ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 8, pp. 835-840, 2016. 8
http://dx.doi.org/10.5626/JOK.2016.43.8.835

몬테카를로 렌더링을 위한 슈어기반 실시간 에이트러스 웨이블릿 필터

(SURE-based À-Trous Wavelet Filter for Interactive Monte Carlo Rendering)

김 수 민 [†] 문 보 창 [††] 윤 성 의 [†††]
(Soomin Kim) (Bochang Moon) (Sung-Eui Yoon)

요 약 몬테카를로 렌더링은 사진과 흡사한 이미지를 렌더링하는 데 널리 쓰이는 기술이다. 그러나 이 기술로 고품질의 이미지를 얻으려면 픽셀 당 샘플의 수를 증가시켜야 하며, 필연적으로 긴 렌더링 시간을 필요로 한다. 이 문제를 풀기 위하여, 이미지 필터링 기술을 적용할 수 있다. 이는 적은 샘플 수로, 노이즈가 존재하는 렌더링 결과를 빠른 시간 내에 구한 뒤, 필터링을 적용하여 추가적인 샘플 없이 정답 이미지에 근사하는 부드러운 이미지를 얻는 방법이다. 본 논문에서는 에이트러스 웨이블릿필터에 스테인의 공평 에러 추정법(SURE)을 적용하여, 실시간에 가까운 속도로 렌더링한 이미지의 노이즈를 제거하는 방법을 제안한다. 슈어(SURE)를 이용하여 에이트러스 웨이블릿 필터의 필터링으로 인한 에러를 추정할 수 있고, 이를 통하여 에러를 줄이는 방향으로 웨이블릿의 계수를 정할 수 있다. 본 연구진은 이 필터링 방법을 최신 실시간 광선추적법 시스템인 엠브리(embree)에 적용하여 성능을 확인하였다.

키워드: 몬테카를로 렌더링, 필터링, 에러 추정법, SURE, 에이트러스 웨이블릿

Abstract Monte Carlo ray tracing has been widely used for simulating a diverse set of photo-realistic effects. However, this technique typically produces noise when insufficient numbers of samples are used. As the number of samples allocated per pixel is increased, the rendered images converge. However, this approach of generating sufficient numbers of samples, requires prohibitive rendering time. To solve this problem, image filtering can be applied to rendered images, by filtering the noisy image rendered using low sample counts and acquiring smoothed images, instead of naively generating additional rays. In this paper, we proposed a Stein's Unbiased Risk Estimator (SURE) based À-Trous wavelet to filter the noise in rendered images in a near-interactive rate. Based on SURE, we can estimate filtering errors associated with À-Trous wavelet, and identify wavelet coefficients reducing filtering errors. Our approach showed improvement, up to 6:1, over the original À-Trous filter on various regions in the image, while maintaining a minor computational overhead. We have integrated our propsed filtering method with the recent interactive ray tracing system, Embree, and demonstrated its benefits.

Keywords: Monte Carlo rendering, filtering, Stein's un biased error estimation, SURE, À-Trous wavelet

· This work was supported in part by NRF-2013R1A1A2058052 and MSIP/NRF (No. 2013-067321)

† 학생회원 : 한국과학기술원 전산학부
soo.kim813@gmail.com
†† 비 회 원 : 한국과학기술원 전산학부 연구원
moonbochang@gmail.com
††† 종신회원 : 한국과학기술원 전산학부 교수(KAIST)
sungeui@gmail.com
(Corresponding author임)

논문접수 : 2016년 4월 18일
(Received 18 April 2016)
논문수정 : 2016년 5월 11일
(Revised 11 May 2016)
심사완료 : 2016년 5월 19일
(Accepted 19 May 2016)

1. Introduction and Related Work

The most common way of rendering scenes in a high-quality manner is solving the rendering equation [1]. Cook et al. [2] introduced Monte Carlo (MC) ray tracing techniques, which compute pixel values by averaging discrete samples through the MC integration. This technique, however, requires a huge amount of samples until we achieve nearly noise-free results. Therefore, unless a large number of samples are used, the rendered image can show random noise. We can reduce such noise by generating more rays, but this approach requires long rendering time.

In order to shorten the rendering time while preserving the image quality, many different techniques have been proposed. One common approach is to sample more on complex areas (e.g., high contrast regions) compared to plain ones. This approach is commonly known as adaptive sampling technique and mainly initiated by Mitchell [3,4]. This method can achieve better quality images through distributing more samples on regions with more errors. Recently, applying image filters on input noisy images to get better quality images has been a popular approach [5]. For interactive rendering, the number of samples allowed is typically very small (e.g., 4 samples per pixel). Therefore, filtering on images rendered with small sample counts is useful for interactive rendering.

Among various denoising filters, applying wavelet transform on images has been widely used for noise reduction [6,7]. Since image's noise statistical properties and wavelet transformed frequency distributions have a correlation, many researchers try to utilize the wavelet properties to eliminate the noise. One of the image wavelet properties that is useful for denoising is that when pixel values are decomposed into the wavelet domain, noises and edges tend to be captured in high-frequency wavelet coefficients, i.e., detail coefficients [8]. Therefore, to denoise images, one can adjust detail coefficients in a way that the noise contained in the image is reduced. Specifically, when we decompose the pixel value from the image into wavelet coefficients, we can modify the coefficients, and transform them back into the original color intensity space. We can then eliminate the image noise theoretically Dammertz et al. [9] use a variant of À-Trous wavelet filter for reducing MC noise for global illumination images. They modified the À-Trous wavelet filter so that G-buffers can be utilized. They achieved real-time performance by implementing the filtering method in GPU. Nonetheless, this method tends to lose detail information in images, mainly because it does not estimate filtering errors caused by adjusting the wavelet coefficients, and simply cancels such detail information.

Our approach estimates this filtering error using Stein's Unbiased Risk Estimator (SURE) [10,11], and using this error metric, we reduce the filtering error in terms of Mean Squared Error (MSE). In this paper, we introduce a SURE-based À-Trous wavelet filter for interactive rendering. We achieve interactive filtering rate, since our method is suitable for GPUs. We have implemented our filtering method including the additional SURE calculation part in a GPU. Using our algorithm, we can achieve less filtering error compared to the edge avoiding À-Trous wavelet transform, while still achieving an interactive rate of rendering performance.

In summary, our main contributions in this paper are:

- Improve the filtering quality of the interactive filtering method by incorporating an error metric.
- Achieve interactive filtering rates based on GPU implementation of our method.

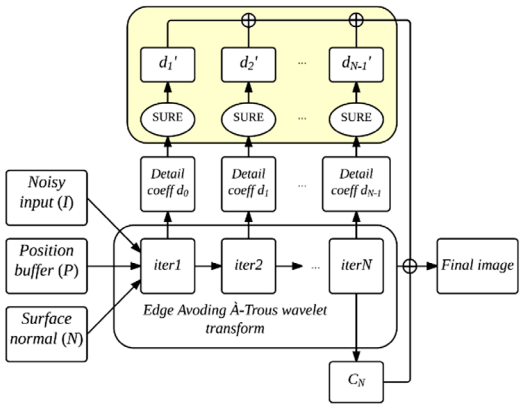

Fig. 1 The proposed denoising framework. SURE is used to find detail coefficient d_i, for reducing the filtering error of the wavelet filtering method. N is the wavelet decomposing level, i.e., the number of iterations of the wavelet filtering process

2. SURE-based À-Trous Wavelet Filter

Our method is based on the edge-avoiding À-Trous wavelet filter [9], but improves it by incorporating the SURE error estimator. Our filtering framework is divided mainly into two stages. We first apply the edge-avoiding À-Trous wavelet transform to get wavelet coefficients for each pixel in the image (Sec. 2.1). The second part is to modify the transformed wavelet coefficient, especially, detail coefficient d_i, using the SURE error estimator (Sec. 2.2). Fig. 1 shows steps of our algorithm. The input image and various geometric information stored in the G-buffer are fed, and we apply the edge-avoiding À-Trous wavelet filter N times; we use five times for our tests. We then modify wavelet coefficient based on the SURE calculation.

2.1 Edge-Avoiding À-Trous Wavelet Transform

Edge-avoiding À-Trous wavelet transform for fast global illumination filtering is one of the efficient interactive image filtering techniques for removing noise in images generated by MC rendering methods. Like many other wavelet filters including this approach, noise and high-frequency of image signals are mixed together and contained in high-frequency wavelet coefficients. To discern those two different quantities in the high-frequency wavelet coefficients, the previous method adopts edge-stopping functions based on geometric information stored in the G- buffer. Unless high-frequency wavelet coefficients are not supported by such edge-stopping functions, we simply cancel those wavelet coefficients, assuming that they come from only noise. However, the detail information can leak from those edge-stopping functions, so we cannot avoid filtering errors caused by simply canceling those wavelet coefficients. In our method, we reduce those detail coefficients according to an error metric. Its pseudo-code is shown in Algorithm 1.

In line 5, it performs a variant of the bilateral filter on c_i, whose initial value c_o is set to the input noisy value. k is the normalization factor, and $h_i(\cdot)$ is a 2D version of the B3 spline filter at i-th iteration. w_n, w_x, w_y are edge-stopping functions considering normals, positions, input values, respectively. $w_n(\cdot)$ and $w_x(\cdot)$ are adopted from the original edge-avoiding À-Trous wavelet. w_y considering input values is defined as the following:

Algorithm 1: Our SURE-based À-Trous Wavelet Filter

Input: Input I_{noisy}, Surface Normal N, Position X
Output: Result image I_{final}

1 **for** *Iteration step* $i = 0,1,2,3,\cdots N$ **do**
2 **for** *each pixel* p *in image* I_{noisy} **do**
3 **if** $i = 0$ **then**
4 $C_0(p) = I_{noisy}(p)$
5 $C_{i+1} = \frac{1}{k}\sum_{q\in\Omega} h_i(q)\cdot w_n(p,q)\cdot w_x(p,q)\cdot w_y(p,q)\cdot C_i(p)$
6 $d_i(p) = C_{i+1} - C_i$
7 $d_i'(p) \leftarrow min(SURE(f(d_i), d_i))$
8 $I_{final}(p) = C_N(p) + d_0'(p) + d_1'(p) + \cdots + d_N'(p)$
9 **return** I_{final}

$$w_y(p,q) = e^{-\frac{\|I_p - I_q\|}{\sigma_y^2}}$$

where I_p and I_q are input values of p and q pixels. We set σ_y to be the variance of the sample mean of input values, y_i, and it is defined as:

$$\sigma_y = \frac{1}{n-1}\left(E(y^2) - (E(y))^2\right) = \frac{1}{n-1}\left(\frac{\sum y_i^2}{n} - \left(\frac{\sum y_i}{n}\right)^2\right)$$

Additionally, as the wavelet iteration increases in the algorithm, filtered values become increasingly smooth. As a result, its variance goes smaller. To reflect this behavior, we adjust σ_y at i-th iteration as: $\frac{1}{2}\sum_{i=0}^{N} i \cdot \sigma_y^0$, where $\sigma_y^0 = \sigma_y$

2.2 Filtering Error Estimation with SURE

We can suppress noise by shrinking the wavelet coefficients. The wavelet shrinkage is introduced by Donoho et. al [7]. They utilized a global thresholding value to entire image for denoising. Compare to that, our method works adaptively on each pixel. Our shrinking weight is set for each pixel to minimize the calculated SURE value.

For i-th wavelet transform iteration, we have detail coefficients d_i. To adjust the coefficients, we multiply pre-defined weights to the wavelet coefficients, and calculate SURE for each case. We then choose the coefficient weight that gives us the minimal SURE value. In our test, we pre-define and test four different weights, 0.2, 0.4, 0.6, and 0.8.

The error associated with our filtering method, f (·), is estimated as the following Mean Squared Error (MSE) between y_i, input value at pixel i, and its ground truth pixel value, x_i :

$$E[(f(y_i) - x_i)^2]$$

This SURE value estimates the filtering error in an unbiased manner by using a few computable terms:

$$E[(f(y_i) - x_i)^2] = E[SURE(f(y_i))] \quad (1)$$

In our case, we adjust wavelet coefficients from the original coefficient d_i to d_i' by multiplying a weight w, which is pre-defined earlier, i.e. $d_i' = wd_i$. As a result, the aforementioned SURE equation needs to be transformed into the wavelet domain as the following [12]:

$$SURE(f(d_i), d_i) = N_c - 2|\{i: |d_i| \le f(d_i)\}| + \sum_{i=1}^{N} min(|d_i|, f(d_i))^2, \quad (2)$$

where $f(d_i)$ is reduced wavelet coefficients. In our application, we set N_c to be the number of pre-defined weight settings, and d_i is noisy detail coefficients (line 6 in Algorithm 1). After calculating the SURE value for each weighting case, we can find the lowest SURE value case. We take that weighting value w to reduce detail coefficients for each pixel.

3. Implementation Details and Results

We implemented our algorithm on the Embree renderer system [13]. We use Monte Carlo path tracing with small sample counts for generating noisy input images, and we utilize surface normal and depth as geometric information for our filtering. Intel(R) Core(TM) i7-3770k CPU @ 3.50GHz with NVIDIA GeForce GTX 580 GPU is used for performance measurement and comparison.

3.1 GPU implementation

Since our algorithm contains additional error estimation computation, we implemented our method on the GPU to minimize its computation overhead. We especially consider coherency to optimize our GPU CUDA implementation.

3.1.1 Optimizing array for coherency

Our CUDA program parallelizes the computation by dividing the whole work to small tasks, and those tasks are handled by many threads simultaneously. Therefore, if we use a multi-dimensional array, their accessing process will be inefficient, and it will result in time loss. Therefore, we manage information all in a 1D array. Moreover, to consider spatial and temporal coherency, we filled the information in the order of computation. We try to put the values in nearby space if they have coherency in the computation. Also, we utilize shared memory for accessing pre-defined B3 spline filter values.

3.2 Runtime Performance

We implemented our method, especially wavelet transform, and SURE calculation part on GPU, so that it works at an interactive rate. The image resolution for our testing is 512×512 and tested scenes are rendered using 16 samples per pixel (spp). We integrated our filtering method with the Embree system.

Table 1 Time comparison of the proposed method and edge-avoiding À-Trous filter. RT is rendering time and Filter is filtering time. Filtering time for our method is less than 60 ms in two scenes

		RT (ms)	Filter (ms)	Total (ms)	MSE
cornell (16 spp)	À-Trous	595	46	641	0.002979
	Our method	595	59	654	0.000445
sphere (16 spp)	À-Trous	441	47	488	0.005829
	Our method	441	57	498	0.003794

3.3 Comparisons

We compared our method with the prior, edge-avoiding À-Trous filtering [9]. Fig. 2 shows results of our method and the prior method in a sphere mirror scene with a HDR texture light. Our method uses only 10 ms more filtering time, which is drastically small compared to the ray tracing time. On the other hand, our method achieves about 40% lower MSE

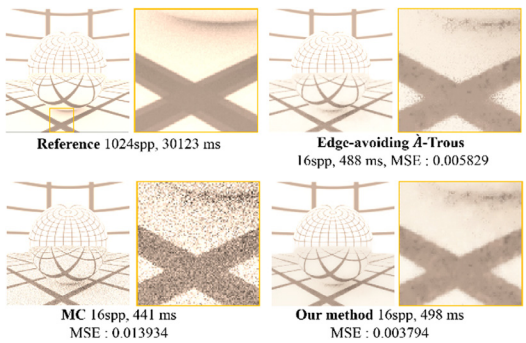

Fig. 2 Sphere mirror scene. The input image is generated by 16 spp, while the prior method fails to denoise regions around lines, our method smoothes out such regions

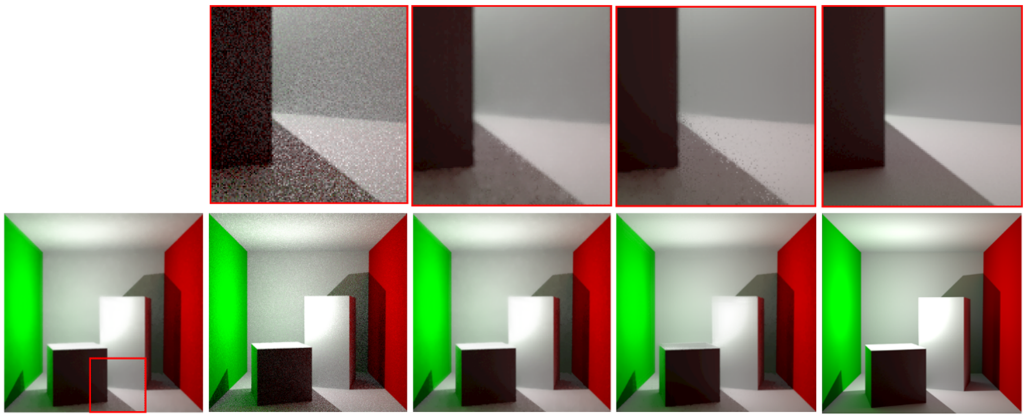

Fig. 3 Cornell box scene. Our proposed method achieves 6:1 MSE reduction over the prior method with a small additional filtering overhead

value and visually better image over the prior work. Especially, the prior method fails to denoise regions around lines, while our method smooths them well.

Fig. 3 shows filtering results of a different scene, the Cornell box scene, lit by a point light. Our method achieves a significantly reduced MSE value, 6:1, over the prior method, while it has 13 ms more filtering time.

4. Conclusion and limitation

We have introduced SURE-based À-Trous wavelet filter for MC rendering. Our method combines the wavelet shrinkage technique and SURE for denoising images with less filtering errors. By doing so, we were able to maintain the detail information that leaks from the edge-avoiding À-Trous wavelet filter. We have also implemented SURE computation on GPU and increased benefits of our method with Embree, interactive ray tracing kernel.

There are several limitations as well. Noise is based on the variances of ray samples generated from Monte Carlo ray tracing, and if the number of samples are large, according to the central limit theorem, the distribution can be considered as normal distribution [14]. However, if the ray samples are insufficient, it might be hard to consider noise as random Gaussian noise, and thus, the result quality cannot be good as expected.

There can be several future directions. In the current method, there are predefined, four cases for calculating SURE equations. Instead of this straightforward approach, we would like to perform analysis on wavelet coefficients and find the optimal wavelet shrinkage value without using predefined cases.

References

[1] Kajiya J. T., "The rendering equation," *Proc. of the ACM SIGGRAPH Computer Graphics*, Vol. 20, No. 4, pp. 143-150, 1986.

[2] Cook R. L., Porter T., Carpenter L., "Distributed ray tracing," *Proc. of the ACM SIGGRAPH Computer Graphics*, Vol. 18, No. 3, pp. 137-145, 1984.

[3] Mitchell D. P., "Generating antialiased images at low sampling densities," *Proc. of the ACM SIGGRAPH Computer Graphics*, Vol. 21, No. 4, pp. 65-72, 1987.

[4] Mitchell D. P., "Spectrally optimal sampling for distribution ray tracing," *Proc. of the ACM SIGGRAPH Computer Graphics*, Vol. 25, No. 4, pp. 157-164, 1991.

[5] Zwicker M., Jarosz W., Lehtinen J., Moon B., Ramamoorthi R., Rousselle F., Sen P., Soler C., Yoon S.-E., "Recent advances in adaptive sampling and reconstruction for monte carlo rendering," *Computer Graphics Forum*, Vol. 34, No. 2, pp. 667-681, 2015.

[6] Kaur L., Gupta S., Chauhan R., "Image denoising using wavelet thresholding," *Proc. of ICVGIP*, Vol. 2, pp. 16-18, 2002.

[7] Donoho D. L., Johnstone I. M., "Adapting to unknown

smoothness via wavelet shrinkage," *Journal of the American statistical association*, Vol. 90, No. 432, pp. 1200-1224, 1995.
[8] Gonzalez R. C., *Digital image processing*, 2nd Ed., Pearson Education India, 2009.
[9] Dammertz H., Sewtz D., Hanika J., Lensch H., "Edge-avoiding à-trous wavelet transform for fast global illumination filtering," *Proc. of the Conference on High Performance Graphics*, Eurographics Association, pp. 67-75, 2010.
[10] Stein C. M., "Estimation of the mean of a multivariate normal distribution," *The annals of Statistics*, pp. 1135-1151, 1981.
[11] Blu, T., Luisier F., "The sure-let approach to image denoising," *IEEE Transactions on Image Processing*, Vo. 16, No. 11, pp. 2778-2786, 2007.
[12] Rangarajan R., Venkataramanan R., Shah S., "Image denoising using wavelets," *Wavelet and Time Frequencies*, 2002.
[13] Wald I., Woop S., Benthin C., Johnson G. S., Ernst M., "Embree: a kernel framework for efficient cpu ray tracing," *ACM Transactions on Graphics (TOG)*, Vol. 33, No. 4, pp. 143, 2014.
[14] Li, Tzu-Mao; Wu, Yu-Ting; Chuang, Yung-Yu, "SURE-based optimization for adaptive sampling and reconstruction," *ACM Transactions on Graphics (TOG)*, Vol. 31, No. 6, pp. 194, 2012.

김 수 민
2013년 KAIST 물리학과/전산학과 졸업(학사). 2015년 KAIST 전산학부 졸업(석사). 2015년~현재 KAIST 전산학부 박사과정. 관심분야는 컴퓨터 그래픽스, 비젼

문 보 창
2008년 중앙대 컴퓨터공학부 졸업(학사) 2010년 KAIST 전산학부 졸업(석사). 2014년 KAIST 전산학부 졸업(박사). 관심분야는 컴퓨터 그래픽스, 실시간 렌더링

윤 성 의
1999년 서울대 전산과 졸업(학사). 2001년 서울대 컴퓨터공학부 졸업(석사). 2005년 Univ. of North Carolina at Chapel Hill 졸업(박사). 2007년~현재 한국과학기술원 교수. 관심분야는 컴퓨터 그래픽스, 비젼, 로보틱스