Vis Comput (2013) 29:707–716
DOI 10.1007/s00371-013-0807-3

ORIGINAL ARTICLE

Probabilistic illumination-aware filtering for Monte Carlo rendering

Ian C. Doidge · Mark W. Jones

Published online: 26 April 2013

Abstract Noise removal for Monte Carlo global illumination rendering is a well known problem, and has seen significant attention from image-based filtering methods. However, many state of the art methods breakdown in the presence of high frequency features, complex lighting and materials. In this work we present a probabilistic image based noise removal and irradiance filtering framework that preserves this high frequency detail such as hard shadows and glossy reflections, and imposes no restrictions on the characteristics of the light transport or materials. We maintain per-pixel clusters of the path traced samples and, using statistics from these clusters, derive an illumination aware filtering scheme based on the discrete Poisson probability distribution. Furthermore, we filter the incident radiance of the samples, allowing us to preserve and filter across high frequency and complex textures without limiting the effectiveness of the filter.

Keywords Global illumination · Monte Carlo · Path tracing · Noise reduction · Poisson probability distribution

1 Introduction

Monte Carlo integration methods for global illumination have shown their elegance and generality when handling physically based light transport. They are able to accurately synthesise a wide range of optical effects and surface materials while maintaining relative simplicity. The cost of this generality is their poor convergence, often requiring thousands of samples per pixel to reduce noise to desirable levels. Techniques for combating noise such as importance sampling and low discrepancy distributions are beneficial, but still provide slow convergence and spiked noise. To accurately estimate the measured radiance through a pixel in a path based Monte Carlo renderer, we must evaluate the radiance from all surface points visible through that pixel:

$$P_{(i,j)} = \int_S \int_\Omega L_e(x,\boldsymbol{\omega}) + L_r(x,\boldsymbol{\omega})\,\mathrm{d}x\,\mathrm{d}\boldsymbol{\omega} \tag{1}$$

where L_e and L_r are the emission from and radiance arriving at a point on a surface x from the set of all visible surface positions $x \in S$ and scattered in the direction of the camera $\boldsymbol{\omega}$. To estimate L_r, we must further integrate over the hemisphere above x to account for the radiance arriving from all directions $\boldsymbol{\omega}'$:

$$L_r(x,\boldsymbol{\omega}) = \int_\Omega L_i\big(x,\boldsymbol{\omega}'\big) f_r\big(x,\boldsymbol{\omega},\boldsymbol{\omega}'\big)\big(\mathbf{n}\cdot\boldsymbol{\omega}'\big)\,\mathrm{d}\boldsymbol{\omega}' \tag{2}$$

where L_i is the radiance arriving at the surface from $\boldsymbol{\omega}'$, f_r and $\mathbf{n}$ are the bidirectional scattering distribution function (BSDF) and shading normal at x, respectively. Filtering techniques rely on pixel coherence, providing low bias noise removal in regions where pixel integrals exhibit similarity. For regions where pixel coherence breaks down, low variance sources such as geometry and texture buffers are often employed to help preserve edges. Such methods have been popular due to their fast performance, but often sacrifice quality, resulting in filtering artefacts, loss of high frequency features, or limitations to certain light transport; wherein lies the strength of Monte Carlo methods.

I.C. Doidge (✉) · M.W. Jones
Department of Computer Science, Swansea University, Swansea, SA2 8PP, UK
e-mail: csiand@swansea.ac.uk

M.W. Jones
e-mail: m.w.jones@swansea.ac.uk

In this work we present a novel framework for the dynamic storage and filtering of Monte Carlo samples that effectively removes noise whilst preserving high frequency features in both texture and incident illumination. We perform per-pixel clustering over the incoming radiance of each path contribution, separated by their path length. This results in a clustering framework of multiple layers that together represent the pixel integral. Treating path tracing as a Poisson process, these clusters allow us to compare the frequency of sample occurrences in a pixel with the occurrence of similar luminance samples in its neighbourhood. From this we gain two advantages. First, we identify and temporarily remove high energy noise in the irradiance, which is otherwise difficult to distribute across the image often leading to splotchy artefacts. Second, we compare clusters of similar luminance across an image-space filter kernel, allowing us to derive filter weights based on similarities in irradiance. Combined with geometric edge detection from depth and normal buffers, we reduce bias across edges from high and low variance sources. Furthermore, complex texture detail is preserved since we filter the incoming radiance, allowing us to maximise kernel bandwidth and pixel correlation across regions with complex textures. The final pixel radiance is reconstructed from our filtered irradiance using texture data from the rendering pass. Our method effectively filters noise from the incident illumination, whilst preserving hard gradients and edges that are otherwise difficult to identify, providing important visual cues and realism for complex materials and light transport. The main contributions of our paper are therefore:

- Novel layer based clustering, which accurately represents changes in incident illumination.
- A probabilistic, discontinuity aware image-space filtering algorithm that is sensitive to illumination from multiple overlapping sources.
- Image based irradiance filtering, enabling the preservation of complex texture detail without the need to store BRDFs.
- To our knowledge, the introduction of the Poisson probability distribution to noise removal, and a framework that is reliable at low sample counts.
- High intensity noise removal, tunable post-render and suitable in image-based filtering.

2 Related work

Adaptive sampling Adaptive rendering techniques that concentrate samples in high variance regions have shown their effectiveness. Hachisuka et al. [6] introduced a multi-dimensional adaptive sampling technique, estimating error in path space, catering for arbitrary lens effects. However their sample storage does not scale well to suit global illumination and sample reconstruction can be costly. Rousselle et al. [12] combine image-based adaptive sampling with filtering, producing good results and a promising avenue for future methods. Many of these adaptive methods can perform poorly for difficult lighting conditions and high frequency features, reducing efficiency. Path space adaptive methods based on Metropolis sampling and Markov Chain mutations [1, 16] are more effective and generally improve convergence, but still suffer from noise early on and in under sampled areas.

High intensity noise removal DeCoro et al. [3] employ a sample rejection technique based on the local density in a 5D joint image-colour space, delaying the addition of *outlying* samples that do not conform to the current neighbourhood. Despite effectively removing high intensity noise in well converged areas, for regions with sparse samples it can have detrimental effects due its harsh rejection properties, resulting in significant energy loss and removal of illumination clues. It also does not consider the relative luminance of samples in the same pixel, removing isolated samples that may not be responsible for noise due to the presence of those with higher energy. Pajot et al. [10] also present density estimation to tackle bright spot removal, based on per-pixel 1D luminance distributions of the observed samples to improve performance. Conversely to DeCoro et al., neighbourhood similarities are not accounted for, leading to the rejection of samples representing important illumination features, but are outliers with respect to a single pixel distribution. Both approaches are unsuitable at lower sample counts, and as a pre-process for image based filtering, the focus of our work. Per-pixel luminance-based clustering is an approach we also employ, supplemented by additional data and with the premise of identifying and filtering illumination discontinuities. As Sect. 4.2 will show, our framework is also suited to high intensity noise removal though we use 1D per-pixel distributions and image space neighbourhood density allowing more robust filtering, adjustable during rendering to eliminate remaining noise that can be left behind by existing methods.

Sample-space reconstruction Recent methods such as [9, 14] rely on identifying statistical relationships between small numbers of Monte Carlo samples, to remove noise. Whilst providing impressive results and handling large dimensional problems, they do not scale well with increased input samples which are often required to capture more difficult lighting phenomenon.

Irradiance filtering Ward et al. [17] introduced irradiance caching, storing a sparse set of samples at surface locations and interpolating between neighbours to reconstruct the irradiance at a surface point. Kontkanen et al. [7] use image-based irradiance filtering to reduce noise, but rely on high numbers of samples and additional costly 'feeler' rays to detect local geometry. Radiance caching and filtering algorithms have also been proposed [5, 8]. However, all these algorithms either rely on low frequency illumination or BRDFs, or are not applicable to general light transport. Additionally, we build upon unbiased Monte Carlo methods, allowing us to trivially revert to the unbiased solution, unlike cache based methods.

Image-based filtering Suykens and Willems [15] use per-sample adaptive kernel widths to distribute radiance across pixels during progressive rendering. This transforms noise from higher to lower frequencies which are less objectionable, but does not consider geometric or texture edges resulting in blurry features. Dammertz et al. [2] successfully modify the à-trous wavelet transform, adding filter weights derived from geometry buffers and input samples to filter global illumination in real time. Despite largely respecting both geometric and illumination edges, the à-trous filter can lose texture detail, and present noticeable filtering artefacts. The cross-bilateral filter [4, 11], makes use of range buffers to preserve strong image features resulting from low variance sources, restricting filtering to regions of high similarity. This has shown to produce good results for Monte Carlo rendering (recently [13]). We use the principle of the cross-bilateral filter in our work, and provide comparisons in our results. Schwenk et al. [13] use a perception based blending operator, relying on pixel variance to combine path traced input samples with a filtered image, producing an unbiased algorithm in the limit. It further allows them to use a higher quality cross-bilateral filter for fast previews. Illumination edges and glossy reflections, however, become blurred due to the ignorance of illumination features, the main problem addressed in our paper. Their blending approach is orthogonal to our filtering technique and is in the spirit of our high quality filtering, making it of interest for future integration with our work. Whilst in this paper we do not focus on interactive or GPU rendering, we address a difficult problem not effectively tackled by many authors; detecting discontinuities in high variance illumination. By using a high quality filter, we can provide biased but low noise previews without tight restrictions on feature frequency or surface characteristics common to many approaches.

3 Poisson distribution

Our edge detection makes use of the Poisson discrete probability distribution, which expresses the probability that a given number of events will occur within a specified interval, provided they occur with a known average rate and are generated independently of the interval since the last event. Assuming we have observed an average of λ events in a fixed interval and that the process or processes generating such events are random with respect to the frequency of occurrence, the probability of observing exactly k events is given by

$$P(X=k)=\frac{\lambda^k e^{-\lambda}}{k!} \tag{3}$$

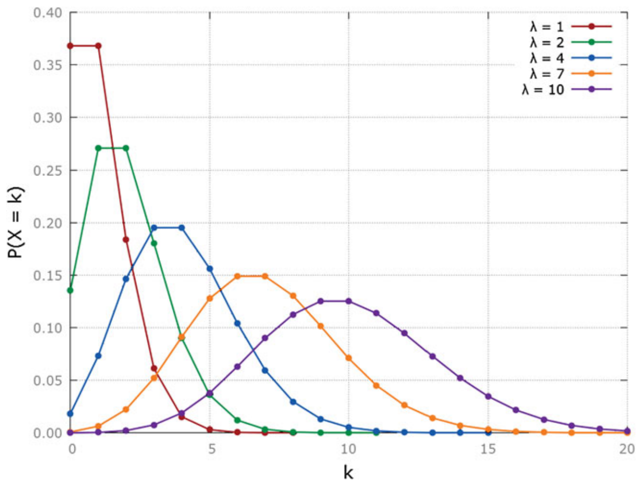


Fig. 1 Poisson distribution showing the probability mass function $P(X=k)$ for various mean number of observed events, λ. Notice as λ increases it more closely resembles the binomial and standard normal distributions

Examples of the Poisson distribution for various mean occurrences are shown in Fig. 1. Notice that the Poisson distribution is a skewed distribution, accounting for the absence of probabilities for negative k. As λ moves away from zero, the Poisson distribution becomes symmetric, following the normal distribution with mean λ and standard deviation $\sqrt{\lambda}$. The Poisson distribution is especially useful when modelling rare events, where a large number of events may be observed as a result of independent processes, but the generation of each event itself is rare, such is the case in Monte Carlo methods. Samples are generated stochastically during rendering, and despite making use of stratification and importance sampling techniques, the generation of a given sample is not influenced by previous samples.

4 Our method

4.1 Rendering and sample clustering

In order to describe our filtering technique, we first detail the organisation of our rendered contributions. We use a standard path tracer with next event estimation, and our method is orthogonal to variance reduction techniques such as importance sampling and Russian Roulette. Our renderer treats the radiance accumulated at the first three path vertices as individual contributions, returning a pair of radiance and irradiance values for each vertex. Although complicating the clustering process, this improves sample separation allowing for easier edge detection during filtering. For low variance pixel integrals this has little impact on filtering performance since we can merge clusters that form a continuum. By separating path contributions, we can filter each layer independently, regardless of the illumination features and noise present in the remaining layers. For each path vertex v, we compute the unbiased radiance contribution R_v:

$$R_v = L_i(x_v, \boldsymbol{\omega}'_v) \prod_{i=1}^{v} f_r(x_i, \boldsymbol{\omega}_i, \boldsymbol{\omega}'_i)(\mathbf{n}_i \cdot \boldsymbol{\omega}'_i) \tag{4}$$

where $L_i(x_v, \boldsymbol{\omega}'_v)$ is the radiance arriving at x_v from the light source in the direction $\boldsymbol{\omega}'_v$ (see Fig. 2). Additionally we obtain our irradiance contribution I_v by scaling $L_i(x_v, \boldsymbol{\omega}'_v)$ by the path throughput from the first non-specular vertex d (where $d \geq 1$) to v:

$$I_v = (\mathbf{n}_d \cdot \boldsymbol{\omega}'_d)$$

$$\times L_i(x_v, \boldsymbol{\omega}'_v) \prod_{i=d+1}^{v} f_r(x_i, \boldsymbol{\omega}_i, \boldsymbol{\omega}'_i)(\mathbf{n}_i \cdot \boldsymbol{\omega}'_i) \tag{5}$$

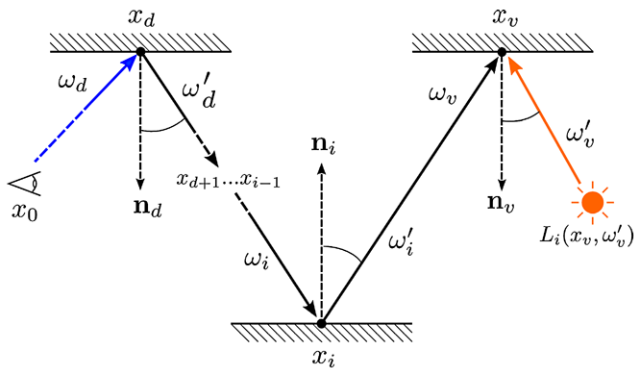


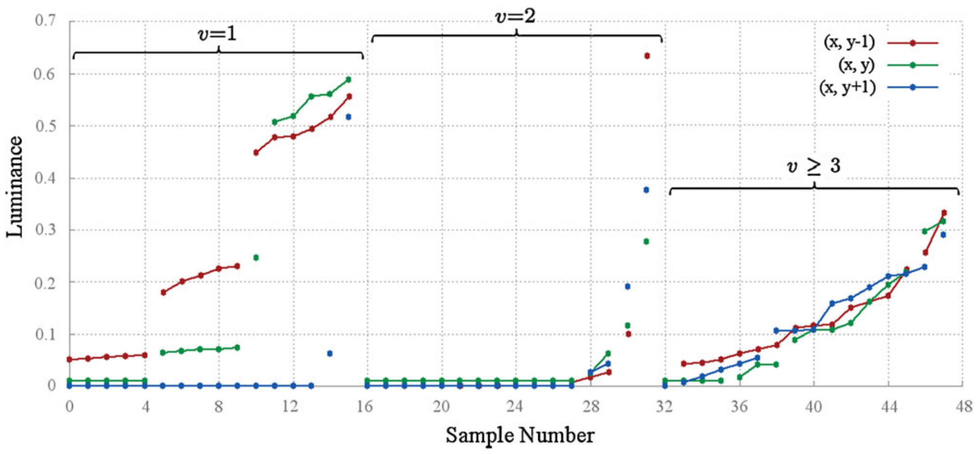
Fig. 2 Geometry illustration for our irradiance contribution calculations in Eqs. (4) and (5). Radiance arriving at each vertex v is scaled by the path throughput from the first diffuse vertex d to v, omitting the BRDF and texture reflectance at x_d

The influence of the BRDF and texture reflectance present at x_d, and the path throughput from x_0 to x_{d-1} are thus ignored in our calculation of I_v. This allows us to preserve high frequency detail for each pixel, whilst sharing the lower frequency irradiance information. The influence of the omitted $f_r(x_d, \boldsymbol{\omega}_d, \boldsymbol{\omega}'_d)$ term and path throughput up to d can be recovered by dividing R_v by I_v; which becomes useful when deriving our final pixel radiance values after filtering (Sect. 5.2). Each cluster stores aggregated statistics for a subset of samples that are contiguous in the 1D luminance domain of a given layer (Fig. 3).

```
ClusterStatistics{
  float[3] radiance
  float[3] irradiance
  float min
  float max
  int numSamples : 31 bits
      outlier    : 1 bit
}
```

Along with R_v and I_v, we also store the number of samples in the cluster and its luminance extents, used to add samples and merge existing clusters during rendering. Clusters for each layer are stored in ascending order of luminance to improve clustering and filtering performance. In order to put an upper bound on our memory usage, we constrain the number of clusters stored for any one pixel layer to eight. Such constraints are adequate for many scenes, since we only need enough clusters to distinguish between samples leading to visible edges. This allows us to merge low luminance groups without reducing the capability of our filter. The sign bit of the sample count indicates whether this cluster has been rejected during our high intensity noise removal.

Fig. 3 Example plot of our layered clustering for a small neighbourhood of pixels in a shadow penumbra (pixel at (x, y) is the cyan point in Fig. 4). The *green* and *red plots* are offset on the y-axis for clarity. Lighting discontinuities form separate clusters, while smooth regions are clustered together. We can see both a highly occluded pixel (*blue plot*), and a pixel lit by all light sources (*red plot*)



We rely upon two heuristics to update and maintain our clusters; adding to existing clusters as samples are rendered (creating new clusters if necessary) and merging clusters to maintain a compact approximation of the integral. We render samples in batches of four per pixel to reduce cluster operations while maintaining short frame updates. Cluster merging allows us to refine our model in regions of the integrand with rapid luminance changes, where discontinuities are likely, and reducing storage costs for smooth regions. From Fig. 3 we can observe that cluster gradients can indicate the behaviour of the arriving illumination. Sudden changes in luminance caused by occlusion or reflected discontinuities can be recognised as step-like changes in the integral, whilst gradual changes from surface curvature or light falloff appear as shallower gradients. If a new sample lies within the bounds of an existing cluster or its distance from the nearest cluster is within a percentage threshold μ of the sample then we increment the cluster's sample count, and add the sample's R_v and I_v values. If it lies outside the threshold, we assume that our current clusters do not sufficiently represent the integral at that luminance and insert a single element cluster. The parameter μ dictates how sensitive the overall framework is to changes in luminance, and is the minimum percentage change in sample luminance that our framework can differentiate between during filtering. Our algorithm is not overly sensitive to μ and values between 0.1 and 1.0 (± 10 % to ± 100 %) produce visibly similar results for our test scenes. We use the same μ value of 0.5 for all images rendered in this paper. Increasing μ improves performance since clusters are merged more readily, providing fewer clusters per pixel. Merging existing clusters relies on a heuristic comparing the absolute luminance difference (first derivative), and change in gradient (second derivative) between neighbouring clusters:

$$S_{(n,n+1)} = (C_{n_\Delta} - C_{n+1_\Delta}) \cdot \phi + \frac{C_{n+1_{\min}} - C_{n_{\max}}}{C_{n_{\max}}} \tag{6}$$

where C_Δ is the gradient of the cluster C:

$$C_\Delta = (C_{\max} - C_{\min})/(C_{\text{num}} - 1)$$

To perform our merging we evaluate $S_{(n,n+1)}$ for each neighbouring cluster pair in the current layer, merging the clusters where S is smallest in order to maintain a minimal cluster set. We use a constant ϕ to weight the influence of gradient change in our heuristic, and have found a value of 0.75 to behave well by experimentation. Favouring gradient change provides a better balance between cluster merging at extreme ends of the luminance range. For low intensity samples the gradient change is comparatively small, so increasing ϕ favours merging between lower energy clusters, where discontinuities have less visible impact. For high luminance samples the opposite is true, and high values of ϕ separate high intensity outliers and discontinuities. It would be desirable to replace S and μ by a perceptual metric, allowing more optimal clustering.

4.2 High intensity sample rejection

Our high intensity noise removal approach builds on recent density estimation methods by DeCoro et al. [3] and Pajot et al. [10] improving on efficiency, robustness and suitability in image-based filtering. In our approach, we filter irradiance from individual path vertices, as opposed to the final radiance contributions of a complete path. This allows us to ignore texture changes as a source of variance, and by using per-vertex contributions we reject a lower percentage of samples since we can isolate noise from multiple sources (Fig. 4). Neither method operates well over low sample densities, resulting in a large proportion of contributions being removed across a region, providing few input samples for an image based filter. We take the view that it is better to filter contributions with a risk of artefacts, than to remove radiance entirely from a region which would provide no indication of the underlying materials or light transport. Algorithm 1 outlines our noise removal strategy. By storing clusters in ascending order of luminance we can process high luminance clusters first and if accepted, enable remaining non-zero clusters in the pixel since we know their lower energies can be handled during the filtering stage. After an initial pass of our algorithm, we make incremental updates when new clusters are created by trivially finding $L_{\max}$, and only running the algorithm again when parameters change or after a set number of frames. Since we use disjoint clusters

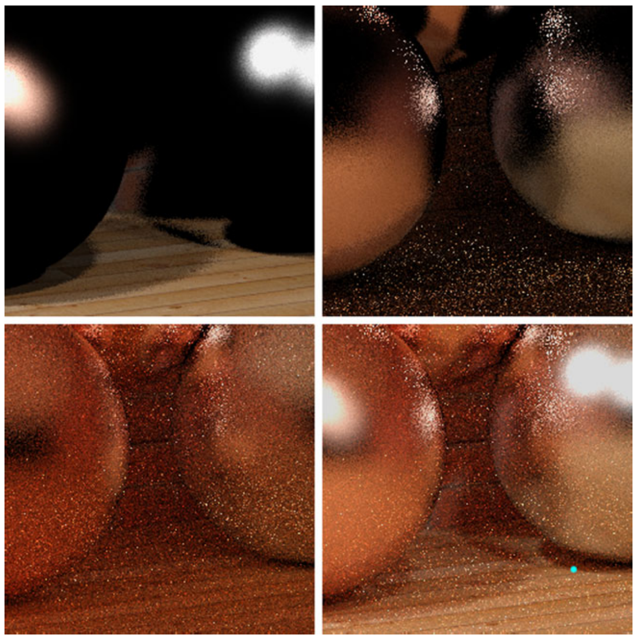


Fig. 4 Example of the layers in our framework. *Top*: Direct lighting $v = 1$ (*left*) and single bounce indirect $v = 2$. *Bottom*: $v \geq 3$ (*left*) and composite image $v > 0$ (*right*)

Algorithm 1 High intensity noise removal. A full search is often unnecessary, once an acceptable high luminance density is found (Lines 10 and 23)

```
1:  $d \leftarrow$ global density threshold
2:  $L_{\max} \leftarrow 0$                              ▷ Highest accepted luminance
3:  for $depth = 0 \rightarrow 3$ do
4:      $density \leftarrow 0$                          ▷ Num. similar samples
5:      $c \leftarrow$ # clusters $- 1$
6:      while $c \geq 0$ do
7:          $C \leftarrow$ Clusters[$c$]
8:          $density \leftarrow density + C_{\text{num}}$
9:          bool $t \leftarrow density \geq d$
10:         if $C_{\min} \leq L_{\max}$ then $t \leftarrow true$
11:         if $!t$ then                         ▷ Check pixel neighbours
12:             for all pixels $P$ in kernel do
13:                 for all clusters $C_p$ in $P$ do
14:                     float $v \leftarrow$ Overlap($C_p$, $C$)
15:                     if $v > 0$ then
16:                         $density \leftarrow density + C_{p_{\text{num}}}$
17:                         $t \leftarrow t \oplus (density \geq d)$
18:                         if $t$ then goto 21
19:         if $t$ then
20:             $L_{\max} \leftarrow$Max($L_{\max}$, $C_{\max}$)
21:             while $c \geq 0$ do
22:                 Clusters[$c$] $\leftarrow$ set as enabled
23:                 $c \leftarrow c - 1$
24:         $c \leftarrow c - 1$                     ▷ where $\oplus$ is boolean OR
```

to represent the sample distribution, as opposed to a sparse set of individual samples, we need to estimate the number of similar samples in the clusters of neighbouring pixels. Our parameter μ is employed again to determine cluster similarity. Treating clusters as sub-integrals of their respective pixel luminance integrals, we compute the difference between the two clusters as a proportion of the neighbouring pixel cluster (denoted C_p). In other words, the intersection of the areas A, of the definite integral for the current cluster and C_p as a ratio of C_p:

$$\text{Overlap}(C_p, C) = \frac{A(C') \cap A(C_p)}{A(C_p)} \tag{7}$$

where C' is the current pixel cluster extended according to μ. Since we store a complete, compact representation of the rendered samples, we can adjust our noise removal parameters during or post-render, allowing us to clean up a final image, without having to choose parameters in advance. This is useful for removing residual noise in the estimate with minimal bias. Additionally, since we do not rely on a high dimensional domain, we can extend our kernel width σ with minimal performance impact. This improves filtering for large regions of sparse samples such as caustics, whilst still removing high intensity noise in regions corrupted with more general noise.

5 Illumination preserving filter

Our method builds upon the principles of the cross-bilateral filter, utilising depth and normal buffers to identify changes in geometry and world space position, and a Gaussian filter to smooth local pixel regions. Our method's effectiveness is independent of texture features, resulting in more usable samples using the same kernel size. Discontinuities caused by occlusion and high frequency BSDFs provide both strong edges and high variance, and hence do not fit into a cross-bilateral framework. We therefore make use of the discrete Poisson probability distribution to provide additional filter weights that detect high gradient changes not present in range buffers. We operate over each depth layer individually, with attention to only the discontinuities present in the current layer. First, this allows us to filter samples without the corruption of noise from features in the remaining layers, meaning we have more usable samples in our input to smooth noise. Second, our filtering is less restricted since we only constrain our filter to discontinuities that affect the current layer. This allows us to filter indirect lighting across the shadow boundaries of direct lighting. Figure 4 is an example of strong direct lighting boundaries diluted by noise from indirect lighting, and where discontinuities in the indirect lighting layers on the glossy spheres interfere with one another despite both resulting in visible edges in the final render.

5.1 Poisson based filter weights

We use the Poisson probability distribution as a basis to compute luminance similarity weights for each contributing pixel c in our filter kernel, centred around the target pixel p that we are de-noising. In order to identify high frequency changes in luminance across our pixel neighbourhood we combine the per-pixel clusters in our filter kernel using our parameter μ once again as an indicator of luminance change. This process and our filtering is performed per layer for each pixel in the image, and is outlined in Algorithm 2. In the resulting set of kernel clusters, we store the radiance $\hat{R}$, irradiance $\hat{I}$ and number of samples N contributed from each filter pixel individually in order to compute our Poisson weights and final radiance. We first assume that the light transport for all filter pixels and the target pixel are the same. For each pixel c in our filter kernel, we take the number of samples N_f contributed to the kernel cluster and compute the expected probability of observing N_p events; the number of events empirically evaluated in the pixel p

Algorithm 2 Incident radiance filtering

```
 1: $P \leftarrow 0$                                   ▷ Final radiance for pixel p
 2: $KClusters \leftarrow$ initialise kernel clusters from $p$
 3: for all pixels $c$ in filter kernel do
 4:     for all clusters $C$ in $c$ do
 5:         for all clusters $K$ in $KClusters$ do
 6:             $b \leftarrow \text{Overlap}(C, K)$                  ▷ Eq. (7)
 7:             if $b$ then
 8:                 add $C$ to $K$
 9:                 break
10:         if $!b$ then
11:             insert $C$ into $KClusters$
12: for all clusters $K$ in $KClusters$ do
13:     for all pixels $c$ in filter kernel do
14:         $W_{\text{pois}}(c) \leftarrow \text{Poisson}(K[N_c], K[N_p])$   ▷ Eq. (8)
15:         $W(c) \leftarrow W_{\text{pois}}(c) \cdot W_g(c) \cdot W_n(c)$     ▷ Eq. (9)
16:         $\hat{R}^c_p \leftarrow$ radiance from $K$                ▷ Eq. (13)
17:         $\hat{R}_p \leftarrow \hat{R}_p + (\hat{R}^c_p \,/\, K[N_c] \cdot W(c))$   ▷ Eq. (14)
18:     $P \leftarrow P + \hat{R}_p$                          ▷ Add normalised radiance
```

that we are de-noising. Using the Poisson equation (Eq. (3)) we obtain a probability W_{pois} for the filter pixel c:

$$W_{\text{pois}}(c) = \frac{N_c^{N_p} e^{-N_c}}{N_p!} \tag{8}$$

obtained from a precomputed lookup table. We normalise these probabilities with respect to the maximum expected probability (with $\lambda = k = N_c$ from Eq. (3)), and interpret them as similarity weightings for each pixel. Performing this for each kernel cluster allows us to de-noise multiple overlapping illumination sources, glossy reflections and high and low frequency features (see Fig. 5). Combined with our per pixel geometry weights W_n and Gaussian weights W_g, we obtain the overall similarity for this cluster in each filter pixel c with respect to our target pixel:

$$W(c) = W_g(c) \cdot W_n(c) \cdot W_p(c) \tag{9}$$

In the majority of cases, pixel neighbourhoods exhibit smooth irradiance, producing a single kernel cluster representative of all samples across the neighbourhood, resulting in equal Poisson weights W_{pois} across our filter pixel. For more complex regions that include illumination edges we obtain multiple kernel clusters, each one representing a distinct band of luminance that are together responsible for illumination edges. Each cluster is filtered independently so that we can deal with overlapping illumination features from multiple sources without interference.

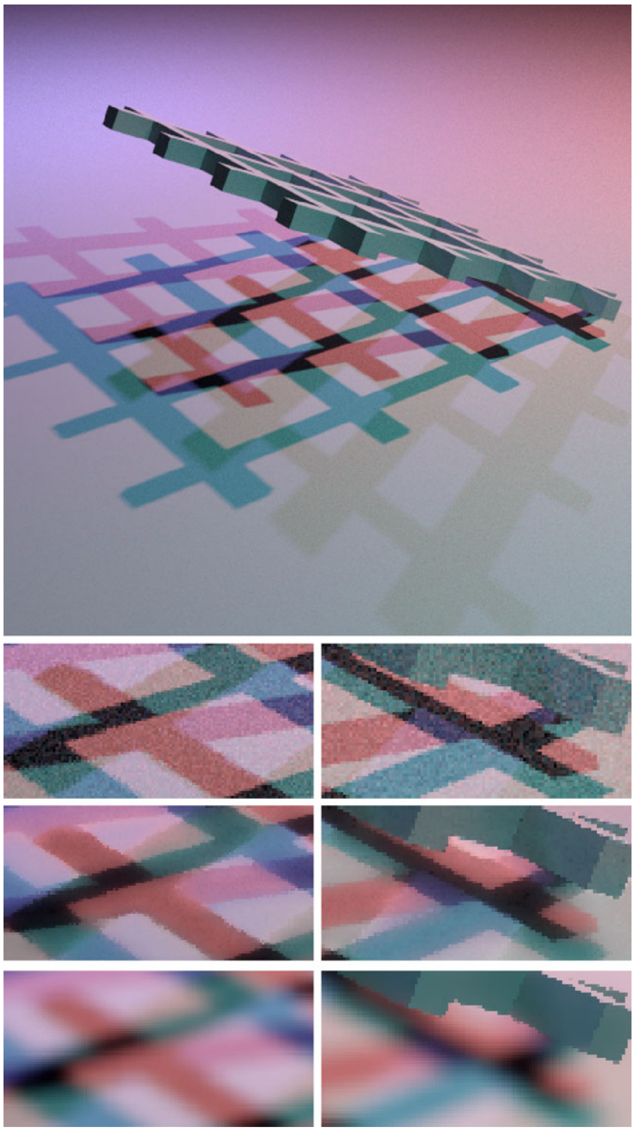


Fig. 5 A green grid above a white plane lit by three coloured lights of varying size. *Top*: Path traced reference. *Second row*: Path traced input with 16 spp. *Third row*: Our approach filters noise effectively handling overlapping sources, shadow gradients, and colours. *Bottom row*: cross-bilateral filter

5.2 Pixel radiance computation

Computing our filter weights now allows us to evaluate a weighted radiance contribution from each pixel c using our set of kernel clusters. In physically based rendering, reflectance at a surface $f_r(x_i, \boldsymbol{\omega}_i, \boldsymbol{\omega}'_i)$ can be broken down into a scalar BRDF coefficient $f_s(x_i, \boldsymbol{\omega}_i, \boldsymbol{\omega}'_i)$ and the texture detail $f_t(i)$ for a surface location x_i. Assume in our framework we wish to obtain a radiance value for a single irradiance contribution I_v (recall Eq. (5)). To compute R_v the scalar surface reflectance (abbreviated f_s) of the BRDF and the path throughput f_t of any preceding specular vertices up to and including d need to be accounted for:

$$R_v = I_v \cdot \prod_{n=1}^{d} \big(f_t(n) \cdot f_s(n)\big) \tag{10}$$

During path tracing, we account for $f_t(1)$ to $f_t(d)$ scaled by the specular throughput $f_s(1)$ to $f_s(d-1)$ for each pixel using a set of texture buffers, one for each layer, accounting for anti-aliasing of textures and detail maps for each pixel. The unknown BRDF scalar coefficient can be obtained by rearranging Eq. (10):

$$f_s(d) = \frac{R_v}{I_v \cdot f_t(d) \cdot \prod_{n=1}^{d-1} f_t(n)} \tag{11}$$

As described in the previous section, each kernel cluster for pixel p contains an individual $\hat{R}$ and $\hat{I}$ value from each contributing pixel c, including values from p itself. We can calculate a representation of $f_s(d)$ for all samples in the kernel cluster from each pixel c using Eq. (11) with $\hat{R}$ and $\hat{I}$ and the texture buffer for c:

$$\hat{f_s} = \frac{\hat{R}}{\hat{I} \cdot \hat{f_t}} \quad \text{where } \hat{f_t} = f_t(d) \cdot \prod_{n=1}^{d-1} f_t(n) \tag{12}$$

Using this approach, high frequency detail is encapsulated for a pixel in $\hat{f_t}$ and the lower frequency changes in BRDF are represented by the $\hat{f_s}$ term calculated for each kernel cluster. For glossy surfaces, f_s can also change quickly, but in such cases the irradiance filtering becomes more tightly restricted by the Poisson derived filter weights. To remove noise caused by insufficient sampling of the illumination, we need to use the irradiance data $\hat{I}^c$ and $\hat{f}_s^c$ from each contributing pixel, while preserving the texture detail $\hat{f}_t^p$ for the pixel being de-noised. We compute the final radiance contributed to p by pixel c for a kernel cluster as:

$$\hat{R}_p^c = \hat{I}^c \cdot \hat{f}_s^c \cdot \hat{f}_t^p \tag{13}$$

We then combine $\hat{R}_p^c$ for each c using the respective filter weights to influence its contribution to the radiance of the kernel cluster:

$$\hat{R}_p = \sum_{c=0}^{\sigma^2} \hat{R}_p^c \cdot W(c) \tag{14}$$

This provides a filtered contribution from each kernel cluster, representing the contributions of a subset of samples in the pixel integral at that luminance range. The total pixel radiance is the sum of $\hat{R}_p$ for each kernel cluster, normalised using the sum of the sample counts in those clusters. To improve performance, we can merge kernel clusters that have uniform Poisson weights over the kernel reducing the loop over K in Algorithm 2 to a single cross-bilateral filter over the irradiance. This can improve performance for smooth regions, without loss in quality.

6 Results and discussion

We implement our method using a CPU-based renderer running on a 2.66 GHz Intel Core i7 920 with 8 Gb of RAM. We trace a single path per pixel using next event estimation at each vertex, making use of BSDF importance sampling and applying Russian roulette after three bounces. To demonstrate the generality of our parameters, we use the same settings for all images in the paper. For our method we use σ = 11 (our filter kernel size) and set μ = 0.5. Our high intensity noise removal uses a low density threshold, removing samples with fewer than four similar samples within the 11^2 pixel neighbourhood. For our comparisons with cross-bilateral filtering, we do not filter the direct lighting on specular or glossy surfaces, and use the same kernel size, texture, depth and normal buffers for both methods. Figure 6 shows a scene with complex realistic textures, pure specular and glossy objects, using materials based on aluminium and copper metals with varying degrees of roughness. We compare our method with the cross-bilateral filter, with two sets of parameters. The first is tweaked post-render to try and visibly reduce noise, and the second uses a narrow range to try and preserve maximum texture detail in the specular reflections and on visible diffuse surfaces. Relying on the incident illumination means that using our method, this texture detail is always preserved, regardless of the parameters used. Our edge detection means we avoid overly blurring high frequency features such as the overlapping direct shadows on the floor, as well as the glossy reflections. We effectively remove noise from all areas of the image using a combination of our high intensity noise removal and irradiance filtering. The cross-bilateral filter fails to reduce noise across small texture features, like the mortar in the brickwork, since the kernel is too restricted by the texture range buffer to suitably distribute the energy across the image.

Figure 7 shows the complex Ajax bust in a flooded Cornell box. The face is partially occluded by the peak of the hat, and geometry detail is only visible due to self-shadowing. Our irradiance-aware Poisson weights minimise blurring of the self-shadows providing more depth and clarity to the beard and hair especially in regions lit indirectly. We can also effectively handle transmission, with fewer artefacts even at low sample counts at which the path tracing image is plagued with noise, and the texture detail is barely visible. Our irradiance filtering ensures that we maximise the use of the input samples in geometrically similar regions, preserving the texture even through the water and handling the sparse illumination better than the cross-bilateral filter.

Figure 5, shows our ability to handle multiple light sources of varying intensities, colours and sizes. The bars result in numerous overlapping shadows with hard to soft boundaries. Since the cross-bilateral filter is ignorant of shadow boundaries, it blurs across the image removing all

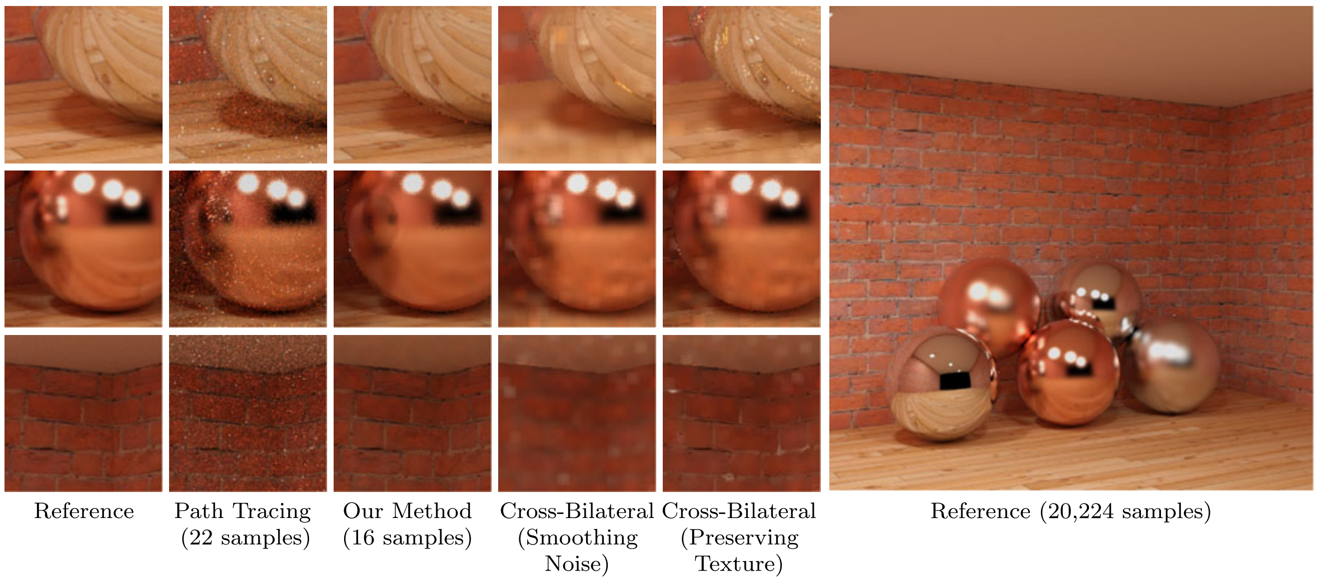

Reference
Path Tracing (22 samples)
Our Method (16 samples)
Cross-Bilateral (Smoothing Noise)
Cross-Bilateral (Preserving Texture)
Reference (20,224 samples)

Fig. 6 This scene contains multiple glossy materials and complex textures. Our method preserves all texture detail, and edges in reflections and illumination. By adjusting the range sensitivity of the texture weightings, the cross-bilateral filter can be tuned to smooth noise, or preserve texture detail, but not both and still cannot preserve small texture features. The images for path tracing and our method are equal time comparisons (see Table 1) and the cross-bilateral filter uses the same input samples as our method

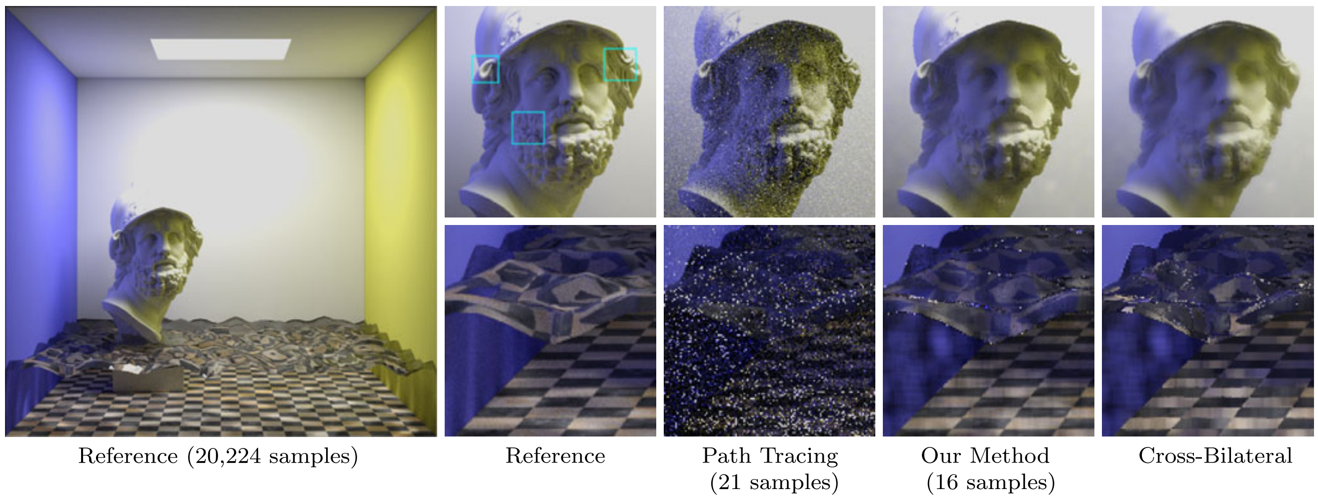


Fig. 7 Ajax bust in water, displaying complex geometry, transmission, caustics and self shadowing. Our method blends soft shadows whilst preserving features on the face of the Ajax bust caused by self shadowing. We better preserve texture detail on the water surface, and in the geometry, where the depth and normal buffers are not sufficient to avoid blurring shadow detail. The images for path tracing and our method are equal time comparisons (see Table 1) and the cross-bilateral filter uses the same input samples as our method

Table 1 Time spent in each stage of our algorithm with reference to path tracing after running both algorithms for equal time. The path tracing stage for our method includes management of the sample clusters, and generation of the geometry and texture buffers. Cluster updates and noise removal are performed every four frames

Scene		Samples	Path tracing	Noise removal	Filtering	Total time
Glossy Spheres	*PT*	22	44.2 s	–	–	44.2 s
	Our method	16	32.7 s	0.51 s	8.6 s	41.8 s
Ajax in Water	*PT*	21	45.2 s	–	–	45.2 s
	Our method	16	35.1 s	0.74 s	7.4 s	43.2 s

Table 2 Clusters and memory statistics for our results. Single denotes the number of clusters representing just one sample, and the last column is the percentage of outlying samples. Memory use includes our texture, depth and normal buffers (29 MB per image)

Scene	Total clusters	Single	Memory	Outliers
Spheres	6.19M	1.67M	236 MB	0.109 %
Ajax	5.56M	1.91M	223 MB	0.274 %

shadow detail. We can preserve these edges, differentiating between samples from different sources and effectively de-noise the image without merging features.

7 Conclusions

We have described a cluster based filtering approach that can reduce noise in Monte Carlo methods, whilst preserving image discontinuities from both high and low variance sources. Using a layered approach, we can separate irradiance contributions according to their path length. This enables us to de-noise each layer independently, oblivious to discontinuities and noise in remaining layers and maximising the use of the input samples. In the future we would like to address the problem of finding optimal per-pixel and layer specific kernel bandwidths, providing a good balance between noise, bias and performance as rendering progresses. Further performance and visual improvements could be gained by applying perception based metrics to produce a more optimal clustering scheme.

Acknowledgements We would like to thank the reviewers for their helpful suggestions to improve the paper, and Jotero.com for donation of the Ajax model to the community. The work presented in this paper was funded by an EPSRC doctoral training grant, EPSRC grant EP/I031243/1 and RIVIC. We would like to be notified about any adoption of this method into rendering software since our funding council seeks to record the impact of its funded research.

References

1. Cline, D., Talbot, J., Egbert, P.: Energy redistribution path tracing. In: ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, pp. 1186–1195. ACM, New York (2005)
2. Dammertz, H., Sewtz, D., Hanika, J., Lensch, H.: Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In: Proc. High Performance Graphics, vol. 2010, pp. 67–75 (2010)
3. DeCoro, C., Weyrich, T., Rusinkiewicz, S.: Density-based outlier rejection in Monte Carlo rendering. Comput. Graph. Forum **29**(7), 2119–2125 (2010)
4. Eisemann, E., Durand, F.: Flash photography enhancement via intrinsic relighting. ACM Trans. Graph. **23**(3), 673–678 (2004)
5. Gassenbauer, V., Křivánek, J., Bouatouch, K.: Spatial directional radiance caching. Comput. Graph. Forum **28**(4), 1189–1198 (2009)
6. Hachisuka, T., Jarosz, W., Weistroffer, R.P., Dale, K., Humphreys, G., Zwicker, M., Jensen, H.W.: Multidimensional adaptive sampling and reconstruction for ray tracing. ACM Trans. Graph. **27**(3), 1–10 (2008)
7. Kontkanen, J., Räsänen, J., Keller, A.: Irradiance filtering for Monte Carlo ray tracing. In: Monte Carlo and Quasi-Monte Carlo Methods 2004, pp. 259–272. Springer, Berlin (2004)
8. Křivánek, J., Gautron, P., Pattanaik, S., Bouatouch, K.: Radiance caching for efficient global illumination computation. IEEE Trans. Vis. Comput. Graph. **11**(5), 550–561 (2005)
9. Lehtinen, J., Aila, T., Chen, J., Laine, S., Durand, F.: Temporal light field reconstruction for rendering distribution effects. ACM Trans. Graph. **30**(4) (2011)
10. Pajot, A., Barthe, L., Paulin, M.: Sample-space bright spots removal using density estimation. In: Proc. of Graphics Interface 2011, pp. 159–166 (2011)
11. Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., Toyama, K.: Digital photography with flash and no-flash image pairs. ACM Trans. Graph. **23**(3), 664–672 (2004)
12. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive rendering with non-local means filtering. ACM Trans. Graph. **31**(6), 195 (2012) (11 pages)
13. Schwenk, K., Kuijper, A., Behr, J., Fellner, D.: Practical noise reduction for progressive stochastic ray tracing with perceptual control. IEEE Comput. Graph. Appl. **32**(6), 46–55 (2012)
14. Sen, P., Darabi, S.: On filtering the noise from the random parameters in Monte Carlo rendering. ACM Trans. Graph. **31**(3), 18 (2012) (15 pages)
15. Suykens, F., Willems, Y.D.: Adaptive filtering for progressive Monte Carlo image rendering. In: WSCG, pp. 220–227 (2000)
16. Veach, E., Guibas, L.J.: Metropolis light transport. In: SIGGRAPH '97: Proc. Computer Graphics, pp. 65–76 (1997)
17. Ward, G.J., Rubinstein, F.M., Clear, R.D.: A ray tracing solution for diffuse interreflection. In: SIGGRAPH '88, vol. 22, pp. 85–92 (1988)

Ian C. Doidge received the B.Sc. degree in computer science from Swansea University. He is currently working towards the Ph.D. degree conducting research into algorithms and data structures for ray tracing, global illumination and rendering.

Mark W. Jones has received the B.Sc. and Ph.D. degrees from Swansea University. He is now a lecturer in the Department of Computer Science at Swansea University, where his research interests include global illumination, visualisation, volume graphics and associated algorithms and data structures.