate pleasing images, and Durand et al. [6] firstly analyze the frequency content of radiance and how it is altered by numerous phenomena. Through performing Fourier transform, Soler [7] and Egan [8] render high-quality depth of field and motion blur respectively. Overbeck et al. [9] harbor the idea that signal yields large wavelet coefficients while noise yields small coefficients, and they propose an iterative denoising framework by wavelet shrinkage. However, it is still challenging for these existing transform methods to analyze non-image spaces together.

Reconstruction algorithms are employed to construct a smooth result by applying appropriate filters. Bilateral filter is applauded for its feature preservation and Tomasi and Manduchi [10] combine spatial kernel and range kernel to construct bilateral filter. However, their efficiency is limited by the color term. Isotropic filters [11] which use symmetric kernels for all pixels cause a blurred result. Recently, anisotropic reconstruction methods have been widely developed. Greedy method proposed by Rousselle et al. [12] presents a pleasing result through selecting Gaussian filters for different pixels, but they didn't take much visual quality into consideration. Non-local mean filter [13] which splits samples into two buffers is also adopted to estimate per pixel error. Li et al. [14] employ anisotropic bilateral filters to better handle complex geometric, but it needs expensive auxiliary information such as depth and normal. The core idea for anisotropic reconstruction is the choice of filter scale. However, a robust criterion is challenging.

Recently, regression analysis functions [15, 16] have been widely used in realistic rendering for their compactness and ease of evaluation. The major shortage for the existing methods is to deal with high-dynamic multidimensional content. Most previous algorithms only analyze local variance of samples. RRF [15], for instance, returns the indirect illumination through giving viewing direction and lighting condition, and for this reason it fails to predict pixel error accurately.

In this paper, we propose a novel two-stage adaptive sampling and reconstruction method based the BP neural network (Fig. 1). The key idea is to design BP network as a nonlinear function of multidimensional content such that it is a global representation and fast to converge. Firstly, multidimensional content of coarse samples is used to train per pixel BP network and then per pixel error is estimated by the BP predicted value. Additional samples are then distributed to the pixels with large errors. A warping algorithm is also carried out to recognize outliers and removes them to further avoid spikes of noise. In reconstruction stage, per pixel error obtained in the former stage is chosen as a robust criterion to select filter scale. All the pixels are reconstructed by suitable anisotropy bilateral filter. Finally, a second reconstruction is recommended to make up the discontinuity caused by abrupt change of filter scales. Important notations and their experimental values are summarized in Table 1.

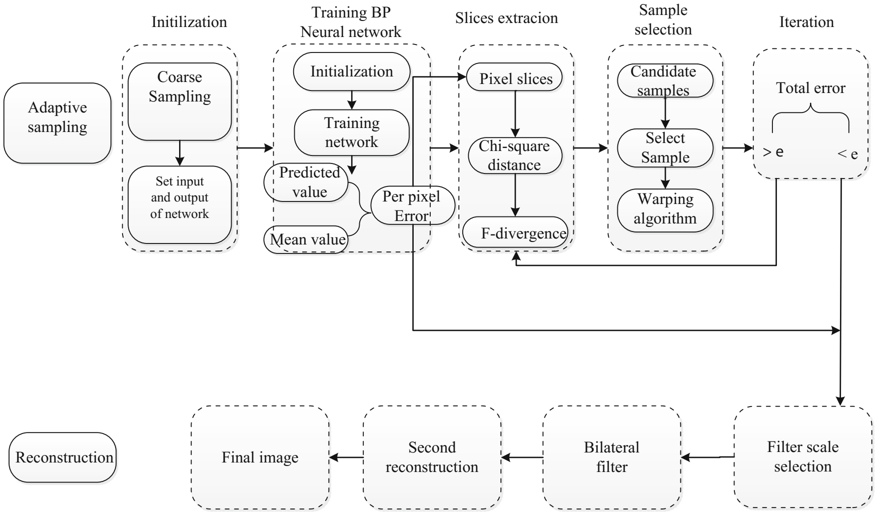


Fig. 1. Algorithm framework

Table 1. Notations and their experimental values

Notations	Content	Value
$\omega_{ih}\,\omega_{ho}$	Connected weights	(−1,1)
Nit	Training times	(10,100)
e	Error thresholding	0.1
$b_h\,b_o$	Output thresholding	0.01 0.01
μ	Learning rate	(0.02,0.08)
Ψ	Pixel contrast thresholding	0.05
N_{candi}	Number of candidate sample	4
r	Reconstruction radius	4
$\alpha\,\sigma_c$	Reconstruction shape parameters	$\alpha = \{0.1, 1, 3, 5, 10\}\ \sigma_c = 0.3$
$\Phi = \{\Phi_1\Phi_2\Phi_3\Phi_4\}$	Filter scale thresholding	$\Phi = \{0.3, 0.45, 0.7, 0.85\}$
$f(\mathrm{x})$	Reward function	$f(x) = \frac{1}{1+e^{-x}}$
m	Quantized levels	5

2 Adaptive Sampling Based on BP Neural Network

BP neural network is a back propagation process that predicts output values. It is a universal function approximator and we use it for its relative compactness and high speed in evaluation. Illustrated as Fig. 2, three layers are presented in the network. Each layer has a number of nodes connected by defined weights. The multidimensional content is regarded as input layer: two image positions, two lens positions and one time position. Additional dimensions such as light source can be easily transplanted into our

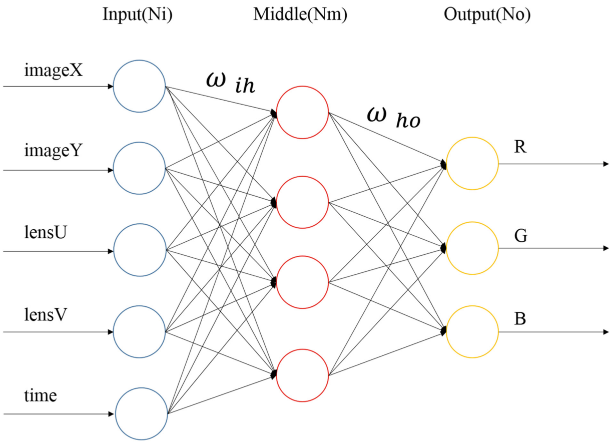


Fig. 2. Per pixel BP neural network (Color figure online)

framework for more effects, and we only describe five dimensions for simplicity. The output layer is constructed by the three components of pixel spectral value: red, green and blue.

2.1 Initialization

At the beginning of sampling stage, coarse samples are distributed to train network. The node numbers in three layers are $Ni = 5$, Nm and $No = 3$ respectively, where Nm is a user-defined parameter. Our algorithm distributes coarse samples by standard Monte Carlo method such as Low Discrepancy [17]. Only one hidden layer is set to balance training cost.

The training number Nit is a critical factor for accuracy. Traditional training methods take only one sample each iteration, which limit the prediction accuracy because only sparse samples are available. Our algorithm guarantees accuracy through employing the patch training model. In this model, all sample errors are added up to form a global error. Partial derivatives of this global error are then computed to adjust node weights $(\Delta\omega_{ih}, \Delta\omega_{ho})$. In theory, the effect of training Nit times with patch model equals to the effect of training $Nit \times Ncoarse$ times with traditional model, where $Ncoarse$ is current sample number. Besides, patch model results in a rapid convergence nature that reduce time.

2.2 Training BP Neural Network

Training BP neural network needs expected outputs which are used to compute training error. Since the actual pixel value are not available, mean value of coarse samples is utilized instead. One of our BP training iteration is described as follows:

Step1: Initialization BP network:

$$\text{input: } x = (x_1, x_2, x_3, x_4, x_5) \tag{1}$$

$$\text{expected output: } do = (do_1, do_2, do_3) \tag{2}$$

$$\text{input of Hidden layer: } hi = (hi_1, hi_2, hi_3 \ldots hi_{Nm}) \tag{3}$$

$$\text{output of Hidden layer: } ho = (ho_1, ho_2, ho_3 \ldots ho_{Nm}) \tag{4}$$

$$\text{input of Output layer: } yi = (yi_1, yi_2, yi_3) \tag{5}$$

$$\text{output of Output layer: } yo = (yo_1, yo_2, yo_3) \tag{6}$$

The connected weights are initialized as random numbers in (−1,1). The max error thresholding is denoted as e and output thresholding of hidden layer (output layer) is b_h (b_o). Learning rate is μ.

Step 2: Value propagation:

$$\text{input of Hidden layer: } hi_h = \sum_{i=1}^{5} \omega_{ih} x_i - b_h \quad (h = 1, 2, 3, \ldots Nm) \tag{7}$$

$$\text{output of Hidden layer: } ho_h = f(hi_h) \tag{8}$$

$$\text{input of Output layer: } yi_o = \sum_{i=1}^{Nm} \omega_{ho} ho_i - b_o \quad (o = 1, 2, 3) \tag{9}$$

$$\text{output of Output layer: } yo_o = f(yi_o) \tag{10}$$

Step 3: Adjust the weights:

$$\omega_{ih} = \omega_{ih} + \Delta\omega_{ih} \quad \omega_{ho} = \omega_{ho} + \Delta\omega_{ho} \tag{11}$$

Each training process is built on the basis of last iteration. When all the *Nit* training processes come to an end, our method set *imageX*, *imageY* to 0 and *lensU*, *lensV*, *time* to random numbers in (0,1) to calculate the predicted value I'. The current pixel error is then estimated as the contrast between I' and expected value I.

$$I = \frac{\sum_{x=1}^{n} I_x}{n} \tag{12}$$

$$bias = \frac{|I' - I|}{I} \tag{13}$$

n is the total number of samples. If $bias > \Psi$, the current pixel is determined to be in a high frequency region (such as edge or texture) that further sampling is needed.

2.3 Slices Extraction

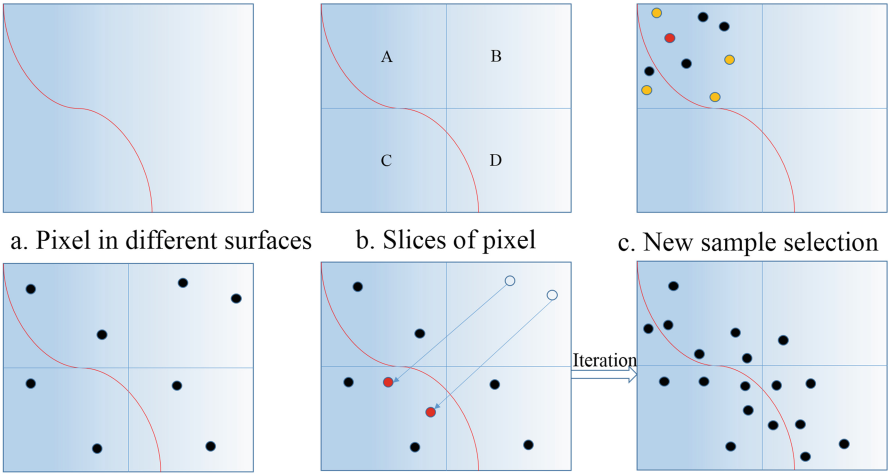
The high frequency pixels obtained in Sect. 2.2 usually locate in complex regions (Fig. 3(a)) which mainly cause noise. It is critical to choose new sample positions for these pixels. Our algorithm distributes new samples as follows. Firstly, these pixels are divided into slices (Fig. 3(b)). We focus on the truth that different slices present different complexities due to the nature of high frequency. For example, slice B has a lower complexity than slice C that locates in the boundary of two different surfaces. We extract the two slices which has the strongest contrast. This is achieved by considering Chi-square distance [18]:

$$diff(X,Y)=\frac{1}{N(X)+N(Y)}\sum_{i=1}^{m}\frac{\left(\sqrt{\frac{N(Y)}{N(X)}}hi(X)-\sqrt{\frac{N(X)}{N(Y)}}hi(Y)\right)^2}{hi(X)+hi(Y)} \tag{14}$$

$diff(X,Y)$ is the contrast between slice X and Y. Pixel value is quantized to m levels and $hi(X)$ is the current sample number of level i for slice $X.N(X)=\sum_i hi(X)$ $(N(Y)=\sum_i hi(Y))$ is the total sample number of slice $X(Y)$.

The two slices extracted above has the strongest contrast. Important notation is that one of them (slice B) is usually far away from the boundary while another one contain it (slice C). To locate more samples in the vicinity of this boundary, a warping algorithm is described in Sect. 2.4. Before our warping algorithm, the less complex slice is recognized in the light of f-divergence [2]:

$$fdiver(X)=\frac{1}{N(X)}\overline{L}\sqrt{\frac{1}{2}\sum_{i=1}^{N(X)}\left(\sqrt{p_i}-\sqrt{\frac{1}{N(X)}}\right)^2} \tag{15}$$



a. Pixel in different surfaces b. Slices of pixel c. New sample selection

d. New samples distribution e. Warping algorithm f. Final sample distribution

Fig. 3. New sample selection

where $\overline{L} = \frac{1}{N(X)} \sum_{i=1}^{N(X)} L_i$ is the mean value of slice X and $p_i = \frac{L_i}{\sum_{i=1}^{N(X)} L_i}$. Slice with a smaller $fdiver(X)$ is determined to be the less complex slice.

2.4 Sample Selection

To approximate Poisson disk distribution for new samples, N_{candi} (Fig. 3(c)) candidates are generated and the one which minimizes per slice error is finally selected to distribute into the image space:

$$S(x) = \frac{1}{N} \sum_{i=1}^{N(X)} \frac{|I_i - I_{can}|}{I_i} \tag{16}$$

$S(x)$ is the slice error for candidate I_{can}. The main difference between ours and naive Poisson disk distribution is the metric choice: per slice error used in this paper performs better in visual image quality while the traditional methods use image space distance. These newly selected samples commonly appear in the vicinity of complex geometries that relate to low-probability yet high-energy light paths. They are recognized as outliers and removed to further reduce noise. Besides, only local variance is minimized if same amount of samples are distributed into all slices, since that it can't focus effort in the most difficult part. New samples which possess smallest slice error are distributed into all the slices of difficult pixels obtained in Sect. 2.2 (Fig. 3(d)).

After suitable sample is chosen (red dot in Fig. 3(c)) from candidates (yellow dots in Fig. 3(c)), our algorithm transfers (Fig. 3(e)) new samples in the slice with a smaller f-divergence (slice B) into the slice with a larger f-divergence (slice C). These samples are transferred symmetrically by the local positions. Finally, the warping algorithm focuses effort in local difficult regions and thus the global variance is also minimized.

2.5 Iteration

Our algorithm distributes additional samples into pixels obtained in Sect. 2.2 with the warping algorithm in Sects. 2.3 and 2.4. The current global error is calculated as:

$$er = \frac{1}{2} \sum_{k=1}^{Ncoarse} \sum_{o=1}^{3} (do_o(k) - yo_o(k))^2 \tag{17}$$

Adaptive sampling stage executes an iterative process until a termination is met, for example, samples are used up or the global error er reaches its thresholding e. After several iterations, samples are mainly distributed in high-frequency regions (Fig. 3(f)) and thus the remained noise is limited to an acceptable level.

3 Reconstruction

Our reconstruction method make a trade-off between noise reduction and keeping clarity through filter scale selection. Pixels with large errors are suitable for small scale filters to keep clarity, and vice versa. We focuses on the fact that filters with small scale have little bias but much noise while the large ones have little noise but much bias. Therefore, large scale filters are selected for pixels in low frequency regions to denoise and small ones in the remain regions to reduce bias.

3.1 Bilateral Filter

Image distance $d()$ and color distance $c()$ are combined to construct the bilateral kernel $\omega_{pq} = d(|p-q|)c(|I_p - I_q|)$:

$$I_p = \frac{\sum\limits_{q \in \Omega_p} \omega_{pq} I_q}{\sum\limits_{q \in \Omega_p} \omega_{pq}} = \frac{\sum\limits_{q \in \Omega_p} e^{-\alpha \frac{|p-q|^2}{2}} e^{-\frac{|I_p - I_q|^2}{2\sigma_c^2}} I_q}{\sum\limits_{q \in \Omega_p} e^{-\alpha \frac{|p-q|^2}{2}} e^{-\frac{|I_p - I_q|^2}{2\sigma_c^2}}} \tag{18}$$

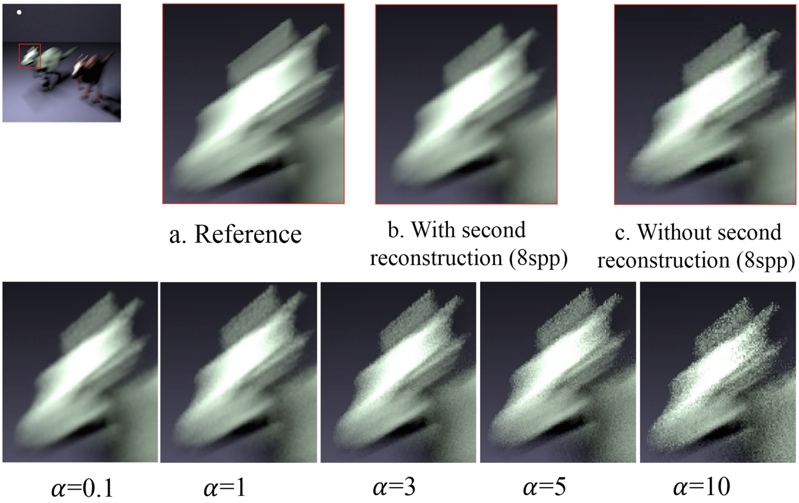
The final pixel value I_p is calculated by averaging pixels over a neighborhood window Ω_p with radius r. α and σ_c control the rate of falloff for $d()$ and $c()$ respectively. In practice, the same σ_c is used for all filters since the related work [14] pointed out that color term doesn't help much. A key observation is that the filter scale increases when the α goes from coarser to finer, and thus filter with a coarser α is selected to reconstruct pixels with larger error.

3.2 Filter Scale Selection

Our filterbank is composed of five spatially-varying bilateral filters, which denoted as $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$ that goes from finer to coarser. The choice thresholding is accumulated as $\Phi = \{\Phi_1 \Phi_2 \Phi_3 \Phi_4\}$. We utilizes the BP pixel error as a robust criterion, and therefor filter scale is simply chosen by considering the *bias*:

$$\begin{array}{lrl}
\text{if} & 0 < bias < \Phi_1 & \alpha = \alpha_1 \\
\text{else if} & \Phi_1 < bias < \Phi_2 & \alpha = \alpha_2 \\
\text{else if} & \Phi_2 < bias < \Phi_3 & \alpha = \alpha_3 \\
\text{else if} & \Phi_3 < bias < \Phi_4 & \alpha = \alpha_4 \\
\text{else} & & \alpha = \alpha_5
\end{array}$$

With suitable filter choice, output pixels are reconstructed. As shown in Fig. 4, filters with coarse α are selected in edges to keep clarity while finer ones are selected in background to denoise. However, this reconstruction may presents discontinuity (or Ringing artifact) in boundaries since the abrupt change of filter scale. As illustrated in


a. Reference

b. With second reconstruction (8spp)

c. Without second reconstruction (8spp)

α=0.1

α=1

α=3

α=5

α=10

Fig. 4. Filter scale selection

Fig. 4(c), image without second reconstruction exhibits more discontinuity and visual artifacts in edges. Through its low probability, it may cut down smoothness. We treat this problem by filtering all pixels again with the same scale α_3. This second reconstruction balances the discontinuity to a pile of nearby pixels. As shown in Fig. 4(b), the second reconstruction leads to a more smoothness result.

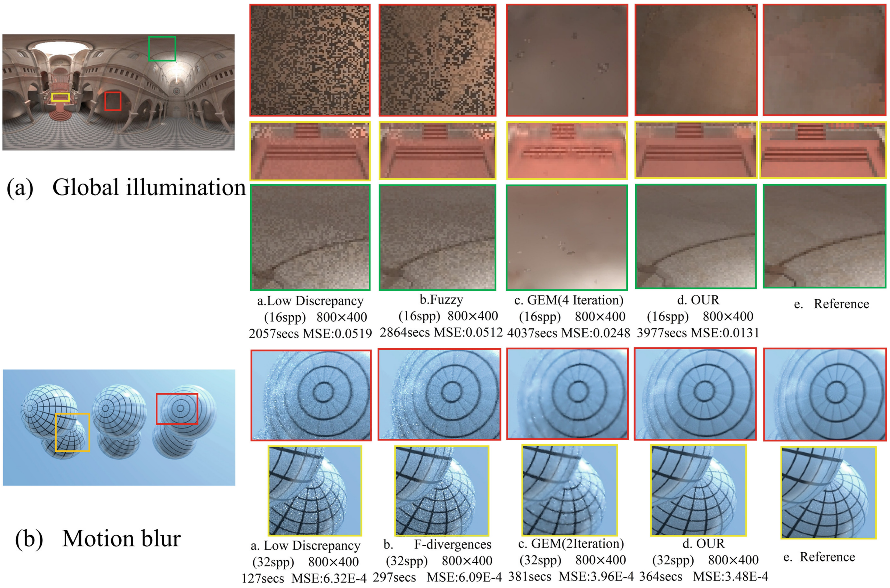
4 Experimental Results

Our algorithm and previous approaches are implemented on the top of PBRT [17]. All the images were rendered by an Intel(R) Core(TM) i7-3630 QM CPU with 8 GB RAM. Rendered images are compared to previous methods including Low Discrepancy [17], F-divergences [2], Fuzzy [3] and GEM [12]. The rendering time and MSE (mean square error) is also compared under the same experimental environment. Learning rate is a variable controlled by the global error *er*. In addition, we also explain how the effect of noise removal is altered by the critical parameter *Nit*. In our experiments, node number of input layer is set to 4 for simplicity:

$$\{image = \sqrt{(imageX)^2 + (imageY)^2}, lensU, lensV, time\} \tag{19}$$

4.1 Scenes

Figure 5(a) is a global illumination scene which is rendered with 16 samples per pixel. The images rendered by Low Discrepancy and Fuzzy suffer serious noise both in low frequency regions (red block and green block) and high frequency regions (yellow



(a) Global illumination

a.Low Discrepancy (16spp) 800×400 2057secs MSE:0.0519

b.Fuzzy (16spp) 800×400 2864secs MSE:0.0512

c. GEM(4 Iteration) (16spp) 800×400 4037secs MSE:0.0248

d. OUR (16spp) 800×400 3977secs MSE:0.0131

e. Reference

(b) Motion blur

a. Low Discrepancy (32spp) 800×400 127secs MSE:6.32E-4

b. F-divergences (32spp) 800×400 297secs MSE:6.09E-4

c. GEM(2Iteration) (32spp) 800×400 381secs MSE:3.96E-4

d. OUR (32spp) 800×400 364secs MSE:3.48E-4

e. Reference

Fig. 5. Global illumination and motion blur effects (Color figure online)

block). In this scene, the noise level of GEM is as good as ours. However, our method presents better details than GEM in high frequency regions such as stair. Besides, our result presents a smoother and less block effect result.

Figure 5(b) demonstrates a motion blur scene consists of three moving balls. This scene is rendered with 32 samples per pixel. Three balls rotates with different speed increasing from left to right. The red block shows that our method is able to sample the motive details better than Low Discrepancy and F-divergences, which generate much more noise. Pay attention to the static ball (yellow block), our method remove the outliers that mainly cause spikes of noise and produce a clearer detail while the GEM obviously causes an overly blurred image. This scene indicates that our warping algorithm is more sensitive to the high frequency details.

In the car scene render by 16 samples per pixel in Fig. 6(a), a soft shadow effect is simulated under the occlusion of tire (red block). GEM performs better than Low Discrepancy, but it failed to capture the texture on tire. Our algorithm suffers the least noise and is smoother on the soft shadow region. Besides, our image is closest to the reference image. Because of the BP neural network used in this article, we captures the details accurately. In addition, our denoising effect is as good as Low Discrepancy (Fig. 6(a).c) which consumes four times samples (64spp).

Figure 6(b) is a dragon scene with depth of field effect. We compared images both in the focus region (red block) and blur region (yellow block). The Low Discrepancy method suffers serious noise in the blur region because of its single sampling rate. Different with the GEM which uses Gaussian filters to reconstruct pixels, our bilateral filter take more visual image quality into account and produce a smoother result in the

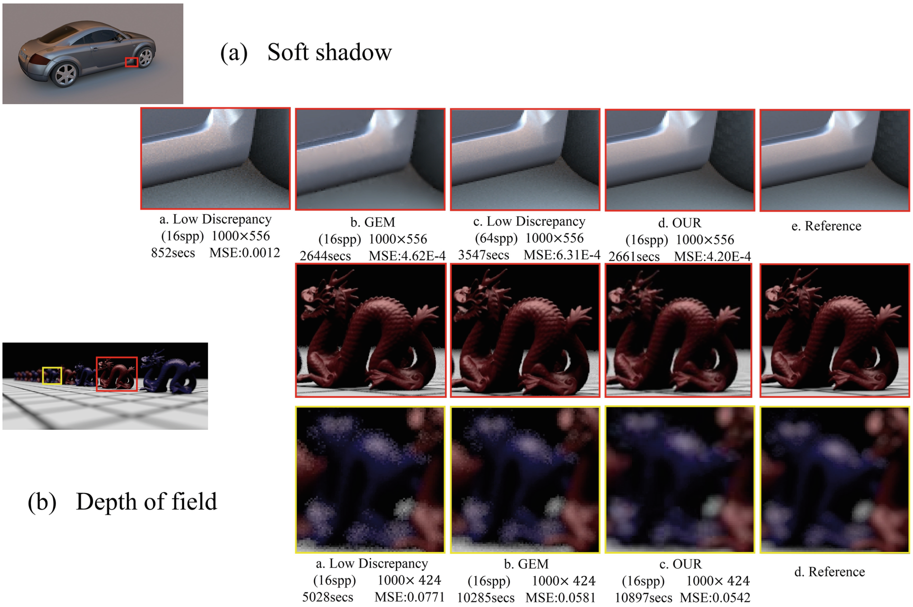


Fig. 6. Soft shadow and depth of field effects (Color figure online)

blur region. Besides, the second reconstruction procedure makes up the discontinuity which is more obvious in the boundaries of GEM.

4.2 Analysis

Figure 7 compares the MSE of Low Discrepancy and our algorithm for the scene in Fig. 5(b). This figure indicates that our method achieves a much lower MSE with the same sample number. The MSE and rendering time in Figs. 5 and 6 also indicate that our algorithm generates images with a lower MSE and better visual quality than previous methods. In summary, our algorithm is more sensitive to high frequency content both in the local and non-local variances and therefore we can handle a vast body of realistic effects.

The parameter *Nit* is critical to efficiency of our algorithm. In theory, the more times we training the more accurate results we obtain. However, time consumption limits the training number. Figure 8 analyze the change of MSE and rendering time when *Nit* goes from 10 to 100 for scene in Fig. 4. This scene is rendered with 8 samples per pixel. The blue line indicates that training time increases violently between (10–30) and (90–100) while gently between (30–90). The green line shows that MSE is relatively high when a small training number (10–40) is employed. Based on the above analysis, a large *Nit* (70–90) is recommended for simple scenes while a small *Nit* (50–60) for complex scenes to make a trade-off between time and MSE.

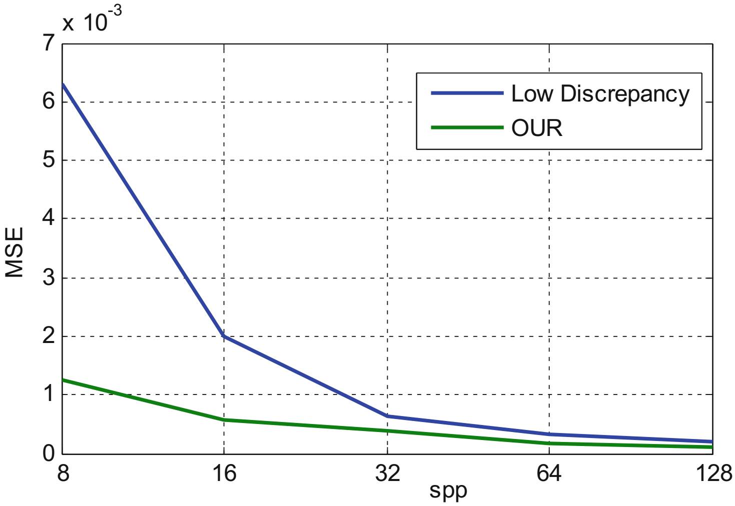


Fig. 7. Mean square error

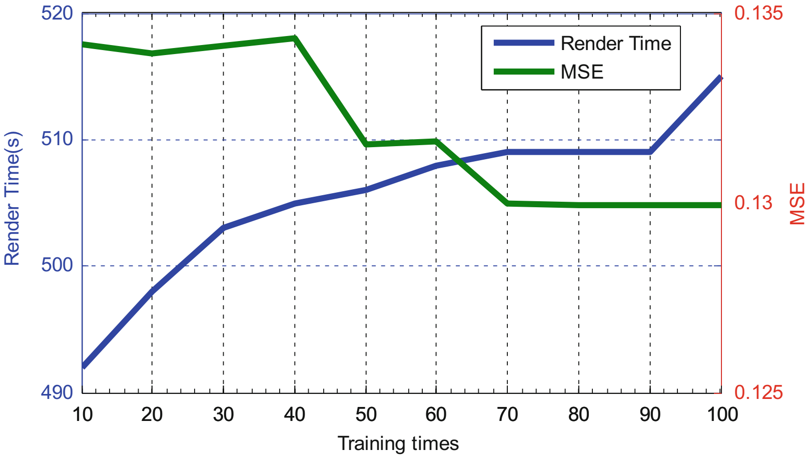


Fig. 8. Parameter analysis

5 Conclusion and Future Work

This paper proposes an efficient two-step rendering algorithm. We vary sampling rate according to the per pixel error calculated through BP analysis. Adaptive samples are distributed to the high frequency regions both in local and non-local regions. Besides, high light noise is also reduced through a warping algorithm. With our BP network analysis, suitable filter scales are selected to keep clarity while limit the noise artifacts. Experiments results indicates that our algorithm handles a wide range of effects and achieves a better result over many state-of-the-art algorithm in terms of MSE and visual image quality. In addition, the patch model we used to train per pixel network has little impact on the time consumption, where a reasonable increment of time is needed to estimate pixel error.

To handle scenes with great geometric complexity, we plan to develop Regression function through analyzing the auxiliary data including depths, surface normal and textured colors. We plan to integrate these features into our BP neural network and obtain a more comprehensive description of rendering details. Besides, our sampling procedure will be further applied to the wave rendering field to improve efficiency [19, 20].

Acknowledgments. The authors acknowledge the support of National High-tech R&D Program (863 Program) (No. 2012AA011206 and No. 2009AA01Z303).

References

1. Mitchell, D.P.: Generating antialiased images at low sampling densities. In: ACM SIGGRAPH 1987, pp. 65–72. ACM Press, New York (1987)
2. Rigau, J., Feixas, M., Sbert, M.: Refinement criteria based on f-divergences. In: Proceedings of the 14th Eurographics Workshop on Rendering, pp. 260–269. Eurographics Association Press, Aire-la-Ville (2003)
3. Xu, Q., Chen, D., Chen, H., et al.: Adaptive sampling based on fuzzy uncertainty. J. Comput. Aided Des. Comput. Graph. **20**(6), 689–699 (2008)
4. Bala, K., Walter, B., Greenberg, D.P.: Combining edges and points for interactive high-quality rendering. ACM Trans. Graph. **22**(3), 631–640 (2003)
5. Hachisuka, T., Jarosz, W., Weistroffer, R.P., et al.: Multidimensional adaptive sampling and reconstruction for ray tracing. ACM Trans. Graph. **27**(3), 33 (2008)
6. Durand, F., Holzschuch, N., Soler, C., et al.: A frequency analysis of light transport. ACM Trans. Graph. **24**(3), 1115–1126 (2005)
7. Soler, C., Subr, K., Durand, F., et al.: Fourier depth of field. ACM Trans. Graph. **28**(2), 18 (2009)
8. Egan, K., Tseng, Y.T., Holzschuch, N., et al.: Frequency analysis and sheared reconstruction for rendering motion blur. ACM Trans. Graph. **28**(2), 93 (2009)
9. Overbeck, R.S., Donner, C., Ramamoorthi, R.: Adaptive wavelet rendering. ACM Trans. Graph. **28**(5), 1–12 (2009)
10. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the Sixth International Conference on Computer Vision, pp. 839–846. IEEE Computer Society Press, Washington, D.C. (1998)
11. Cook, R.L.: Stochastic sampling in computer graphics. ACM Trans. Graph. **5**(1), 51–72 (1986)
12. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive sampling and reconstruction using greedy error minimization. ACM Trans. Graph. **30**(6), 159 (2011)
13. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive rendering with non-local means filtering. ACM Trans. Graph. **31**(6), 195 (2012)
14. Li, T.M., Wu, Y.T., Chuang, Y.Y.: SURE-based optimization for adaptive sampling and reconstruction. ACM Trans. Graph. **31**(6), 194 (2012)
15. Ren, P., Wang, J., Gong, M.: Global illumination with radiance regression functions. ACM Trans. Graph. **32**(4), 130 (2013)

16. Liu, X.D., Zheng, C.W.: Adaptive cluster rendering via regression analysis. Vis. Comput. **31**(1), 105–114 (2015)
17. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann, San Francisco (2010)
18. Delbracio, M., Musé, P., Buades, A., et al.: Boosting monte carlo rendering by ray histogram fusion. ACM Trans. Graph. **33**(1), 8 (2014)
19. Wu, F., Zheng, C.: A comprehensive geometrical optics application for wave rendering. Graph. Models **75**(6), 318–327 (2013)
20. Wu, F., Zheng, C.: Microfacet-based interference simulation for multilayer films. Graph. Models **78C**, 26–35 (2015)