Sample-Space Bright Spots Removal Using Density Estimation

Anthony Pajot Loïc Barthe Mathias Paulin
IRIT-CNRS-Université de Toulouse, France

Abstract

Rendering images using Monte-Carlo estimation is prone to bright spots artefacts. Bright spots correspond to high intensity pixels that appear when a very low probability sample outweighs all other sample contributions. We present an average estimator that is robust to outliers, which detects and removes samples that are considered as outliers, and lead to bright spots in images computed using Monte-Carlo estimation. By progressively building a per-pixel representation of the luminance distribution, our method is able to delay samples whose luminance is considered as an outlier with respect to the current distribution. This distribution is continuously updated so that delayed samples may be re-considered as viable later in the rendering process, thus making the presented approach robust. Our method does not suffer from blurring in high-frequency zones. It can be easily integrated in any Monte-Carlo-based rendering system, used in conjunction with any adaptive sampling scheme, and it introduces a very small computational overhead, which is negligible compared to the use of over-sampling.

1 Introduction

Physically-based rendering attempts to solve an integral equation known as the *light transport equation* [8]. This integral links the radiance arriving at a point $\mathbf{x}$ along all possible incident directions to the radiance leaving $\mathbf{x}$ in a direction of interest ω_o. The various terms of this integral are highly varying into the scene, making the integral impossible to solve analytically. Moreover, generalisations of this equation to the so-called "path-space" transform the problem of computing an image into the problem of solving an integral in an infinite-dimensional space [16]. Therefore, it is impossible to use classical deterministic numerical integration algorithms, such as quadrature rules.

Let f be the function giving the energy brought by a direction or a path v to the pixel p, the value (color, spectrum, *etc.*) of each pixel I_{p} is:

$$I_{\mathrm{p}} = \int_{\Omega} f(v)dv. \quad (1)$$

Because of its ease of use and generality, Monte-Carlo estimation is one of the most widely used technique in physically-based rendering algorithms, such as path-tracing [8] and photon-mapping [7]. It consists in randomly sampling N elements v_i using a probability density function p, and then estimating I_{p} as :

$$< I_{\mathrm{p}} >_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(v_i)}{p(v_i)}. \quad (2)$$

Equation (2) tells that the pixel value is computed as the estimated mean value of a positive quantity. This estimator is not robust to outliers, represented here by very large values of $\frac{f(v_i)}{p(v_i)}$. This means that even if a very large number of elements is used to compute each pixel, the presence of only one very large value can lead to an estimation much larger than the actual pixel value. Visually, this generates very bright pixels, denoted as *bright spots* (illustrated in Figure 1) that are still present even in high-quality images. In general, a post-processing such as filtering is thus required in order to get slightly blurred, however bright-spot-free images.

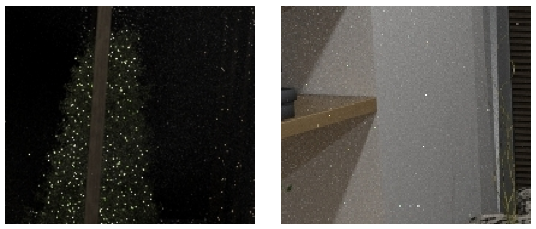
Figure 1: Examples of bright spots, which are very bright pixels surrounded by pixels whose value is much nearer from the real expected value.

The outliers which lead to bright spots are produced by the creation of a sample v whose probability density is low, and which yields a large or very large contribution (a large $f(v)$ value). If $p(v)$ is low, this means that few samples that can lead to bright spots are created when computing an image. Note that this property is true for most rendering algorithms, such as path-tracing [8], bidirectional path-tracing [17, 9], or photon-mapping [7]. However, this does not apply to Metropolis light transport [18] and similar algorithms, as their bright spots are caused by the accumulation of many samples at a single pixel, all the samples having exactly the same luminance.

Our main contribution is a method which detects and delays on the fly outlier values from a set of samples. More specifically for Monte-Carlo-based rendering, it avoids the presence of bright spots in the final image, without introducing blur. As our algorithm only weakly depends on the integration method, it can be seamlessly applied to any Monte-Carlo-based rendering system. As shown in Section 2, many existing methods devoted to removing bright spots try to minimize their visual impact by filtering the final image [1, 10, 12, 14, 20], leading to visible smears or blur. Also, recently, a method using the joint image-color space to detect samples that can cause bright spots has been presented [3]. Similarly to this method, our method is in essence an outlier detection method. This kind of methods is mostly used in data-mining applications. The methods proposed in this field assume a vast amount of data (several million samples), whereas in our case we only have a few tens or hundreds of samples for a given pixel. As presented in Section 3, we approximate the probability distribution of the luminance of the screen samples for each pixel using density estimation. We use this distribution to temporarily discard samples that are susceptible to be outliers, and definitively accept those that are surely not. This probability distribution is updated during the rendering, making our method progressive, and well suited to take advantage of adaptive sampling. As storing all the samples for each pixel would be too costly, we develop compact representations of the distribution in Section 4.2 and Section 4.4. These representations have different and complementary properties with respect to memory cost and precision. As shown in Section 4.6, these two representations can in fact be used jointly, with parameters allowing the user to control the precision/memory cost ratio. Moreover, our method has a time cost which is independent of the scene, and which is shown to be a small fraction of the pure rendering time. Using our method, we obtain images in Section 5 that do not suffer from bright spots, without neither having to use a very large number of samples, nor having images that exhibit smears or blur.

2 Previous Works

The estimator used in Monte-Carlo rendering systems to compute the value of each pixel, which is the mean value estimator, is not robust to outliers. Using over-sampling will not efficiently reduce the impact of an outlier, as a bright spot will be present unless a prohibitively large number of samples are computed for each pixel containing a bright-spot. This explains the existence of methods specifically developed to remove bright spots, which are based on filtering, either at the samples level, or directly on the final image.

2.1 Image-Space Bright Spots Removal

Image-space approaches directly process the final image, using anisotropic diffusion [10] or filtering. Xu *et al.* [20] recently proposed a technique based on bilateral filtering [14]. Bilateral filtering uses two filters to compute each pixel value. The *domain filter*, function of the *positions* of the current pixel p_1 and the neighbor pixel p_2, reinforces the influence of nearby pixels. The *range filter*, function of the *value* of both pixels $f(p_1)$ and $f(p_2)$, reinforces the influence of pixels that have nearby values, using a gaussian filter centered at $f(p_1)$ and with a user-set standard deviation. Xu *et al.*'s method, instead of directly using $f(p_1)$ in the range filter, uses an estimate $\tilde{f}(p_1)$ of the true value of the current pixel, obtained by filtering the neighboring pixels. This modification makes the range kernel less sensitive to outlier pixels when reconstructing their value. This method is highly efficient, and requires a negligible amount of memory. However, even though providing much better results than previous approaches, bright spots caused by diffuse interreflections – which, in general, have lower values than bright spots involving caustics, as their $f(v)$ value is lower – still lead to visible smears, making high-frequency textures blurred. Moreover, for the range filter to perform correctly, its standard deviation must be chosen carefully, taking into account the typical orders of magnitude of the samples for the current scene, and the variance of the underlying integration algorithm. Yet more recently, Dammertz *et al.*[1] used a wavelet-based filtering to better approximate the hemispherical integrals that are typically computed in Monte-Carlo rendering from low-samples estimates, while avoiding the edges of features to avoid blurring these high-frequency elements. It is very efficient for high-frequency/low-amplitude noise on scenes where the illumination changes slowly (typically mostly-diffuse scenes), but it fails when many high-frequency details are present, and no tests with bright-spots were presented. As the edge detection is partially based on the pixel values, it is likely that the bright spots would be considered as single-pixel objects, therefore not being filtered at all.

2.2 Sample-Space Bright Spots Removal

Rushmeier *et al.* [12] changed the way image reconstruction is performed. Instead of using a constant-width filter for all samples, they compute a per-sample width, and use normalized filters to evaluate the contribution of the sample to each pixel. The width is computed so that all samples have comparable contributions, leading to lower high-frequency noise. However, when used on samples causing bright spots, this tends to blur the final image, as the filter width has to cover a very large number of pixels.

DeCoro *et al.* [3] recently proposed to build a tree of the samples described in a joint image-color space, and accept upcoming samples only if the density of samples in the tree is sufficient to assess the new sample's correctness. This algorithm is simple, elegant, memory-efficient, and uses most of the correlation present when rendering an image. It works very well for well converged zones of the image, even for a very low number of samples per pixel. However, for parts where convergence is far from reached – typically parts where indirect illumination dominates –, more samples are delayed, leading to slightly darker zones. Moreover, a k-NN query is required per Monte-Carlo rendered sample, which can lead to an important computational overhead.

Although targeting similar final goals and using the same base toolkit – density estimation –, our approach has major differences with the method by DeCoro *et al.*. From a mathematical toolkit point-of-view, we use kernel-based density estimation on a 1D samples set, while they use k-NN queries on a 5D joint image-color space. From a resultant estimator point-of-view, they define a set of biased correlated average values estimators that are robust to outliers, while we define a biased average estimator that is robust to outliers, and use one such estimator per-pixel. Our estimator can therefore be used as a direct replacement for any average estimator – at the cost of introducing bias –, and can therefore target a wider range of applications.

2.3 Outlier Detection

Outlier detection is a well studied approach in the statistical analysis domain. According to the survey by Hodge *et al.*[6], outlier detection methods can be split into three different categories, depending on the prior knowledge required on the data:

1. Methods requiring tagged data, of the form normal/abnormal, in order to build a model and then classify candidate data [19]. It is impractical in rendering, as the user should tag enough samples per pixel before any automatic processing can be performed.

2. Methods requiring normal data tagged, and figure out abnormality [2]. For the same reason as above, no manual tagging must take place in the algorithm.

3. Methods that do not require any prior knowledge of the data [11]. As no user tagging is required, this category is well suited for rendering.

Methods for outlier detection in a rendering context must fit in the third category, as we do not want to put any constraints on the algorithms used to compute each sample's value. Such existing methods presented in [6] target static distributions, where all the samples are available at once. As we want to be compatible with adaptive sampling, the number of samples can not be fixed in advance. The methods presented in the survey could be adapted by first building a model using a fixed number of samples, and then classifying the other samples without updating the model. As in rendering, each pixel is computed using very few samples (tens or hundreds of samples, compared to millions or billions for database applications), building a fix model can lead to a strong lack of robustness, we thus need a progressive method.

3 Bright-Spot Removal Using Density Estimation

The insertion of our method in the Monte-Carlo rendering pipeline only requires the addition of a single step lying between the integration and the splatting of the sample on the final image (Figure 2).

The integration method computes many radiance samples r_i for a given screen position. Ideally, outliers could be detected by estimating the probability density function (pdf) of the samples that contribute to each pixel. The samples with probability density lower

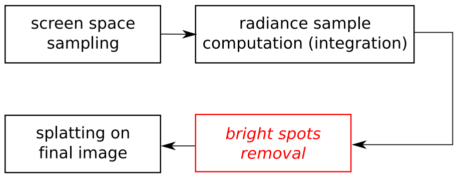

Figure 2: Our method integrates seamlessly as a supplemental stage (in italic red) in a Monte-Carlo rendering pipeline.

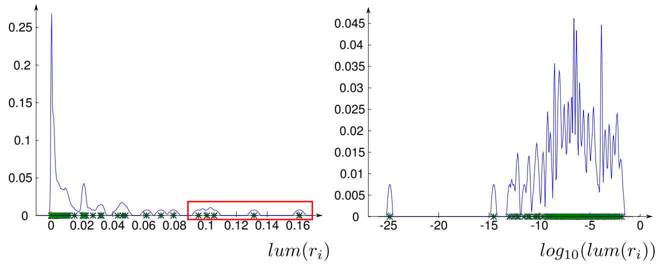

Figure 3: Left: density estimation (blue curve) based on the luminance of samples obtained by path-tracing (green crosses on the horizontal axis). Right: density estimation based on log-luminance. Note that the surrounded isolated values in the left image seem to have contributed to the same light-transport mechanisms, because they have similar orders of magnitude. They should therefore belong to the same mode. It is the case only when using log-luminance.

than a given threshold could then be ignored, as their $p(v)$ value is low and can therefore lead to bright-spots. However, accurately estimating a pdf from a set of samples usually requires a large number of samples, typically several thousands or even millions. As typically only tens or hundreds of samples are available per-pixel, this solution can not be used.

Instead, we cluster the samples in groups, delaying samples which are not part of any group. This clustering uses density estimation on scalar values l_i, each being obtained from a radiance sample r_i. We obtain these l_i values by using the underlying structure of the radiance samples. In the case of path-tracing, each of the radiance samples is the sum of contributions from different light-transport mechanisms: direct-lighting, first-bounce indirect lighting, *etc.*. Each combination of these mechanisms has typical values that have different orders of magnitude. We can therefore differentiate the samples with respect to their contributing mechanisms by using the logarithm of the luminance of these samples (Figure 3).

Our algorithm detects outliers by finding the l_i values that are uncommon. To find them, we cluster the available values in groups, that we call *modes*. We define a mode as the biggest log-luminance interval $[a,b]$ in which the estimated pdf of the distribution defined by the l_i values is strictly positive, as illustrated in Figure 4. If a mode contains several l_i values, it is likely that the associated r_i are viable. We call such a mode an *extended mode*. Otherwise, if a mode contains only a single element, we can not conclude on the viability of the associated r_i sample. Such a mode is called *single*. As bright spots are caused by samples with very large luminance values, we only focus on dubious samples whose log-luminance is greater than the upper bound of the extended mode with largest values. This extended mode is called *last extended mode* from now on, illustrating that it is the last extended mode along the l_i axis. From this observation, our characterization of a dubious sample is: *a sample should not be added to the final image if its associated l_i value generates a single mode greater than the last extended mode*, or

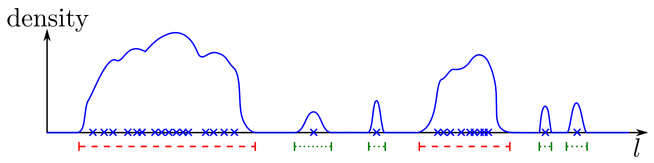

Figure 4: Modes are the biggest intervals of log-luminance values where the density is strictly positive. Red dashed intervals are extended modes. Green dotted intervals are single modes. The x-axis corresponds to the l values, the blue crosses representing the l_i values of the samples. The y-axis corresponds to the value of density obtained using kernel density estimation based on the l_i values. The blue curve is the density obtained. The last extended mode is the right-most extended mode.

equivalently, if its l_i value is greater than the upper bound of the last extended mode, m.

Algorithm 1 presents a basic algorithm performing outlier filtering using our method. Note that it adds previously rejected samples that are finally classified as viable when the acceptance value m is updated, making our method computationally efficient.

Algorithm 1 Basic algorithm for kernel-density-estimation-based outlier rejection

```
Initialize samples log-luminance distribution $S = \{\}$
Initialize dubious list $L = \{\}$
for each rendered sample $r$, with log-luminance $l$ do
    add $l$ to $S$, updating or adding modes
    if the last extended mode has changed then
        recompute $m$, the upper bound of the last extended mode
        for each element $s$ of $L$, with log-luminance $y$ do
            if $y < m$ then
                splat $s$
                remove $s$ from $L$
            end if
        end for
    end if
    if $l < m$ then
        splat $r$
    else
        add $r$ to $L$
    end if
end for
```

As we need to know where the estimated pdf is strictly positive, our method requires to estimate the pdf of the distribution based on its samples l_i. The low number of samples we have is sufficient to retrieve this information, as it does not require a very accurate pdf estimation. Kernel density estimation [13] is very well designed to build a pdf from sample values. From a set of samples $x_1,\ldots,x_n$, the estimated pdf $\overline{p}(x)$ at x is computed based on a per-sample bandwidth h_i:

$$\overline{p}(x) = \frac{1}{n}\sum_{i=1}^{n} K(x - x_i, h_i) \qquad (3)$$

where K is a normalized kernel, in our case Epanechnikov kernel [5].

3.1 Adaptive Bandwidth

The most difficult task in kernel density estimation is estimating each kernel bandwidth, so that the reconstructed density is as close as possible to the original. On the one hand, the choice of a too low bandwidth leads to a very jaggy density, with many spikes that are not present in the actual distribution. In terms of modes, many modes will be single, while they should not. On the other hand, a too large kernel bandwidth leads to a lot of smoothing, putting all the samples in one extended mode. For a better reconstruction, it is advised to compute a per-sample kernel bandwidth [13]. From a reasonable base bandwidth h, an adequate reconstruction can be obtained using an adaptive bandwidth h_i for each log-luminance sample l_i:

$$h_i = nn \times \left(\frac{\exp\left\{ \frac{1}{n} \sum_{j=1}^{n} log\overline{p}(l_j) \right\}}{\overline{p}(l_i)} \right)^{1/2}, \qquad (4)$$

where nn is the average distance to the nearest neighbor of each sample, and $\overline{p}(l)$ is the density at a point l estimated using nn as bandwidth for all samples (Equation (3)).

4 Lowering Memory Consumption

Computing adaptive bandwidths requires us to store all the samples, in order to be able to compute the nn value and the final bandwidth at each sample. Although this storage can be done when computing one pixel after another, all the samples cannot be stored in a more generic rendering context, when using image-based adaptive sampling for instance. As we also want to be able to deal with these situations, we develop a more specific method based on Algorithm 1 to limit memory consumptions.

4.1 Algorithm Overview

Figure 5 presents the general procedure used in our method for each pixel. Two main phases are performed: a pure learning-phase, which basically computes a base m value, and then normal processing, which updates it.

The pure learning phase just consists in storing the first N r_i values, N being fixed by the user. For non-black samples, the logarithm of the luminance l_i is also computed. After the N-th sample has been stored, modes are built using a specific representation, based on the N l_i values (Sections 4.2 and 4.4). m is computed, and each of the N samples is tested: if its l_i value is below m, the associated sample is splatted, otherwise it is put in the dubious list. Once this is done, the algorithm switches to normal processing.

In the normal processing phase, each time a new radiance sample r_i is computed, the modes representation is updated to take into account the sample's log-luminance value l_i. This update can have three different consequences on the representation: creation of a new single mode, modification of an existing extended mode, or creation of a new extended mode from an existing single mode. In the two last cases, the maximal value m is updated if the last extended mode has been modified or replaced. If m has been updated, all the samples in the dubious list are tested, as they may be below the new m value. This makes our method computationally efficient, as samples are not permanently rejected, but just delayed. After this update phase, the new sample is accepted if its l_i value is below m. If l_i is above m, r_i is added to the dubious list. Note that the dubious list size can be bounded.

4.2 Approximate Distribution Representation (ADR)

As outliers are samples that are far from the viable ones, having an exact value for the base bandwidth nn is often not necessary. To avoid consuming too much memory, we develop a compact approximate representation of the distribution, which keeps an exact representation of the single modes, while approximating the extended modes with only the lower and larger samples that belong to it. This representation allows us to compute a close approximation of nn, while drastically reducing the memory consumption. As a matter of fact, a large majority of the samples belonging to a pixel are in extended modes, as these modes represent log-luminance intervals where samples are the most probable to lie.

In an extended mode, the precise position of each sample is not useful in order to get a correct approximate value of the nn base bandwidth. Instead, we approximate the position of the M samples of an extended mode by M regularly spaced artificial samples. Therefore, for each extended mode, we just need the two extrema samples, and the number of samples in the mode.

Initial learning: It consists in storing a user-defined number N of samples, and then building an approximate distribution from these samples. Once the N samples have been stored, the list is sorted by increasing log-luminance values, the bandwitdh of each sample is computed according to Equation (4) and each sample l_i is browsed sequentially to build modes by aggregation: when the kernels of two consecutive samples overlap, they belong to the same mode. Each time a mode is detected, its extrema samples and number of samples are stored. Once the modes are built, the maximum acceptable value m is set to the log-luminance of the maximum sample of the last extended mode.

Incremental update: Once the base modes have been computed, the representation is updated for each new radiance sample r, with a log-luminance value equal to l. If l is included in the interval of an extended mode, this mode's number of samples is just increased by 1. Otherwise, a single mode centered at l is created and added to the representation, and we compute its potential overlap with existing modes. If an overlap is found, it leads to either extending the bounds of an existing extended mode, or creating a new extended mode from two overlapping single modes. When l does not belong to any mode, its kernel bandwidth is computed using the samples of the approximated distribution, and overlap between this kernel and the kernel of each sample of the approximated distribution. Note that merging of modes can occur if the kernel of l overlaps both the mode before it and the mode after it. In this case, a new extended mode replaces the two existing modes.

4.3 ADR : Analysis

Robustness: The third row of Figure 8 shows the results obtained when we removed bright spots using our approximate distribution representation (ADR). We have tested our method on several scenes, using path-tracing and 50 samples for the initial learning phase. All these scenes exhibit many bright spots when rendered using path-tracing, which makes them relevant to evaluate our method. Each pixel of each image can be considered as an independent test, as each pixel has its own independent estimator. Our method successfully delays most of the samples leading to bright spots without delaying the viable ones, consequently improving the image quality.

Progressiveness is an important property of our method, as it allows us to splat samples only when they are viable. We have tested the effectiveness of this property on a glossy caustic. The second column of Figure 9 presents the results obtained by ADR when adaptive sampling is used. It shows that ADR first delays many of the caustic samples, but finally updates its maximum value as they are considered correct, leading to an adequat caustic. Note that even for the very difficult case of the glossy caustic on the glossy part of the floor, it correctly separates caustic samples from bright spots.

Discussion: The approximation introduced in this mode representation allows us to use our density-estimation-based method without having to store all the samples. It is both robust and easy to parameterize (using 50 as initial learning size has been proved efficient on all our tests), and introduces a small rendering time overhead. However, as illustrated by Table 1, this representation can lead to the storage of a large number of modes, and so, a large memory cost, even if it is drastically reduced compared to a brute-force approach. This is due to the necessity of using all modes in the computation of the average nearest-neighbor base bandwidth nn, and the adaptive bandwidth.

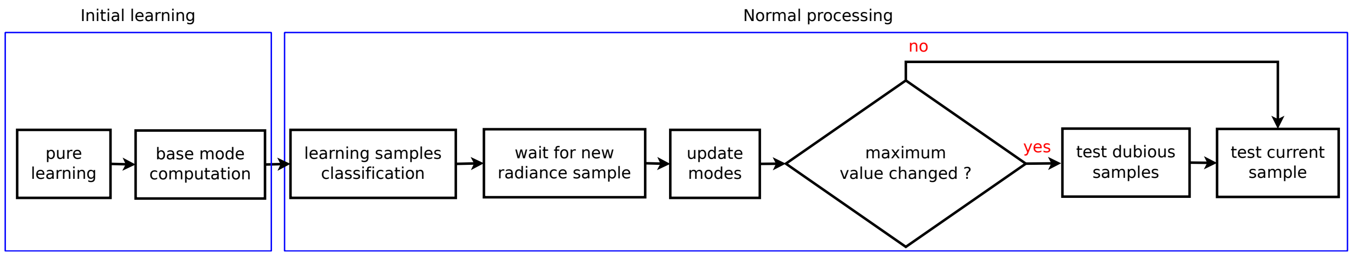

Figure 5: The general algorithm used in our bright spot removal method.

	ring		living room		computer room	
	sing.	ext.	sing.	ext.	sing.	ext.
100	2.3M	4.0M	1.7M	3.2M	568K	1.2M
HQ	3.1M	4.7M	4.0M	5.2M	1.8M	1.8M

Table 1: Total number of single and extended modes for various scenes when using ADR, for images computed with 100 samples per pixel and high quality ones (HQ), obtained using adaptive sampling and between 200 and 1000 samples per pixel. The ring images have been computed at a resolution of 800×600, the others at a resolution of 800×450 pixels.

4.4 Rule-Of-Thumb Bandwidth (ROT)

To further reduce the memory consumption of the ADR representation, we propose an alternative method to compute each kernel bandwidth, which does not require us to store all the modes. This method, although less robust, can still lead to very good results when the outliers are easy to identify. In this case, our density estimation method performs efficiently even with very conservative kernel bandwidths. Therefore, we choose to use a constant bandwidth instead of the adaptive one used in ADR. This bandwidth is computed using the statistical properties of the underlying distribution. Such a bandwidth called *rule-of-thumb* (ROT) has been introduced by Deheuvels [4, 15]. For Epanechnikov kernels, the ROT bandwitdh is obtained as:

$$h_{rot} = (25 \times n)^{-1/5} \times \sigma, \qquad (5)$$

where n is the number of samples in the distribution, and σ their standard deviation. Using this bandwidth, we just need to store the log-luminance of the largest sample of the last extended mode, and the larger single modes. All the modes located before the last extended mode are no longer needed, as any sample that is below the last extended mode is automatically accepted.

Initial learning: Initial learning for this representation is very similar to the one of the ADR method. The only essential differences are during the base modes computation: there is no adaptive bandwidth to compute, and when a new extended mode is built, all the modes that have lower upper bounds are released, as they are not used anymore.

Incremental update: When a new sample is added, if it is lower than the current maximum acceptable value, nothing has to be updated. When it is larger, overlap is examined using the constant bandwidth given by Equation (5). If merging occurs and a new extended mode is created with values larger than the last extended mode, this new extended mode becomes the last one, and the lower modes are released. Otherwise, a new single mode is added.

4.5 ROT : Analysis

We perform the same tests as in Section 4.3.

Robustness: the results, presented in the fourth row of Figure 8 show that this method, although a little less robust than ADR, succeeds in identifying the outliers in most cases.

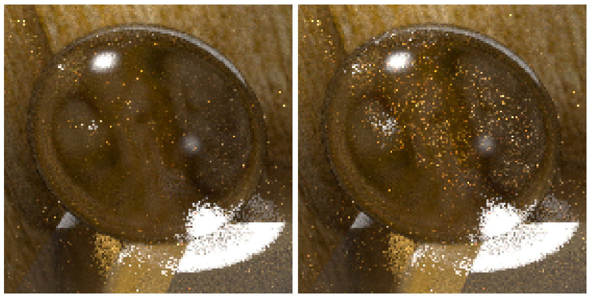

Figure 6: Close-up on the glass sphere of the ring scene, where the ADR method (left) performs better than the ROT one (right).

	ring		living room		computer room	
	sing.	ext.	sing.	ext.	sing.	ext.
100	198K	482K	184K	360K	102K	331K
HQ	206K	482K	200K	360K	99K	331K

Table 2: Total number of single and extended modes for the various scenes using ROT. Note that the number of extended modes corresponds to the number of sampled pixels.

Progressiveness: the same progressiveness test as for ADR (Section 4.3) has been performed. Figure 9 shows that similarly to the ADR method, the ROT method correctly found that caustic samples are acceptable, while still delaying samples producing bright spots.

Discussion: the results provided by this method are not as good as those provided by ADR (Figure 6). However, for most pixels, results are either identical or very similar, with a tendency for ROT to accept more samples than ADR. This comes from the constant bandwidth computed using ROT that is more conservative than the adaptive bandwidths computed with ADR, thus leading to larger modes.

4.6 Switching From ADR to ROT

ADR is robust but can consume a lot of memory, while ROT works well for most cases, with a lower memory consumption and lower computational cost. However, it performs poorly in more intricate cases. We therefore develop a method that allows us to use, per-pixel, either one or the other, depending on the difficulty to differentiate outliers.

Both the ROT and the ADR methods store the last single samples after their last extended mode. More precisely, as ROT is more conservative, it is likely that the larger single modes of ROT are included in the larger single modes of ADR. These modes can therefore be shared by both methods. Moreover, the computational cost of ROT is negligible. Thus, in order to reduce memory consumption, we want to use ROT whenever possible. We introduce two user parameters, δ and T, which allow us to control the transition

	ADR only		ROT only		$T=1,\delta=0$		$T=1,\delta=0.5$		$T=2,\delta=0$		$T=2,\delta=0.5$	
	sing.	ext.	sing.	ext.	sing.	ext.	sing.	ext.	sing.	ext.	sing.	ext.
100	2.2M	3.8M	190K	433K	710K	1.4M	421K	872K	1.8M	3.2M	1.3M	2.3M
HQ	6.6M	8.7M	316K	483K	1.3M	1.9M	709K	986K	4.8M	6.5M	2.6M	3.3M

Table 3: Left: number of modes when using only ADR or only ROT on the ring scene. Right: number of modes when using our hybrid method, in function of T and δ. As expected, fewer modes are present when the parameters favor ROT over ADR.

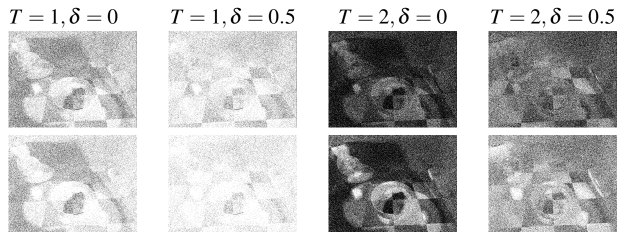

Figure 7: Top: in gray, pixels for which only ROT is used on the ring scene, after 100 samples per pixel have been computed. Bottom: the same after the computation of 500 samples per pixel. Note that ROT does not tend to be used everywhere, but rather in regions.

from ADR to ROT. At the beginning, both methods are maintained, using ADR's maximum acceptable value to test the samples. For each sample leading to an update of the maximum acceptable value of one of the two methods, we compute the relative distance between the maximum of ROT m_{rot} and the maximum of ADR m_{adr}, defined as $|m_{rot} - m_{adr}|/m_{adr}$. If T such successive distances are less or equal to δ, ROT becomes the only method used for a pixel, and the data used for ADR is released.

These two parameters allow the user to specify a measure of similarity between the results. If the two methods give sufficiently close maximum values (δ) during a sufficiently long time (T) for a pixel, then ROT can safely be used alone, leading to a greatly reduced memory consumption for this pixel as the data for ADR is no longer needed. Experimentally, requiring equal maximum values ($\delta = 0$) during two successive steps ($T = 2$) proved a good compromise between memory consumption and robustness for all our test scenes.

Figure 7 shows where ROT is used as the only method in the ring scene, for various values of the parameters T and δ. The use of ROT has been observed after 100 samples per pixel have been computed, and after uniform over-sampling has been performed. These parameters effectively allow the user to choose between precision and memory, by defining the amount of similarity desired between the two methods, and verifying this similarity over a small or large time interval before using ROT only. As shown by Table 3 and as expected, the more ROT is used, the lower the memory consumption. The column labeled ADR+ROT of Figure 8 presents the results obtained when using both methods as indicated here, with $T = 2$ and $\delta = 0$. These images have been computed using the same radiance samples as those used when testing ADR and ROT.

5 Results

All the results shown below have been computed with an initial learning set size of 50 samples.

Robustness: Figure 8 shows the results obtained using ADR, ROT, and both at the same time with parameters $T = 2$ and $\delta = 0$, on three different scenes. We compare it to the image obtained using the standard average estimator, and the one obtained by the method from DeCoro *et al.*(called DBOR in the remaining). These images have been shot after 100 samples per pixel have been computed. Note that for each scene, the exact same radiance samples were used to compute each image. Each pixel of each image can be considered as an independent test of our method, as we define one independent estimator per-pixel. Each pixel of the images produced by DBOR can be considered as a reference, as their method employs virtually a much larger number of samples for their classification, through the use of the joint-space: for each sample, they potentially use all the spatially nearby samples. We can see that results of all the estimators obtained using our method are really close to the ones from DBOR, demonstrating the robustness of our method, even though less samples are used for classification: for each sample, only the samples that contribute to the same pixel are used. Very few pixels are still bright spots, but our method removes less samples than DBOR in dim zones (which leads to slightly darker results for DBOR), or in large variance zones such as caustics. This confirms that our method can be used as a drop-in average estimator replacement even when the number of samples is relatively low. Table 4 confirms that the average number of samples delayed by our method is kept small, smaller than the one of DBOR while still removing most of the bright spots.

Progressiveness: as shown by Figure 9, progressiveness has been tested in a case where the sample distribution is hard to handle when using path-tracing: a glossy caustic on a glossy floor. As the glossy caustic is rarely sampled and its samples have a quite large value, it can be mistaken for outliers. Our method first considers these samples as outliers, but as more samples are computed, it detects they are viable, while still delaying high-value samples which are not part of the caustic.

scene	ADR + ROT	DBOR
ring	1.3%	4.0%
living room	1.4%	2.3%
computer room	0.5%	1.1%

Table 4: Percentage of samples delayed in average by the ADR + ROT method and the one by DeCoro *et al.*, after 100 samples per pixel have been computed.

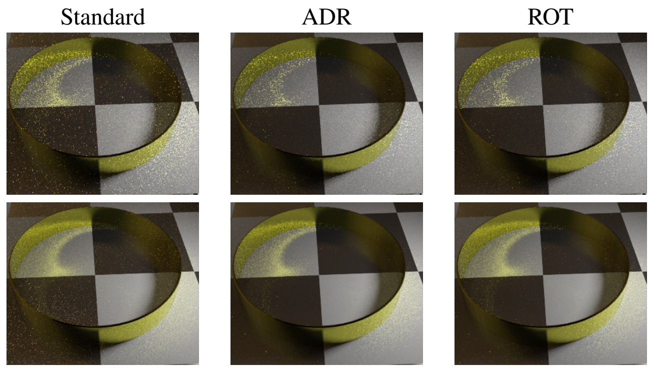

Figure 9: Top: images obtained with 100 samples per pixel. Bottom: images obtained with 1000 samples per pixel. Note that a large number of very visible bright spots are still visible on the brute path-tracing image.

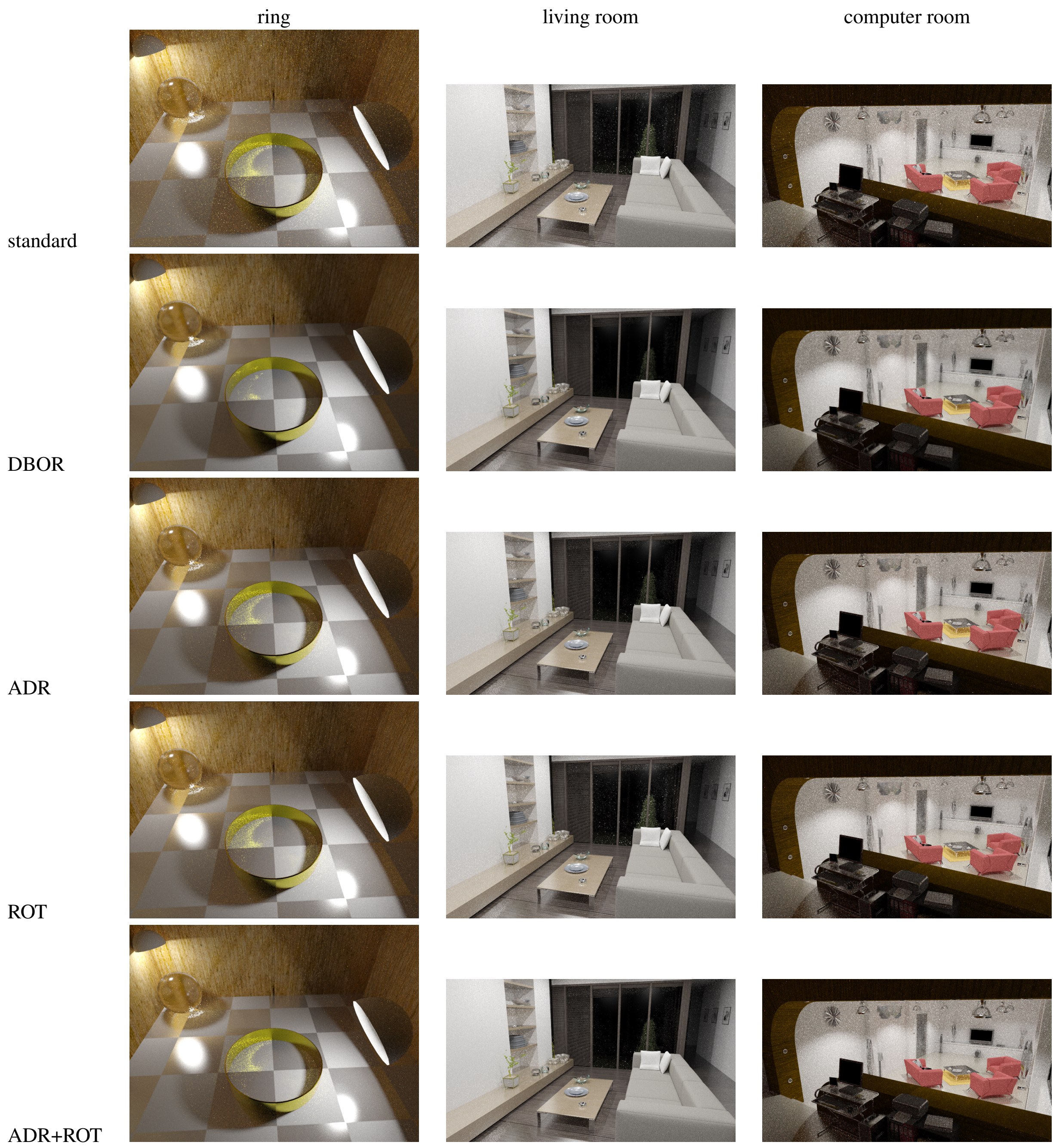

Figure 8: Images obtained using different average estimators to compute each pixel's final value from 100 samples obtained using path-tracing. First row: standard average estimator. Second row: average estimator as defined by the DeCoro *et al.* method, which uses inter-pixel correlation. Third, fourth and fifth rows: estimators obtained using one or both of the approximations we developed to lower the memory consumption. (See additional material for full size pictures.)

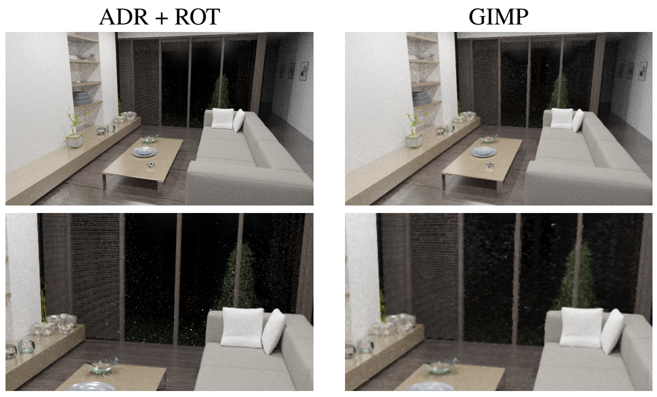

Figure 10: Left: image obtained using our method. Right: image obtained by applying the GIMP despeckle filter on the image obtained using the standard average estimator. Top: full image, bottom: close-up. Note the amount of blur and the deformations at the edges of the objects added by the GIMP filter.

Performance: the representations we use induce a small overhead in computation time (from 5 to 10 percent for the measurements we have done). Note that it is independent from the scene complexity as we only rely on the values returned by the integration algorithm, not on the scene itself. For a bright-spot-removal application, it is negligible compared to the time that would be required to remove the bright spots by pure over-sampling (to avoid blurring when filtering), and remains lower than the overhead caused by DBOR, which suffers from poor parallelization efficiency when used on a many-core architecture. As a matter of fact, the accesses to the samples tree have to be thread-safe. Our method does not suffer as much from thread-safety, as the accesses to different pixels representation can be made in parallel.

Comparison with image-space methods: specific image-space methods have been developped to handle bright-spots. Figure 10 shows that even if removing the bright spots, they still introduce a lot of artefacts in the resulting images, such as blur or deformations of objects.

6 Conclusion

We use kernel-density estimation to define an average estimator robust to outliers. As the direct use of this method to remove bright-spots in Monte-Carlo-based rendering requires too much memory, we develop two specific approximate representations, namely ADR and ROT. Our sample level approach delays samples which are considered as outliers. Applied to bright-spot removal, it allows us to anticipate and prevent the appearance of bright spots, without introducing neither blur nor smears, which are typical artefacts introduced by image-space denoising methods. Our method is based on a proximity-based characterization – the modes – to robustly separate viable samples from dubious ones even with a low number of samples. This characterization is implemented using a kernel density estimation method. The base method and the representations we develop are progressive to allow delayed samples to be finally splatted on the final image, thus making our algorithm computationally efficient.

When using one of the two representations we develop for bright-spot removal, our algorithm is only parameterized by the number of samples to use during the initial learning phase. The lower this parameter, the faster the first image obtained as learning is shorter, but results might be less accurate. Our tests, performed on various scenes, show that setting this parameter to 50 leads to good results. This shows that this parameter is in fact almost scene-independent, meaning that our algorithm is in practice parameter-less and can be directly applied on any new scene using $N = 50$. We also show how to use both representations at the same time to further reduce memory consumption without sacrifing robustness, switching from ADR to ROT when results are sufficiently close during a sufficient amount of time. Two parameters T and δ allow the user to control the robustness/memory consumption ratio. We compare the results we obtain to a method that uses the inter-pixel correlation to obtain very robusts results, and show that our method, which can replace any average-value estimator, leads to very close results. This allows us to test our method on a very large number of sample sets (one per pixel), and assess its robustness even when the number of samples is kept relatively low.

References

[1] H. Dammertz, D. Sewtz, J. Hanika, and H. Lensch. Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In *Proceedings of HPG 2010*, pages 67–75, 2010.

[2] D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of The International Conference on Intelligent Systems*, 1995.

[3] C. DeCoro, T. Weyrich, and S. Rusinkiewicz. Density-based outlier rejection in Monte Carlo rendering. In *Pacific Graphics*, volume 29, Sept. 2010.

[4] P. Deheuvels. Estimation non paramétrique de la densité par histogrammes généralisés (in french). *Revue de Statistique Appliquée*, 25:5–42, 1977.

[5] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability and its Applications*, 14(1):153–158, 1969.

[6] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, 2004.

[7] H. Jensen. Global illumination using photon maps. In *EGWR '96*, pages 21–30, 1996.

[8] J. T. Kajiya. The rendering equation. In *SIGGRAPH '86*, pages 143–150, 1986.

[9] E. P. Lafortune and Y. D. Willems. Bi-directional path tracing. In *Compugraphics '93*, pages 145–153, 1993.

[10] M. D. McCool. Anisotropic diffusion for monte-carlo noise reduction. *ACM Trans. Graph.*, 18(2):171–194, 1999.

[11] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427–438, 2000.

[12] H. E. Rushmeier and G. J. Ward. Energy preserving non-linear filters. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 131–138, 1994.

[13] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. 1986.

[14] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the 1998 IEEE International Conference on Computer Vision*, pages 839–846, 1998.

[15] B. A. Turlach. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*, pages 23–493, 1993.

[16] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD. thesis, Stanford University, 1997.

[17] E. Veach and L. J. Guibas. Bidirectional estimators for light transport. In *EGWR '94*, pages 147–162, 1994.

[18] E. Veach and L. J. Guibas. Metropolis light transport. In *SIGGRAPH '97*, pages 65–76, 1997.

[19] D. Wettschereck. *A study of distance-based machine learning algorithms*. PhD thesis, 1994. Adviser-Dietterich, Thomas G.

[20] R. Xu and S. N. Pattanaik. A novel monte carlo noise reduction operator. *IEEE Comput. Graph. Appl.*, 25(2):31–35, 2005.