MATHEMATICA MONTISNIGRI
Vol XLIII (2018)

# TWO DENOISING ALGORITHMS FOR BI-DIRECTIONAL MONTE CARLO RAY TRACING

**S. V. ERSHOV[1], D. D. ZHDANOV[2], A. G. VOLOBOY[1], V. A. GALAKTIONOV[1]**

[1]Keldysh Institute of Applied Mathematics of RAS,
Miusskaya Sq. 4, Moscow, Russia, 125047
e-mail: voloboy@gin.keldysh.ru, web page: http://keldysh.ru/

[2]ITMO University, 49 Kronverksky Pr., St. Petersburg, 197101, Russia

**Summary.** Two methods are suggested to reduce noise in images obtained by a bi-directional stochastic ray tracing.

The first one uses filtering of the ratio of the ray-traced noisy image to the noise-free "pivot" image obtained by a deterministic ray tracing. After back multiplying the filtering result by the "pivot" image one gets a fine-detailed image with much reduced noise.

The second one uses filtering, but not of the resulting image but of the effective illumination, integrated into ray tracing.

In most cases both methods do not destroy fine details while much reduce noise. The first method is computationally cheaper, but for some scenes it may produce artifacts. The second method is more universal, but requires more memory.

## ABBREVIATIONS

**MCRT** = Monte Carlo ray tracing

**FMCRT** = forward Monte Carlo ray tracing. It is tracing of rays from light sources toward scene objects accumulating of illumination on scene objects. Usually it is used to calculate the secondary (indirect) illumination.

**BMCRT** = backward Monte Carlo ray tracing. It is tracing of rays from camera through virtual screen toward scene objects. Usually it is used to get the virtual scene image.

**BDF** = bi-directional scattering function. It describes surface luminance as a function of the illumination and observation direction

**BDD** = backward diffuse depth. It is a specific parameter of a hybrid ray tracing, when FMCRT calculates illumination and BMCRT is used to convert it to the observed luminance. In this method the backward ray usually has a limited "length" and terminates after BDD diffuse events.

**2010 Mathematics Subject Classification:** 78-04, 65C05, 65C20.
**Key words and Phrases:** Filtering, denoising, ray tracing, Monte Carlo ray tracing.

## 1 INTRODUCTION

Images calculated with the bi-directional Monte Carlo ray tracing usually contain substantial noise, which vanishes only after large run time. Depending on the peculiarities of the method used to calculate the image, the noise can be different: either a rather homogeneous random fluctuations, or isolated rare bright points or clusters consisting of dozens of bright points. In either case small-scale or low contrast image elements can be completely obscured by that noise.

A usual denoising method is filtration but even a bilateral filter can distort small-scale details. Therefore many efforts were spent on the search of an "optimal" filter that reduces noise without distortion of small scale image structure.

Filtering was used long ago [1], [2] but its results were not that good. The noise was reduced by averaging over a neighborhood of the target point, thus strong noise reduction required averaging over a large "window" that kills high-frequency components of the signal. A good reviews can be found in [3] and [4].

Therefore for the traditional linear filters (which convolve the signal with the averaging "kernel") reduction of noise and keeping small details are incompatible. Possible approaches to overcome that include expansion by wavelet, Laplace pyramid (level of detail) etc, see e.g. [5]. Besides there is a well-known bilateral filter which is nonlinear and its kernel vanishes where the input signal deviates too much from its value in the target point (filter center). As a result it keeps the boundaries between different object well. But a low contrast texture such as foliage can be completely destructed.

Recently this filter had been revisited and extended in [6], [7], [3]. The key idea is to add a criterion of point exclusion. In the original bilateral filter it was just the difference in brightness. But besides it one can use the difference of the gradient or another accompanying information. For images obtained by ray tracing it can be the normal field [6]. However using the noisy signal (say nothing about its gradient) as a criterion does not work when the noise level is comparable or exceeds the "regular" difference. A possible remedy is that the criterion compares not the original signal but some pre-filtered one or histograms [6].

At last it can be good to compare not the pointwise values but their spatial distribution in some neighbor areas. For example, when filtering a noisy image of a printed text we compare "windows" of the size of a text letter. If the window around a point differs too much from that around the central point, all points in that window are effectively excluded from averaging. Then ideally this filter will average all pixels of one letter "a" over the matching pixels of another "a" entries, even if they are spatially distant. Since averaging goes over windows with the same regular distribution, it does not destroy any detail. And if there were many entries of the "a" letter, they will be completely denoised. Such a filter is described in [7].

For periodic or quasi-periodic images like wallpaper it works excellently: in the noisy source image the wallpaper texture is completely indistinguishable while the filtered image shows it with all detail and fine contrast. But for a completely irregular texture like foliage, fur, etc. the results are much worse, although still much better than for the traditional

convolving filters.

Filters based upon such nonlinear and nonlocal averaging are popular now, and many possible ways of improvements were suggested [6]. But even they do not work well in all cases. Besides, they are computationally expensive because the averaging area is large and for each point the whole image area is processed even if eventually the criteria assigns zero weight for most of pixels.

One can apply filters not to the image itself, but to the ray paths [8] as well. The method from [9] is in a sense similar: a Monte Carlo ray trajectory depends on several random (and usually independent) variables used to choose the scattering direction, etc. As a result, the image part is also a function of such random parameters, and the idea is to reconstruct this dependence and then use fewer samples for the same noise level.

A promising direction is to approximate a surface BDF via some analytic models that admit fast ray tracing but such that the difference from the exact BDF is a slowly varying function that can be filtered [10].

Another class of methods is based upon the usage of the additional "image layer". This is possible not always, but for the images calculated by ray tracing usually one can obtain a plenty of that additional layers that help in effective denoising. For example, let us imagine a textured plane under illumination that changes slowly along it, such as a newspaper illuminated by skylight. Its ray traced image can be strongly noisy, but it is known that the exact image (to be obtained by denoising) is a product of a slowly varying illumination by a high-frequency modulator (texture). That texture is known *a priori* or can be obtained by a noise-free deterministic ray tracing. Then filtered must be only the illumination, and because it is slowly changing, even traditional convolving filters proceed it well and fast. Then the result is multiplied by the known texture and we obtain the denoised image with all details [11]. Other decompositions of the original image into several components are also possible, see e.g. [5].

Depending on the ray tracing method the scene and so on advantageous are different decompositions. This class of methods is not a pure filtration, because it is integrated into the Monte Carlo ray tracing. To some extent mutations of the ray trajectories also can be related to this area. Although mutations are mainly used in the Metropolis Light Transport [12], they can be also applied in the "standard" Monte Carlo ray tracing.

Filtering of the luminance of a turbid medium (sun-lit clouds, etc.) in [13] also utilizes an additional information, which here is a gradient of the density of medium.

This paper describes a two filter-like methods that advance the above ideas.

The former filter uses decomposition of the original noisy image into a product of a noise-free small-scale "pivot" image and a noisy large-scale function that thus can be efficiently filtered [11]. It can be thought of as an evolution of [14] but using ambient illumination instead of their "virtual flash light" coming from camera.

The second approach operates not the luminance image, but "effective illumination" assuming the indirect illumination is smooth. In the bi-directional Monte Carlo ray tracing indirect illumination of a camera ray hit point is calculated by emitting a BMCRT ray from that point and tracing it gathering FMCRT hit points in each its scattering event. This can be termed "effective illumination". This filter operates just that effective illumination and is advantageous when the reflected BMCRT rays rarely hit well illuminated areas. It is integrated into the bi-directional Monte Carlo ray tracing engine instead of processing

the final image [15]. It can be thought of as an evolution of [8] which averaged over ray trajectories. Our approach adds compensation for fast variation of the normal, BDf, texture etc.

Both methods work good for a large class of scenes, but in complex cases the first one sometimes produces artifacts. The second method processes these cases correctly, but it is more computationally expensive. Besides, as being integrated into the ray tracing process, it must be started from the very beginning.

## 2 BRIGHTNESS OF PIXEL IN BI-DIRECTIONAL RAY TRACING

For the further explanations we have to describe how the luminance in a pixel is calculated in our variant of bi-directional ray tracing for scenes like shown in Figure 5. It must be emphasized that camera ray does not undergo any specular events before the first diffuse hit. This luminance is

$$\begin{aligned} L(\boldsymbol{v}) &= \left(\int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})d^2\boldsymbol{u}\right) \\ &\times\frac{1}{N_B N_F}\sum_{i=1}^{N_B}\sum_{j=1}^{N_F} f_{B(\boldsymbol{v},\boldsymbol{u}_i)}(\boldsymbol{u}_i,\boldsymbol{V}_j;\boldsymbol{n}_{B(\boldsymbol{v},\boldsymbol{u}_i)})\frac{\chi_{i,j}E_j}{\pi R_i^2} \end{aligned} \tag{1}$$

where $i$ enumerates camera (BMCRT) rays, traced through this pixel, and $j$ enumerates FMCRT rays from lights. Then, $\boldsymbol{v}$ is the direction of the first segment of camera ray which we assume fixed for all rays traced through this pixel, $\boldsymbol{u}$ being the direction of its second segment (already random). Direction of the FMCRT ray segment is denoted $\boldsymbol{V}_j$ and the ray energy in the segment end as $E_j$.

Then, $A(\boldsymbol{v})$ and $B(\boldsymbol{v},\boldsymbol{u}_i)$ are the first and the second diffuse hits of that camera ray. The former is fixed while the second is already random because after diffuse reflection the ray direction becomes random. The normal to a surface in the point $X$ is denoted $\boldsymbol{n}_X$. BDF in luminance factor units in point $X$ is $f_X(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n})$, where $\boldsymbol{v}$ is direction of observation and $\boldsymbol{u}$ is direction of illumination.

It should be noted that if there are specular surfaces in the camera ray path, it becomes random after the vertex $A(\boldsymbol{v})$, and thus $B(\boldsymbol{v},\boldsymbol{u})$ is a stochastic function of $\boldsymbol{v}$ and $\boldsymbol{u}$.

The indicator function $\chi_{i,j}$ is 1 if the $j$-th FMCRT ray hits integration sphere around $B(\boldsymbol{v},\boldsymbol{u}_i)$, and 0 otherwise. If the camera ray does not have a second diffuse hit, $\chi_{i,j}=0$. At last $R_i$ is the radius of the integration sphere around $B(\boldsymbol{v},\boldsymbol{u}_i)$.

In our MCRT direction $\boldsymbol{u}$ of the reflection of the camera ray is sampled according to BDF and thus its angular density is

$$\rho(\boldsymbol{u}|\boldsymbol{v},\boldsymbol{n}_{A(\boldsymbol{v})}) = \frac{f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})}{\int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})d^2\boldsymbol{u}} \tag{2}$$

If contribution of the $i$-th camera ray (and *all* FMCRT rays) is

$$F(\boldsymbol{v},\boldsymbol{u}_i) \equiv \frac{1}{N_F}\sum_{j=1}^{N_F} f_{B(\boldsymbol{v},\boldsymbol{u}_i)}(\boldsymbol{u}_i,\boldsymbol{V}_j;\boldsymbol{n}_{B(\boldsymbol{v},\boldsymbol{u}_i)})\chi_{i,j}\frac{E_j}{\pi R_i^2} \tag{3}$$

then the increment of the pixel luminance from the $i$-th ray can be written as

$$\Delta L(\boldsymbol{u}_i) = \frac{1}{N_B}\left(\int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})d^2\boldsymbol{u}\right)F(\boldsymbol{v},\boldsymbol{u}_i) \tag{4}$$

## 3 OBSERVED LUMINANCE OF SCENE

The expression (3) is the estimate of luminance observed in direction $\boldsymbol{u}_i$ from the point $A(\boldsymbol{v})$.

Indeed, the expectation of the value (3) *over the set of FMCRT rays* (but *fixed* camera ray path!) is

$$\langle F(\boldsymbol{v},\boldsymbol{u}_i)\rangle = a(\boldsymbol{u}_i)L_{B(\boldsymbol{v},\boldsymbol{u}_i)}(\boldsymbol{w}_i) \tag{5}$$

where $a(\boldsymbol{u}_i)$ is attenuation (volumetric or by specular surfaces) along this camera path, started from $A(\boldsymbol{v})$ in direction $\boldsymbol{u}_i$ and ended in $B(\boldsymbol{v},\boldsymbol{u}_i)$ while having direction $\boldsymbol{w}_i$, and $L_{B(\boldsymbol{v},\boldsymbol{u}_i)}(\boldsymbol{w}_i)$ is luminance of the point $B(\boldsymbol{v},\boldsymbol{u}_i)$ in direction $\boldsymbol{w}_i$. Let us note that if there are no specular surfaces in the camera ray path, $B(\boldsymbol{v},\boldsymbol{u}_i)$ is fixed and $\boldsymbol{w}_i = \boldsymbol{u}_i$. If there are specular surfaces in that path, $B(\boldsymbol{v},\boldsymbol{u}_i)$ and $\boldsymbol{w}_i$ *are random.*

The expectation of (5) with respect to the camera ray trajectory *after* reflection in $A(\boldsymbol{v})$ in the *given* direction $\boldsymbol{u}$ is

$$\langle\langle F(\boldsymbol{v},\boldsymbol{u})\rangle\rangle = \mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u}) \tag{6}$$

where $\mathcal{L}_X(\boldsymbol{u})$ is the luminance of the whole scene seen from $X$ in direction $\boldsymbol{u}$.

The expectation of the pixel luminance, i.e the average of (1) over both FMCRT and BMCRT sets can then be written as

$$\langle\langle L\rangle\rangle = \int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u})\Theta((\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))d^2\boldsymbol{u} \tag{7}$$

## 4 IMAGE-BASED FILTER

The first of the suggested methods processes the noisy image after completion of ray tracing.

The main idea behind it is the usage of the "pivot" image obtained by a deterministic ray tracing under a unit ambient illumination. The original noisy image is firstly divided by that pivot and then this ratio (usually it is a smooth large-scale function unless noise) is processed by a usual filter like bilateral. After multiplying the result back by the pivot image one obtains the denoised image which retains the fine structure.

This filter can be applied for both direct and caustic illumination as well, but is most advantageous for the diffuse (secondary) illumination.

### 4.1 What to filter: "pseudo-brightness"

Let dependence of the BDF on space point be just its modulation by *single* texture $t$. In this case (7) reduces to

$$\langle\langle L\rangle\rangle = t_{A(\boldsymbol{v})} \int f(\boldsymbol{v}, \boldsymbol{u}; \boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u} \cdot \boldsymbol{n}_{A(\boldsymbol{v})})\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u})\Theta((\boldsymbol{u} \cdot \boldsymbol{n}_{A(\boldsymbol{v})}))d^2\boldsymbol{u} \tag{8}$$

In the image part which does not contain boundaries of the scene objects, small-scale components of luminance are caused by either texture, or variation of normal (relief), or caustic illumination.

Frequently the reason is texture, so (8) is a product of a deterministic modulator $t$ and a slowly changing integral.

While ray tracing we get the hit point $A(\boldsymbol{v})$ and thus can obtain the texture value in it. Dividing the (noisy) image by it we get a noisy estimate of a slowly changing field $\int f(\boldsymbol{v}, \boldsymbol{u}; \boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u} \cdot \boldsymbol{n}_{A(\boldsymbol{v})})\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u})\Theta((\boldsymbol{u} \cdot \boldsymbol{n}_{A(\boldsymbol{v})}))d^2\boldsymbol{u}$, which thus admits efficient filtering with a "conventional" image filter, e.g. bilateral. Back multiplying the result by the texture one gets the denoised image which retains the small-scale details.

This algorithm is computationally inexpensive as compared to that described in Section 5 concerning both memory and time. It has, however, stronger limitations: does not work with multiple textures or surface relief, see Section 7.

There is a slight modification that does not require extension of tracer to retrieve texture. In some cases it provides better results, e.g. for multiple textures (when the surface BDF is a sum of several ones, each modulated with own texture), although formally this is not supported.

Instead of texture it uses the so-called "pivot" image calculated with a *deterministic* method for a uniform ambient illumination. Pixel luminance under this illumination is

$$L_a = t_{A(\boldsymbol{v})} \int f(\boldsymbol{v}, \boldsymbol{u}; \boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u} \cdot \boldsymbol{n}_{A(\boldsymbol{v})})\Theta((\boldsymbol{u} \cdot \boldsymbol{n}_{A(\boldsymbol{v})}))d^2\boldsymbol{u} \tag{9}$$

Usually both this and the "main" image have the same small-scale structure, the difference being rather slowly varying brightness (due to different illumination), see Figure 1. That is, the ratio

$$z = \frac{L}{L_a}$$

is a noisy function with *slowly changing expectation.* Indeed, if the fast variation of luminance is due to texture, then the integrals in both (8) and (9) are smooth fields, the more so being their ratio $\langle\langle z\rangle\rangle$. The field $z$ is termed "pseudo-brightness".

After filtering this $z$ with a "usual" image filter (e.g. bilateral) and back multiplying the result by $L_a$ one obtains the denoised image which retains the small-scale texture details.

Steep changes of $\langle\langle z\rangle\rangle$ which would result in image distortion are possible in case of

- large gradients of illumination, e.g. in highlight spots, caustics, boundaries of shadows or near light sources. In most cases, though, the secondary illumination varies slowly and the above effects are not fatal for filtering;
- fast variations of the normal field, e.g. for surface relief.

Figure 1: The effect of illumination. The left image was calculated with bi-directional ray tracing for the real illumination. The right image was calculated by a deterministic rendering assuming unit ambient illumination.

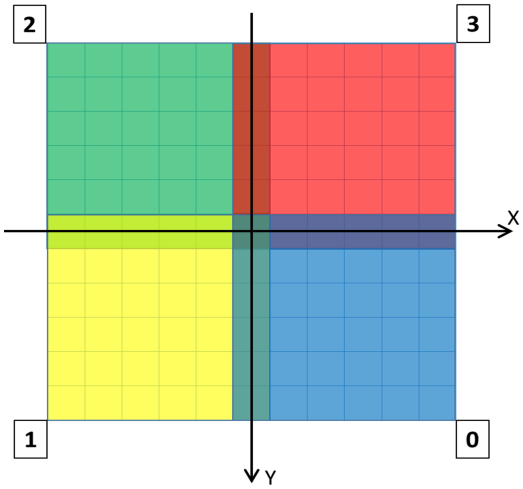


Figure 2: Filtering area and its 4 quadrants.

A natural remedy against the above is to exclude pixels where $\langle\langle z\rangle\rangle$ is not smooth enough. For example, use only areas where it admits a *local* linear or parabolic fitting. The difficulty is that we do not have $\langle\langle z\rangle\rangle$ but only $z$ where it is difficult to distinguish between the noise and large gradients.

### 4.2 Local approximation to $z(x, y)$

Filtering of a noisy function over the area $\mathcal{R}(x_0, y_0)$ around the target pixel $(x_0, y_0)$ can be formulated as construction of a smooth approximation in that area so that its value at $(x_0, y_0)$ gives the filtered function.

In our case the filtering area is always a *square* $\mathcal{R}(x_0, y_0)$ with centre at $(x_0, y_0)$, see Figure 2. How the size of the square is chosen is described in Section 4.3.

For both the whole square and its 4 quadrants we calculate the best-fit plane $Z = ax + by + c$:

$$\sum_{x,y} w_a(x, y)(ax + by + c - z(x, y))^2 = \text{min!}$$

where $(x, y)$ are the pixel coordinates and $w_a$ is the weight somewhat similar to that in a bilateral filter:

$$\begin{aligned} w_a(x, y) &\equiv \exp\left(-k\frac{|z(x, y) - \langle z\rangle(x, y)|}{\langle z\rangle(x, y)} - \sigma(x, y) - \frac{(x - x_0)^2 + (y - y_0)^2}{R(x_0, y_0)}\right) \quad (10) \\ \langle z\rangle(x, y) &\equiv \frac{1}{N}\sum_{x,y} z(x, y) \end{aligned}$$

Here $R(x_0, y_0)$ is the "radius" of filter, i.e the size of the square, $\sigma(x, y)$ is the estimated variance of the value to fit. The above fitting includes only the "good" pixels i.e. those where the noise $\sigma(x, y)$ is below the allowed threshold.

The best fit plane for the whole square is denoted $Z(x, y)$, while $Z_m(x, y)$ is the one for the $m$-th quadrant.

The final approximation to $z(x, y)$ is then defined as

$$\overline{z}(x_0,y_0; x, y) = \frac{Z(x, y) + w_m Z_m(x, y)}{1 + w_m} \tag{11}$$

where $m$ is the index of the quarter which includes $(x, y)$, and the weight $w_m$ is defined as

$$w_m(x_0,y_0; x, y) = \max\left(\frac{Z(x, y)}{Z_m(x, y)}, \frac{Z_m(x, y)}{Z(x, y)}\right)^8 \tag{12}$$

### 4.3 Choosing radius of filter

The search for the optimal size of the square begins with $5 \times 5$ pixels, increasing it until one of the stop criterion is satisfied:

- The size achieved is sufficient to reduce noise below the target value.
- The size reached allowed maximum.
- Too strong variation of luminance over the area, e.g. the area includes shadow boundary. This criterion means that the difference between $Z(x_0, y_0)$ and four $Z_m(x_0, y_0)$ defined in Section 4.2 exceeds allowed maximum.

If none of the above criteria is satisfied we increase the size by one pixel and check again.

It should be noted that strong variation of normal and discontinuities in the projection of the visible scene geometry are not stop criteria. Instead the pixels where these variations are too strong are effectively excluded from fitting.

### 4.4 Two stages of averaging and the filtered value

Usually the filtered value in $(x_0, y_0)$ is calculated by finding the smooth fitting in the filter area around $(x_0, y_0)$ and evaluating the fitting function in $(x_0, y_0)$.

We however use slightly different method. Again, we find the best-fit smooth function (11). But then it is evaluated in *all* pixels in the filter area and these values are *added* to the accumulated result of filtering with weight (12). Therefore, the resulting filtered image is a weighted sum over all filter squares $\mathcal{R}(x_0, y_0)$ that include the given pixel $(x, y)$:

$$\zeta(x, y) = \sum_{x_0,y_0:\ \mathcal{R}(x_0,y_0)\ni(x,y)} w(x, y;\ x_0, y_0)\overline{z}(x_0,y_0; x, y) \tag{13}$$

In most cases this method allows to decrease the error of approximation. For example, let us apply filtering to a noise-free 1D function $f(x) = x^2$. An ideal filter should yield that same function, but a real one slightly distorts it. The linear approximation in a cell of size $\Delta$ has accuracy about $\frac{\Delta^2}{12}$. Therefore, when calculating the value of the filtered function in the center of cell we introduce distortion about $\frac{\Delta^2}{12}$. Meanwhile if it is calculated as an average over all cells including the target point, this value has smaller deviation from the

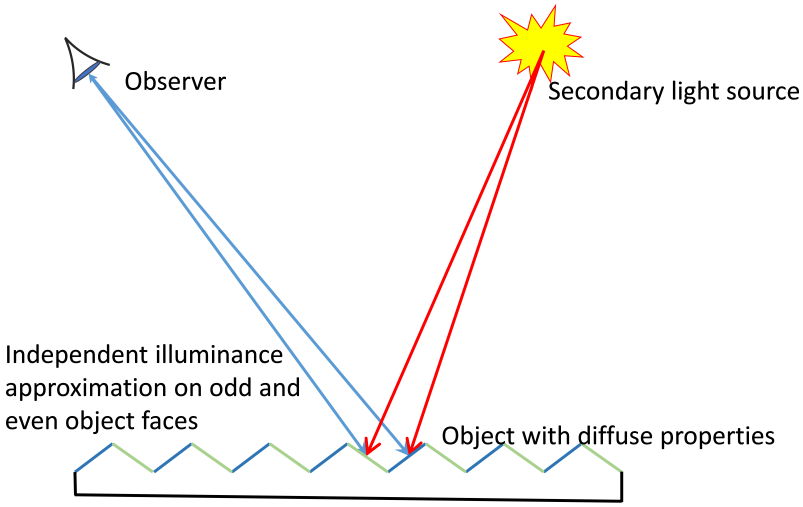


Figure 3: Approximation of illumination for a saw-teeth profile. Illumination of each even face of a tooth is approximated only from illumination of even faces (i.e. half of area) while illumination of an odd face is approximated only from that of another odd faces.

exact $x^2$. If the averaging weight is constant this deviation is exactly 0 and for a variable weight it is still much less than $\frac{\Delta^2}{12}$.

The above example is not a proof, but usually the approach described is really advantageous and it is used to calculate the filtered image.

The weight $w(x, y;\, x_0, y_0)$ ranges from 0 thru 1. It is defined as a product of the base weight (12) and several additional components:

- The term due to the difference of normals:
- $$w_n(x, y;\, x_0, y_0) = \begin{cases} ((\boldsymbol{n}(x, y)\cdot\boldsymbol{n}(x_0, y_0))^4, & (\boldsymbol{n}(x, y)\cdot\boldsymbol{n}(x_0, y_0) > \frac{1}{2} \\ 0, & (\boldsymbol{n}(x, y)\cdot\boldsymbol{n}(x_0, y_0) \leq \frac{1}{2} \end{cases}$$

  This component allows to have sharp boundaries of scene objects and well approximate a saw-teeth profile like shown in Figure 3. Illumination of each even face of a tooth is approximated only from illumination of even faces (i.e. half of area) while illumination of an odd face is approximated only from that of another odd faces.

- The component $w_b$ for a boundary of a scene object. An example of an object boundary as a step profile is shown in Figure 4. When approximating the point $A$ the domain in the bottom half has zero weight, i.e. is effectively excluded. Be the weight unit, the step would visually disappear because illumination would be smoothed to nearly uniform, while the luminance factor (BDF) for both halves is the same. As a result the filtered luminance is constant across the step. Therefore $w_b = 0$ near projections of the boundaries of the scene objects.

- The component to not intermix different "parts". In rendering, adjacent pixels can be projected onto different objects, or to different *parts* of the same object which have different optical properties. Besides, in scenes with specular elements, while looking at the partially specular floor camera sees both that floor and also reflection

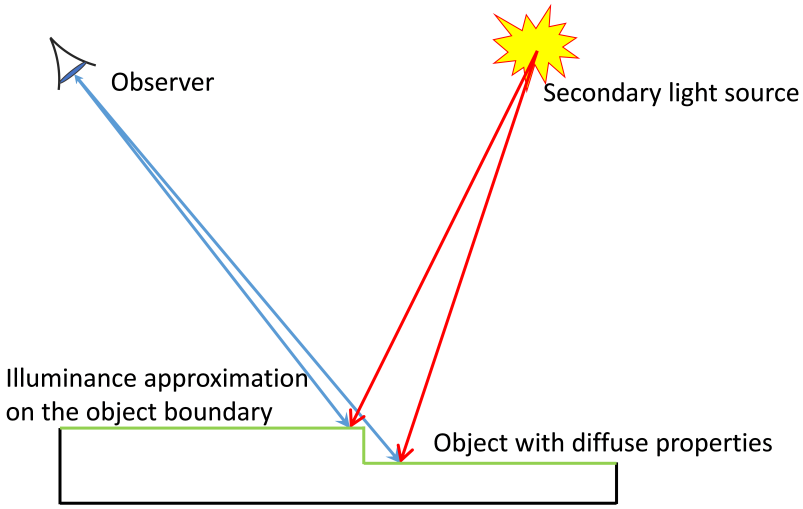


Figure 4: Approximation of illumination near a boundary of a scene object.

of the rest scene in it. In this case this weight component ranges from 1 when all visible objects and their parts in pixels $(x_0, y_0)$ and $(x, y)$ are the same to 0 when they are different.

After processing all central pixels $(x_0, y_0)$ we obtain the filtered field $\zeta(x, y)$ given by (13). It is then back multiplied by the pivot image $L_a$

$$L(x, y) = L_a(x, y)\zeta(x, y)$$

This is the final denoised image.

## 5 RAY TRACING FILTER

The second of the suggested methods is integrated into the process of bi-directional ray tracing with the use of photon maps. It affects only the *secondary* (diffuse) illumination component of the image. While formally it can also be applied to the direct and caustic components, the gain is very limited.

The basic idea behind the method is that pixel luminance is calculated using camera rays traced through *neighbor* pixels, i.e. it is somehow similar to *mutations* of ray paths. The part of the path *before* the first diffuse hit is taken for this pixel, while the rest part (*after* it) is from a neighbor pixel. The luminance is then calculated for this glued trajectory.

### 5.1 Scenes where this method is most advantageous

The method is rather universal and can work with a wide class of scenes, including those where the camera ray undergoes multiple specular scattering. However it can be better explained for a scene like shown in Figure 5.

Camera sees the right cylinder directly. The camera ray reflects from it and then either leaves the scene or hits the left cylinder illuminated by diffuse light and there

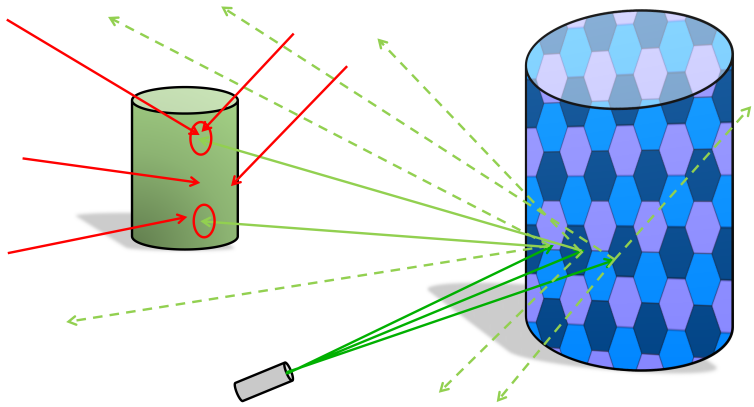

Figure 5: Example scene consisting of two diffuse cylinders, camera and lights (not shown). Red rays are from lights. Green rays are from camera. Dashed rays are those which do not contribute to the image. In this example BDD=1.

collects illumination brought by those FMCRT rays which hit its integration circle. The first hit point (in the right cylinder surface) collects only direct and caustic illumination which we suppose does not reach that surface.

The noise in a scene of such type mainly arises because

- many camera rays after reflection miss the 2nd cylinder and thus do not contribute to the image.
- the density of FMCRT hits for the left cylinder is low and thus even those camera rays which reach it estimate illumination with too high variance.

### 5.2 Filter

Filter area consists of the central pixel $p$ and the "periphery" $\{p'\} = \mathcal{F}_p$ around it.

Now let that whenever a reflected camera ray brings luminance to $p$, it is also applied to all peripheral pixels in $\mathcal{F}_p$ incrementing their luminance.

Let its contribution to the luminance of $p$ itself be $w_{p\leftarrow p}(\boldsymbol{u}_i)\Delta L_{p\leftarrow p}(\boldsymbol{u}_i)$ where $\Delta L_{p\leftarrow p}(\boldsymbol{u}_i)$ is given by (4), the $w_{p\leftarrow p}(\boldsymbol{u}_i)$ being the yet unknown "self-weight". Let also the contribution of that same ray to the luminance of a pixel $p' \in \mathcal{F}_p$ be

$$\Delta L_{p'\leftarrow p}(\boldsymbol{u}_i) = w_{p'\leftarrow p}(\boldsymbol{u}_i) C_{p'\leftarrow p}(\boldsymbol{u}_i) \Delta L_{p\leftarrow p}(\boldsymbol{u}_i)$$

where

$$C_{p'\leftarrow p}(\boldsymbol{u}) = \frac{f_{A(\boldsymbol{v}')}(\boldsymbol{v}', \boldsymbol{u}; \boldsymbol{n}_{A(\boldsymbol{v}')})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})}{f_{A(\boldsymbol{v})}(\boldsymbol{v}, \boldsymbol{u}; \boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})}\Theta((\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})) \tag{14}$$

and $w_{p'\leftarrow p}(\boldsymbol{u}_i)$ is an "inter-pixel" weight that can depend on any camera ray parameters for these pixels, such as positions of their first hit points, the normals to the surface in the first hit points, BDF, directions of segments of the camera ray, etc., The step function ($\Theta(x) = 0$ for $x < 0$ and 1 otherwise) ensures that only those camera rays which are *above* the local horizon of $p'$ are used, i.e. those for which $(\boldsymbol{u}_i \cdot \boldsymbol{n}_{A(\boldsymbol{v}')}) \geq 0$.

As a result, a camera ray fired through $p'$ increments the luminance of $p$ by $w_{p\leftarrow p'}(\boldsymbol{u}'_i)C_{p\leftarrow p'}(\boldsymbol{u}'_i)\Delta L_{p'\leftarrow p'}(\boldsymbol{u}'_i)$ so its luminance accumulated after tracing of $N_B$ BMCRT rays and $N_F$ FMCRT rays is

$$\begin{aligned}\widetilde{L}_p &= \sum_{i=1}^{N_B} w_{p\leftarrow p}(\boldsymbol{u}_i)\Delta L_{p\to p}(\boldsymbol{u}_i) \\ &\quad + \sum_{i=1}^{N_B}\sum_{p'\in\mathcal{F}_p} w_{p\leftarrow p'}(\boldsymbol{u}'_i)C_{p\leftarrow p'}(\boldsymbol{u}'_i)\Delta L_{p'\leftarrow p'}(\boldsymbol{u}'_i)\Theta((\boldsymbol{u}'_i\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))\end{aligned}$$

In view of (4) this can be rewritten as

$$\begin{aligned}\widetilde{L}_p &= \left(\int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})d^2\boldsymbol{u}\right)\frac{1}{N_B}\sum_{i=1}^{N_B} w_{p\leftarrow p}(\boldsymbol{u}_i)F(\boldsymbol{v},\boldsymbol{u}_i) \\ &+ \left(\int f_{A(\boldsymbol{v}')}(\boldsymbol{v}',\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v}')})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})d^2\boldsymbol{u}\right) \\ &\quad\times\sum_{p'\in\mathcal{F}_p}\frac{1}{N_B}\sum_{i=1}^{N_B} C_{p\leftarrow p'}(\boldsymbol{u}'_i)w_{p\leftarrow p'}(\boldsymbol{u}'_i)F(\boldsymbol{v}',\boldsymbol{u}'_i)\Theta((\boldsymbol{u}'_i\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))\end{aligned}$$

where $\{\boldsymbol{u}'_i\}$ and $\{\boldsymbol{u}_i\}$ are the directions (of the 2nd segment) of camera rays fired through $p'$ and $p$, respectively.

The FMCRT ray set is independent from our new rule of the luminance calculation. Averaging over that set and over the trajectory of the camera ray *after* the first diffuse hit in $A(\boldsymbol{v})$, one obtains with the help of (6) that

$$\begin{aligned}\langle\widetilde{L}_p\rangle &= \left(\int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})d^2\boldsymbol{u}\right)\frac{1}{N_B}\sum_{i=1}^{N_B} w_{p\leftarrow p}(\boldsymbol{u}_i)\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u}_i) \\ &+ \left(\int f_{A(\boldsymbol{v}')}(\boldsymbol{v}',\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v}')})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})d^2\boldsymbol{u}\right) \\ &\quad\times\sum_{p'\in\mathcal{F}_p}\frac{1}{N_B}\sum_{i=1}^{N_B} w_{p\leftarrow p'}(\boldsymbol{u}'_i)C_{p\leftarrow p'}(\boldsymbol{u}'_i)\mathcal{L}_{A(\boldsymbol{v}')}(\boldsymbol{u}'_i)\Theta((\boldsymbol{u}'_i\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))\end{aligned}$$

Directions $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{u}'_i\}$ are those of *really traced rays* after reflection in $A(\boldsymbol{v})$ or $A(\boldsymbol{v}')$, respectively. Therefore they are distributed with density $\rho(\boldsymbol{u}|\boldsymbol{v},\boldsymbol{n}_{A(\boldsymbol{v})})$, respectively $\rho(\boldsymbol{u}'|\boldsymbol{v}',\boldsymbol{n}_{A(\boldsymbol{v}')})$, given by (2). Averaging over that directions and applying (14) for *swapped* pixels $p\leftrightarrow p'$, one arrives at

$$
\begin{aligned}
\langle\langle \widetilde{L}_p \rangle\rangle &= \int w_{p\leftarrow p}(\boldsymbol{u}) f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u})d^2\boldsymbol{u} \\
&+ \sum_{p'\in\mathcal{F}_p}\int w_{p\leftarrow p'} f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})\mathcal{L}_{A(\boldsymbol{v}')}(\boldsymbol{u})\Theta((\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))d^2\boldsymbol{u} \\
&= \int f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})\widetilde{\mathcal{L}}_{A(\boldsymbol{v})}(\boldsymbol{u})d^2\boldsymbol{u}
\end{aligned}
$$

Comparing the last line with (7) we recognize the luminance of $p$ under *filtered illumination*

$$
\widetilde{\mathcal{L}}_{A(\boldsymbol{v})}(\boldsymbol{u}) \equiv w_{p\leftarrow p}(\boldsymbol{u})\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u}) + \sum_{p'\in\mathcal{F}_p} w_{p\leftarrow p'}\mathcal{L}_{A(\boldsymbol{v}')}(\boldsymbol{u})\Theta((\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})). \tag{15}
$$

If illumination of the surface of the right cylinder varies slowly, the difference between $\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u})$ and $\mathcal{L}_{A(\boldsymbol{v}')}(\boldsymbol{u})$ can be neglected and then

$$
\begin{aligned}
\langle\langle \widetilde{L}_p \rangle\rangle \approx \int &\left( w_{p\leftarrow p}(\boldsymbol{u}) + \sum_{p'\in\mathcal{F}_p} w_{p\leftarrow p'}(\boldsymbol{u})\Theta((\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})) \right) \\
&f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})\mathcal{L}_{A(\boldsymbol{v})}(\boldsymbol{u})d^2\boldsymbol{u}
\end{aligned}
$$

For this to match the exact value (7) it suffices that

$$
w_{p\leftarrow p}(\boldsymbol{u}) = 1 - \sum_{p'\in\mathcal{F}_p} w_{p\leftarrow p'}(\boldsymbol{u})\Theta((\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})) \tag{16}
$$

The term $\Theta$ means that for directions $\boldsymbol{u}$ that *could not* be emitted for $p'$ and thus come *only* from $p$ itself, the self-weight is 1, while for directions that *could* come from another pixel, the self-weight must be reduced to avoid overestimation from summing both contributions.

Usually, $w_{p\leftarrow p'}(\boldsymbol{u})$ is symmetric against the two pixels and is defined so that it vanishes when

- the distance between $A(\boldsymbol{v})$ and $A(\boldsymbol{v}')$ is too large
- $\boldsymbol{n}_{A(\boldsymbol{v}')}$ is too different from $\boldsymbol{n}_{A(\boldsymbol{v})}$
- $f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})$ is too different from $f_{A(\boldsymbol{v})}(\boldsymbol{v}',\boldsymbol{u}';\boldsymbol{n}_{A(\boldsymbol{v}')})(\boldsymbol{u}'\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})$; notice the BDFs are taken in *the same space point* because otherwise filtering can be effectively disabled in case of modulation by a texture

and besides, the weight is a descending function like Gauss of a distance between the image points $p'$ and $p$. Therefore, the weight can be defined as e.g.

$$
\begin{aligned}
w_{p\leftarrow p'}(\boldsymbol{u}) &= C\exp\left(-\alpha\left\|p-p'\right\|^2\right)\exp\left(-\beta(1-(\boldsymbol{n}_{A(\boldsymbol{v}')}\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))\right) \\
&\quad\times\exp\left(-\gamma F^2\right) \qquad (17)\\
F &\equiv \max\left(\frac{f_{A(\boldsymbol{v})}(\boldsymbol{v}',\boldsymbol{u}';\boldsymbol{n}_{A(\boldsymbol{v}')})(\boldsymbol{u}'\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})}{f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})},\ \frac{f_{A(\boldsymbol{v})}(\boldsymbol{v},\boldsymbol{u};\boldsymbol{n}_{A(\boldsymbol{v})})(\boldsymbol{u}\cdot\boldsymbol{n}_{A(\boldsymbol{v})})}{f_{A(\boldsymbol{v})}(\boldsymbol{v}',\boldsymbol{u}';\boldsymbol{n}_{A(\boldsymbol{v}')})(\boldsymbol{u}'\cdot\boldsymbol{n}_{A(\boldsymbol{v}')})}\right)
\end{aligned}
$$

The constant $C$ and two first terms are independent from the direction of scattering $\boldsymbol{u}$ and therefore are named the "fixed part" of the weight. They can be calculated in advance and kept. The last term is the "variable part" which can *decrease* weight depending on the direction of the scattered ray.

The constant $C < 1$ is chosen so that $w_{p\leftarrow p}(\boldsymbol{u})$ calculated from (16) should be not too small (say, at least 0.5) even for the fixed part of weight:

$$
C\sum_{p'\in\mathcal{F}_p}\exp\left(-\alpha\left\|p-p'\right\|^2\right)\exp\left(-\beta(1-(\boldsymbol{n}_{A(\boldsymbol{v}')}\cdot\boldsymbol{n}_{A(\boldsymbol{v})}))\right) < \frac{1}{2}
$$

and then $w_{p\leftarrow p}(\boldsymbol{u}) > \frac{1}{2}$ for the full weight (17).

## 6 RESULTS FOR THE IMAGE FILTER

Results of the filter are presented in Figures 6 and 7. The top panel shows the reference image, the bottom left panel is the noisy image obtained with bi-directional ray tracing, and the bottom right panel is the filtered image.

Figure 6 is for the well-known scene Cornel Box with textures on the walls. Figure 7 is for a room interior.

One can see that the noise is strongly reduced with small-scale details, textures and boundaries of objects are not distorted.

## 7 RESULTS FOR THE RAY TRACING FILTER

Results of the application of this method are demonstrated for three scenes.

### 7.1 Plane with relief

Scene is a $2\times 2$ meter square, illuminated by a point light in a matte (pure diffuse scattering) spherical shade of radius 10 cm located 1.5 meter above the square centre. The square surface has a random relief and a Gaussian BRDF (integral reflectance 0.9 and width 90°), modulated by texture of three letters "ABC".

Figure 8 shows the reference image obtained by ray tracing in 4000 seconds without filtering, the original image (obtained by bi-directional stochastic ray tracing in 200 sec), and the results of application of the two suggested filters to this calculation. Radius of both filters was 30 pixels. One can see that both filters greatly reduced noise, but the "image filter" blurred the sharp boundaries of the relief facets and overestimated the brightness of the texture letters "ABC".

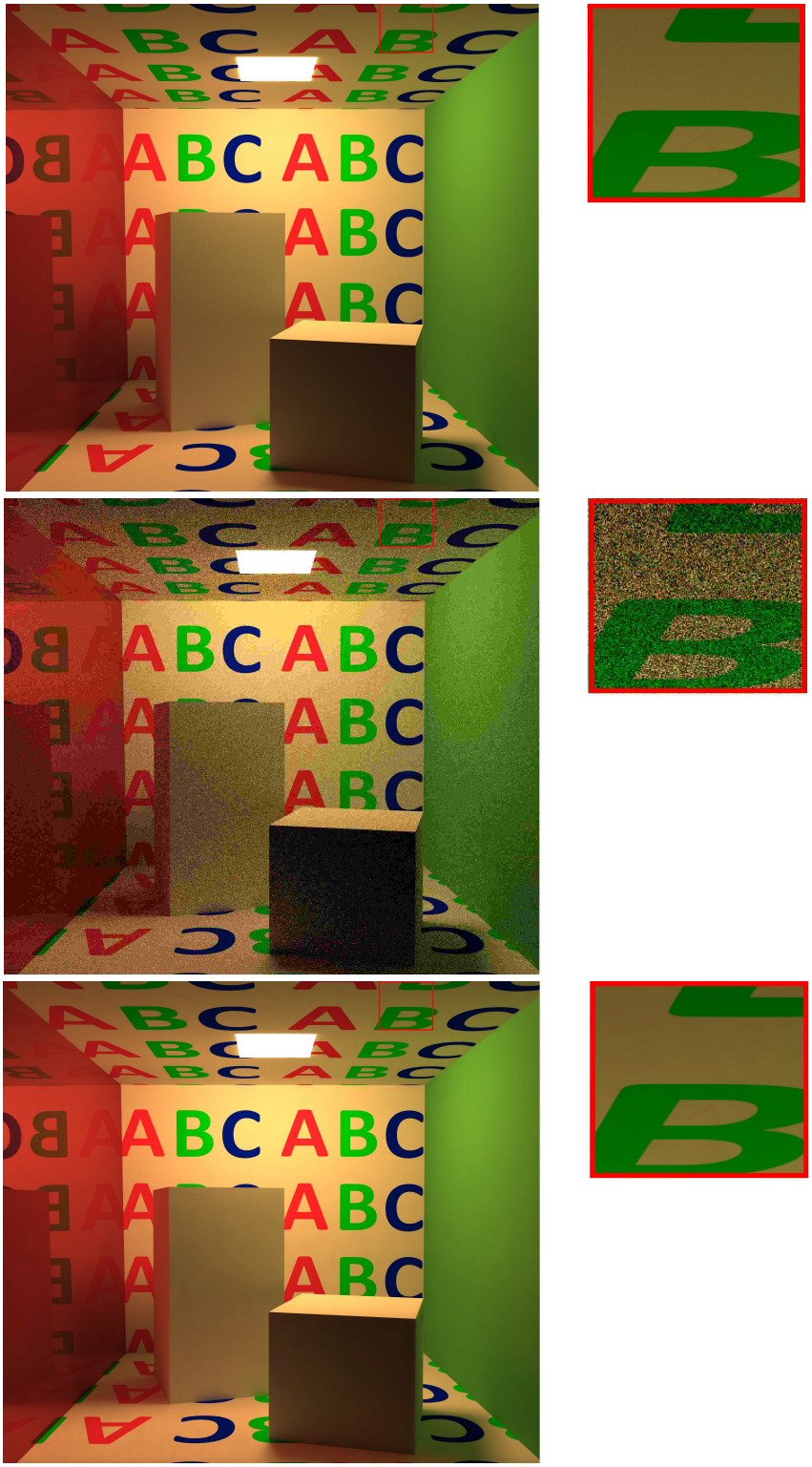

Figure 6: Scene Cornell Box with textures on the walls. Top to bottom: the reference image, the noisy image obtained with bi-directional ray tracing, the filtered image. Next to each image an enlarged ceiling rectangle is shown.

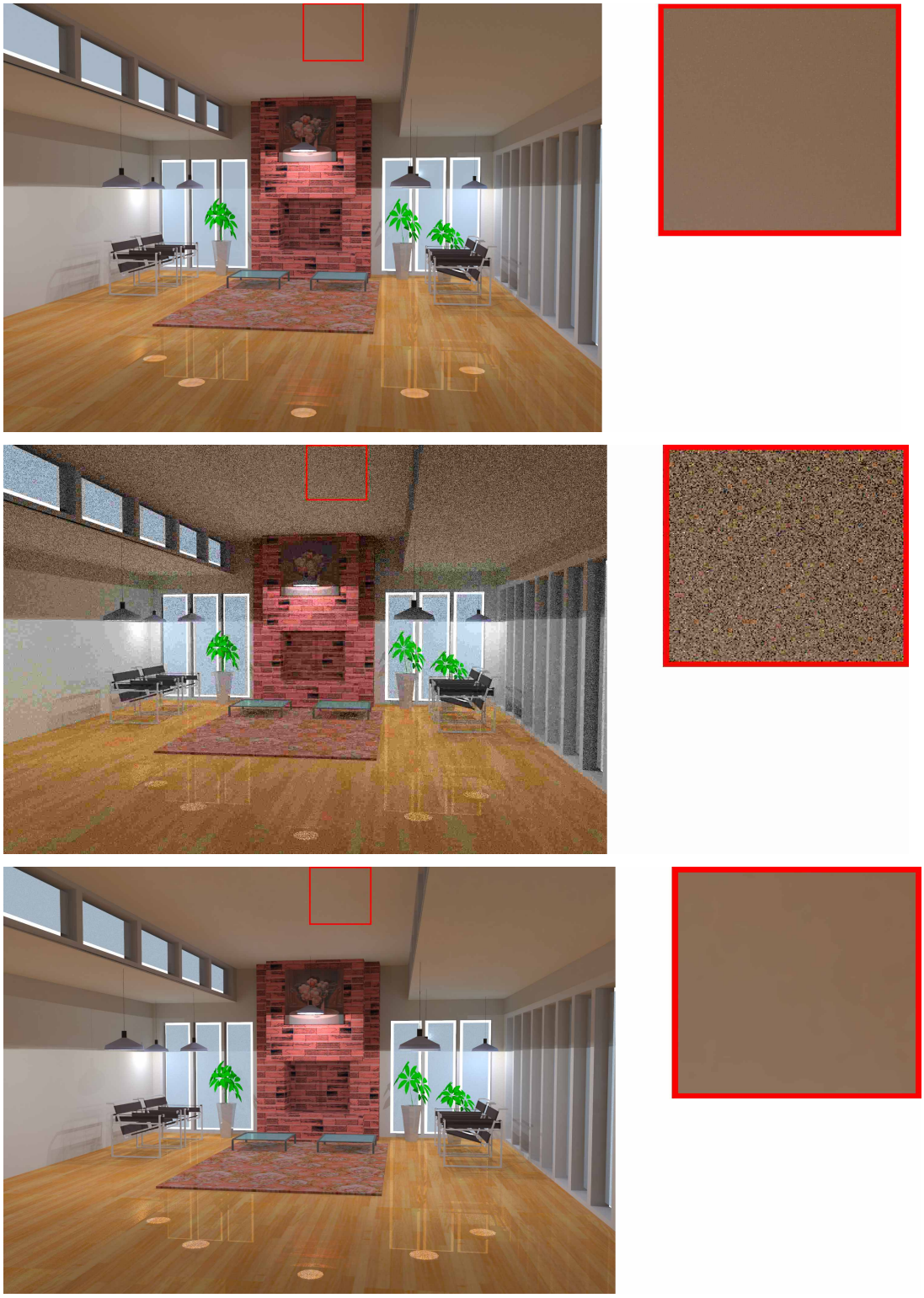

Figure 7: Scene for interior of a room with partially specular oor. Top to bottom: the reference image, the noisy image obtained with bi-directional ray tracing, the filtered image. Next to each image an enlarged ceiling rectangle is shown.

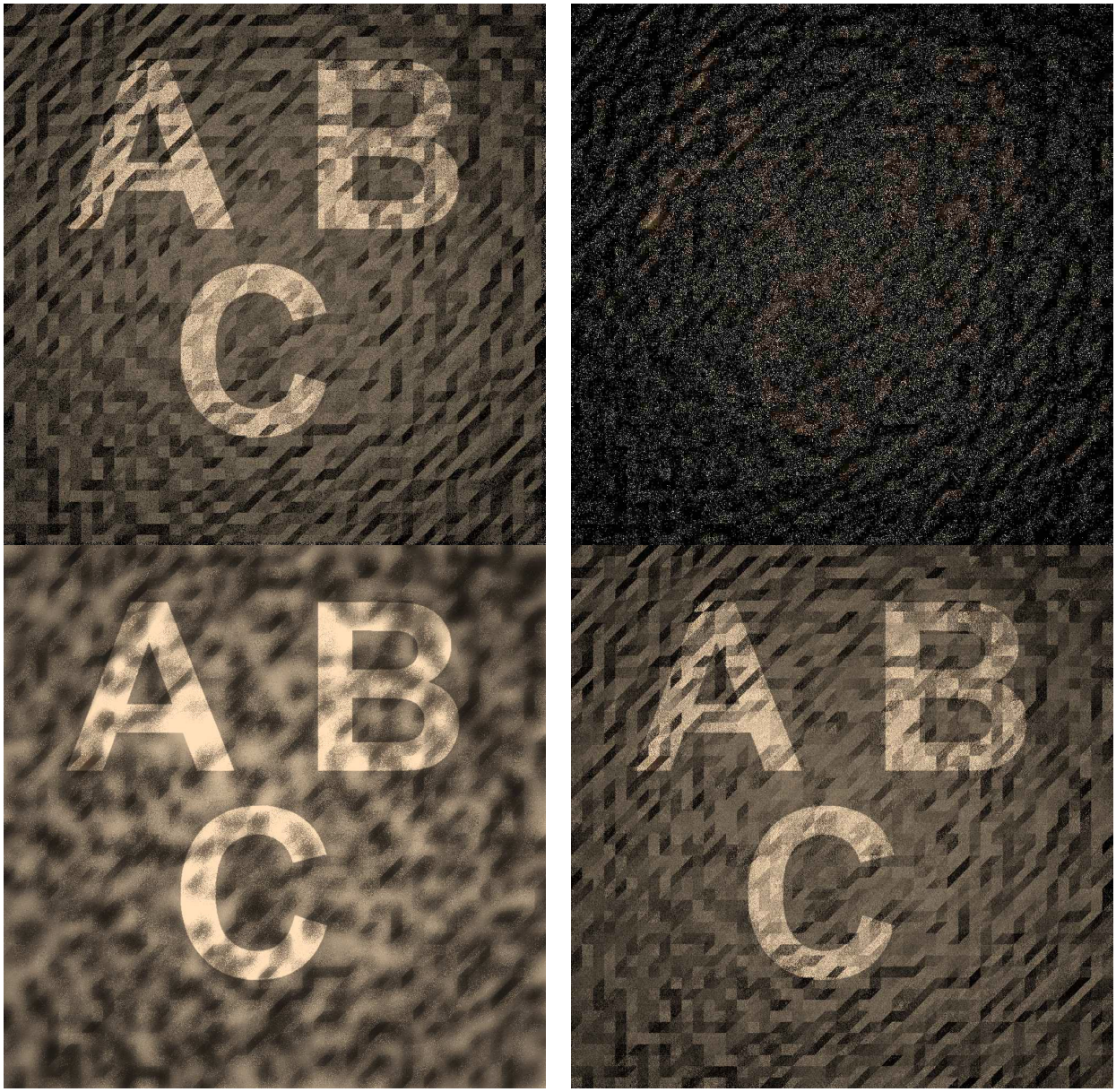


Figure 8: Top row: reference image, original image. Bottom row: denoising with the image filter, denoising with the tracing filter.

### 7.2 Double textured plane

Scene is a $5 \times 5$ meter size square, illuminated by a point light in a matte (pure diffuse scattering) spherical shade of radius 30 cm located 1.5 meter above the square centre. BDF of the surface is a sum of Lambert component (albedo 0.5) and Gaussian component (integral reflectance 0.5). The Lambert component is modulated by the chessboard texture, and the Gaussian component is modulated by the texture of periodically repeated text "FRONT".

Figure 9 shows the reference image obtained by ray tracing in 4000 seconds without filtering, the original image (obtained by bi-directional stochastic ray tracing in 100 sec), and the results of application of the two suggested filters to this calculation. Radius of both filters was 30 pixels. One can see that both filters greatly reduced noise, but the "image filter" produced some artifacts: first, halo around the white squares of the chessboard and second, overestimated the contrast of text in the upper part.

### 7.3 Scene with specular reflections

This is a well known test scene "Cornell Box" with surface attributes changed. Now there is 20% of mirror-like BRDF and about 60% of Gaussian BRDF of width 60°. For the two front boxes the diffuse BRDF is modulated by the texture of letters "ABC".

Figure 10 shows the reference image obtained by ray tracing in 7000 seconds without filtering, the original image (obtained by bi-directional stochastic ray tracing in 140 sec), and the results of application of the two suggested filters to this calculation. Radius of both filters was 40 pixels. One can see that both filters greatly reduced noise, but the "image filter" produced more artifacts (it is more spotted).

## 8 PROCESSING OF SPECULAR EVENTS

In the initial formulation of our methods we assumed that there are no specular events in the camera ray path before the first diffuse hit. However frequently this is not so, e.g. when we look through a window glass or see reflections in a glossy floor. In this case the camera ray hits a specular surface where it can reflect or refract and then travels to different paths. The reflected or refracted rays may in turn hit a specular surface and reflect/refract there and so on.

Both our methods take this effect into account in a similar way.

In stochastic ray tracing we choose reflection or refraction event at random and then continue *this one* ray. In deterministic rendering used to obtain the "pivot" image *both* reflected and refracted rays are traced further. Then each of them can be later split in two on the next specular surface. As a result instead of a single path we have a *tree*. The number of rays grows exponentially so we *limit* the number of specular events processed.

It is possible that a surface can have a mixture of specular and diffuse properties as it is for a lacquered table. In this case the luminance of a pixel is a *sum* over all specular paths plus the "own" luminance of the diffuse part.

Similarly the "exact" image which differs from the pivot in illumination is a sum over all specular paths. Each *type* of specular path (e.g. "reflection–transmission–reflection–reflection") yields a "partial" image. The same is true for the pivot image.

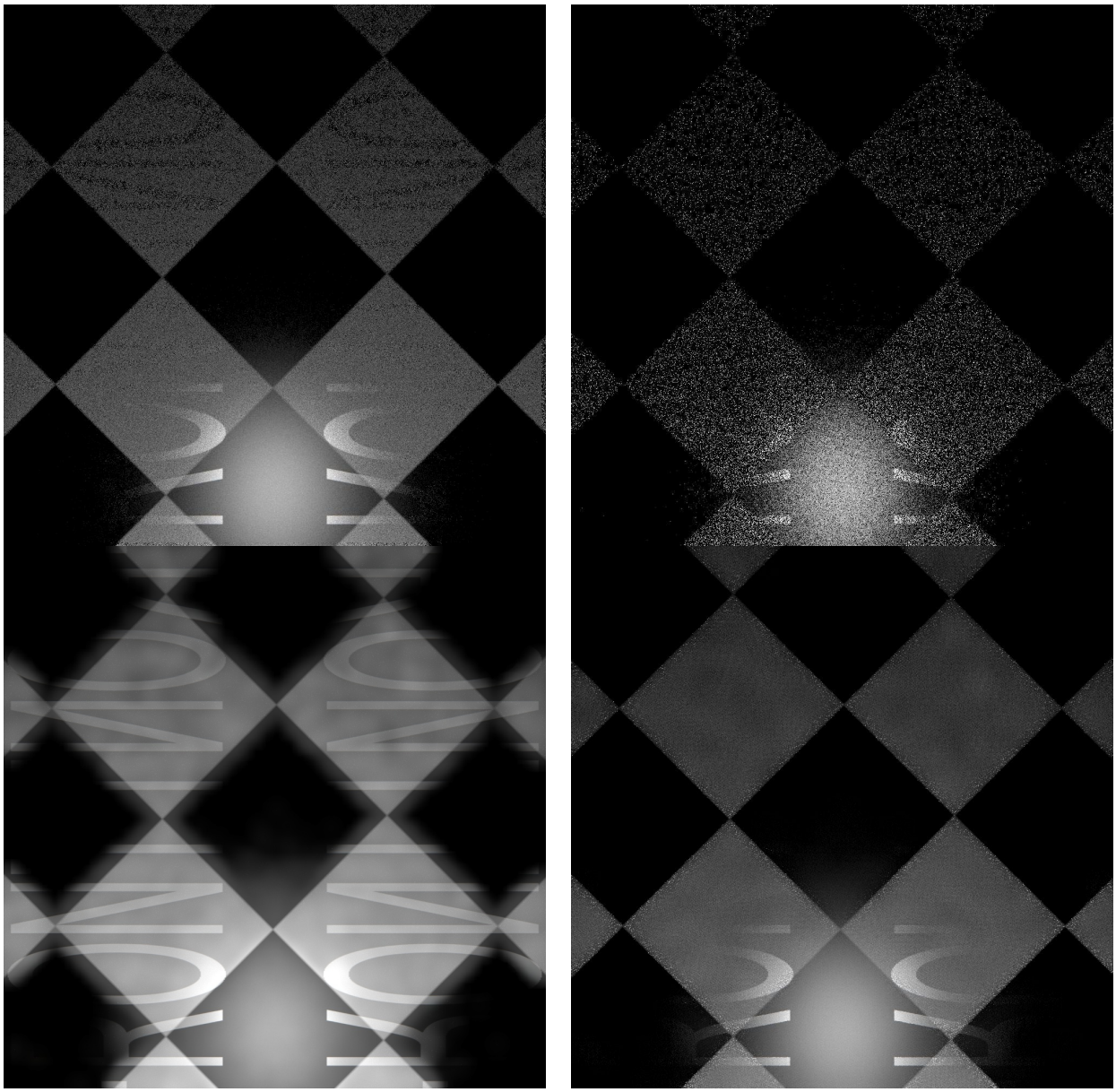

Figure 9: Top row: reference image, original image. Bottom row: denoising with the image filter, denoising with the tracing filter.

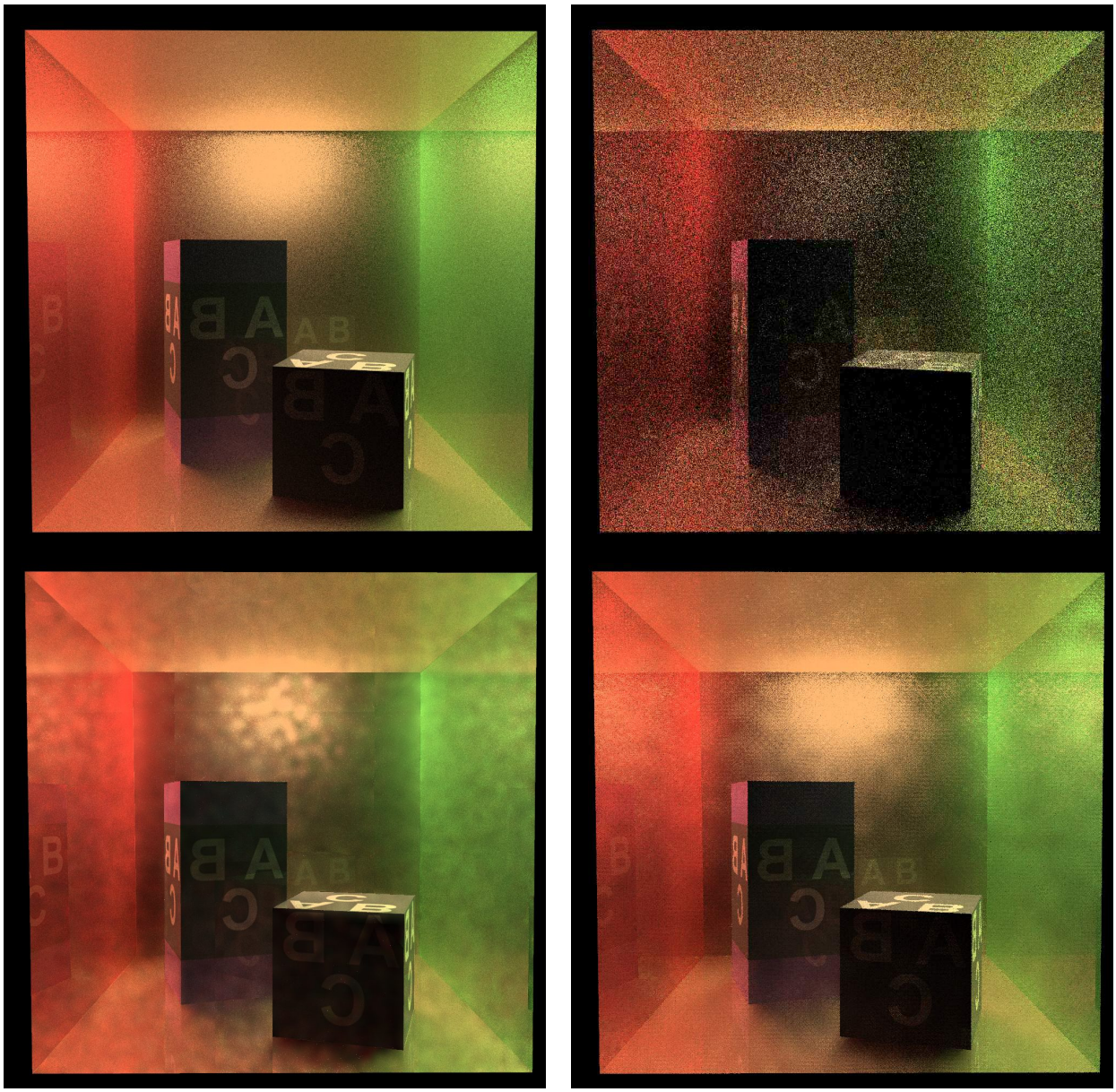

Figure 10: Top row: reference image, original image. Bottom row: denoising with the image filter, denoising with the tracing filter. Shown is only the image component related to diffuse (secondary) illumination.

Each *deterministic* "partial image" has its counterpart as the stochastic partial image; they have *the same* camera ray path before the first hit. But since the "exact" image is an *infinite* sum, there is also the "remainder" which is caused by specular paths whose length is beyond the limit allowed for the deterministic rendering.

Denoising methods from Sections 4 or 5 are then applied to each "partial image" independently and successively. Obviously only a finite number of them can be processed; the rest are left unfiltered. This does not have a strong effect because since the infinite sum *converges*, the remainder is small.

Therefore both BMCRT and deterministic rendering keep the partial images instead of summing them on the fly as usual.

The final result is the sum of the filtered partial images and the unfiltered remainder.

## 9 CONCLUSION

As shown above, both filters well denoise images obtained by bi-directional Monte Carlo ray tracing while preserving fine details. So they allow to reduce the time needed to compute image of the same quality by an order of magnitude.

The image-based filter has more limitations and can produce artifacts in case of multiple textures or surfaces with relief. But for most of "usual" scenes it works good while requires a few extra resources (time and memory). It can be turned on any time when needed.

The ray-tracing filter is more universal and can process correctly those scenes where the image-based filter produces artifacts. But it is more computationally expensive; besides, it must be turned on *in the beginning of ray tracing* and can not be turned on in arbitrary moment later.

**Acknowledgements:** The work was partially supported by RFBR grants, numbers 18-01-00569 and 18-31-20032.

### REFERENCES

[1] H. Jensen and N. Christensen, "Optimizing path tracing using noise reduction filters," in *WSCG* (1995).

[2] F. Suykens and Y. D. Willems, "Adaptive Filtering for Progressive Monte Carlo Image Rendering," in *WSCG* (2000).

[3] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *SIMUL*, **4**, 490–530 (2005).

[4] N. K. Kalantari and P. Sen, "Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms," *Computer Graphics Forum* (2013).

[5] H. Dammertz, D. Sewtz, J. Hanika, and H. Lensch, "Edge-avoiding a-trous wavelet transform for fast global illumination filtering," in *Proc. High Performance Graphics 2010*, 67–75 (2010).

[6] B. Bitterli, F. Rousselle, B. Moon, J. A. Iglesias-Guitian, D. Adler, K. Mitchell, W. Jarosz, and J. Novak, "Nonlinearly weighted first-order regression for denoising Monte Carlo renderings," *Computer Graphics Forum (Proceedings of EGSR)*, **35**, 107–117 (2016).

[7] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *Trans. Img. Proc.*, **16**, 349–366 (2007).

[8] A. Keller, K. Dahm, and N. Binder, "Path space filtering," in *SIGGRAPH Talks*, 68:1, ACM (2014).

[9] P. Sen and S. Darabi, "On filtering the noise from the random parameters in Monte Carlo rendering," *ACM Transactions on Graphics*, **31**, 18:1–18:15 (2012).

[10] M. Mara, M. McGuire, B. Bitterli, and W. Jarosz, "An efficient denoising algorithm for global illumination," in *Proceedings of High Performance Graphics*, (New York, NY, USA), ACM, (2017).

[11] D. D. Zhdanov, S. V. Ershov, and A. G. Voloboy, "A simple image-based filter for MCRT simulation results," *Preprints of KIAM*, no. 194 (2018). doi:10.20948/prepr-2018-194

[12] V. A. Frolov and V. A. Galaktionov, "Memory-compact Metropolis light transport on GPUs," *Programming and Computer Software*, **43**, 196–203 (2017).

[13] Q. Xu, Y. Liu, R. Zhang, S. Bao, R. Scopigno, and M. Sbert, "Noise reduction for path traced imaging of participating media," in *EUSIPCO*, 1683–1687, IEEE (2011).

[14] B. Moon, J. Y. Jun, J. Lee, K. Kim, T. Hachisuka, and S.-E. Yoon, "Robust Image Denoising Using a Virtual Flash Image for Monte Carlo Ray Tracing," *Computer Graphics Forum*, 139–151 (2013).

[15] S. V. Ershov, D. D. Zhdanov, and A. G. Voloboy, "Modification of stochastic ray tracing to reduce noise on diffuse surfaces," *Preprints of KIAM*, no. 204 (2018). doi:10.20948/prepr-2018-204

Received October 10, 2018