Effective Despeckling of HDR Images

Anthony Pajot, Loïc Barthe, Mathias Paulin

Effective Despeckling of HDR Images

Anthony Pajot, Loïc Barthe, Mathias Paulin*
IRIT - University of Toulouse, France

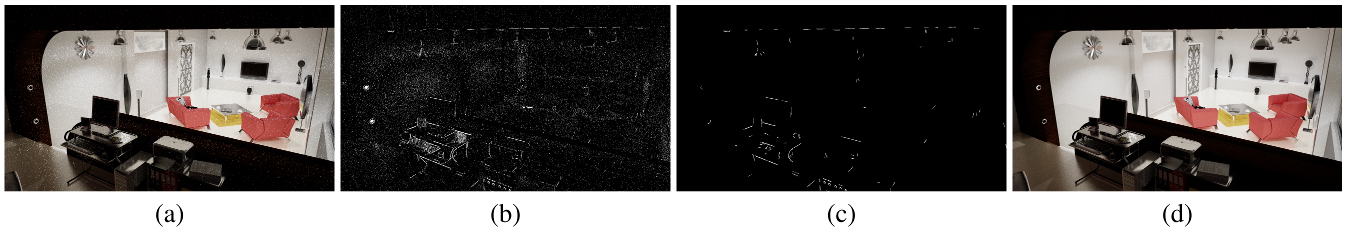

Figure 1: *(a) The original speckled HDR image. (b) The pixels selected by the median detection pass. (c) In white, the groups obtained by the groups filtering pass, mostly corresponding to objects edges. These pixels will not be reconstructed. (d) The despeckled HDR image. Image resolution:* 1600×900.

CR Categories: I.4.3 [Image Processing]: Enhancement—;

Keywords: High Dynamic Range, Despeckling, Robust Filtering

1 Introduction

High Dynamic Range (HDR) images are common in computer graphics. In these images, speckles can appear for various reasons, such as high variance for Monte-Carlo-based rendering engines. These speckles are responsible for large artefacts if non-robust post-processing methods are used, such as tonemapping operators relying on a maximal or average luminance value. Ensuring that all post-processing operators are robust is tedious, therefore we propose to handle these speckles before any other processing is done. This way, we ensure that any HDR image post-processing pipeline produces acceptable results.

In the specific case of Monte-Carlo-based rendering, *sample-space* methods such as [DeCoro et al. 2010] or [Pajot et al. 2011] process the samples during the rendering process to detect those that can cause bright spots. These methods pre-suppose a stochastic rendering method with known properties such as samples independence, but provide a complete control over the committed error. *Image-based* methods are not based on any asumption on the way the image is produced. In addition of being more general, they require less computational power and are often easier to integrate in an image-production pipeline.

A popular image-based method to remove speckles is to use bilateral-filtering or more specific versions [Xu and Pattanaik 2005], but these methods induce a noticeable blur and artefacts (Figure 3, 3^{rd} row and Figure 4, 3^{rd} row). Our method is an image-based method, but it is designed to not induce any blur nor artefacts.

Our approach consists of two steps: we first detect speckles, and then reconstruct them using pixels not tagged as speckles. This is highly different from (bilateral-) filtering methods, which process all pixels the same way, reconstructing each pixel using all their neighbours. Our method does not introduce blur, naturally preserves edges and thin image features, yielding an artefact-free reconstruction.

2 Tag-and-Reconstruct Despeckling Filter

Speckles detection and tagging is done in three passes, which we now describe in the three next paragraphs.

First, a binary image I_d of potential speckles is computed by comparing the input image I to a despeckled image I_m. I_m is obtained using median filtering, which is highly robust to speckles, but produces highly blurred images. Median filtering is performed on a square pixel-centered window of width w_m. For each pixel of coordinate (x, y), a potential speckle is detected if $l(I(x,y))/l(I_m(x,y)) > \tau$, where τ is experimentally set to 2, and $l()$ is the luminance function. Figure 1b shows I_d obtained from the image in Figure 1a, using $w = 5$.

*{pajot,barthe,paulin}@irit.fr

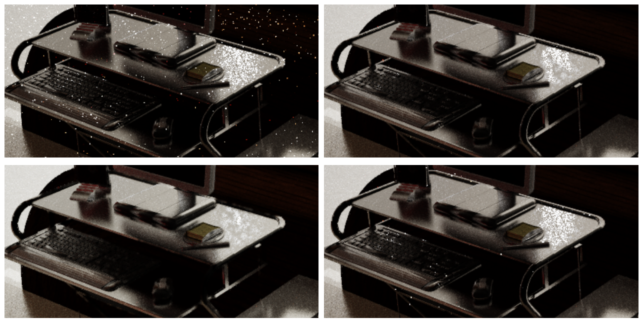

Figure 2: *Top left: original, right: final despeckled image. Bottom left: without clusterization. Note that the borders of the table have been considered as speckles and therefore reconstructed. Right: clusterization without using a threshold ratio. Note that speckles that lie on the table border are still present, as they are part of the border cluster.*

Then, another binary image I_c, corresponding to coherent image features whose size is less than w_m, is computed from I_d. This allows us to differentiate actual speckles from small image features which must not be reconstructed. This separation is done by a clusterization/group-filtering step: for each pixel coordinate (x, y) for which $I_d(x, y) = 1$, we compute the associated cluster if (x, y) is not already part of a cluster. This cluster is the set of connected pixels such that, for any (x', y') belonging to the cluster, $I_d(x', y') = 1$ and $1/r < l(I(x', y'))/l(I(x, y)) < r$, where r is a user-defined threshold ratio experimentally set to 10. This last criteria allows us to correctly handle actual speckles that are on the border of an object, border which is also present in I_d. Once all the clusters have been computed, we only keep those that contain at least N_c pixels (experimentally set to 20), therefore removing from I_c all potential speckles that are too isolated, and which are most likely to be actual speckles. The r parameter is a tolerance parameter. Increasing it makes bigger clusters, but they are more likely to mix image features and speckles. Decreasing it leads to smaller clusters, maybe separating pixels of a same feature in separate clusters. Pixels of these clusters could be wrongly considered

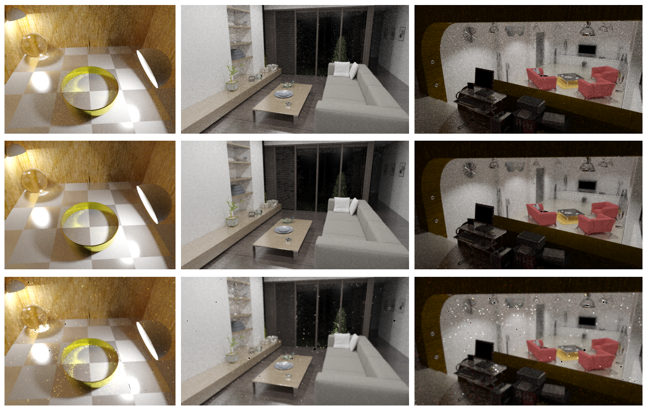

Figure 3: *Results of our method, and comparison with Xu et al. method. First row: original HDR images. Second row: after despeckling by our method. Third row: Results obtained by Xu et al. method, with a window size of* 4 *and a range factor set to* 0.01*. Fourth row: resolution of the original images.*

as speckles if their cluster are too small. Figure 1c shows I_c computed on Figure 1b, using $r = 10$ and $N_c = 20$, and Figure 2 shows the non-negligible impact of this step on the results.

Finally, pixels for which $I_d(x, y) = 1$ and $I_c(x, y) = 0$ are tagged as speckles.

Reconstruction of the tagged pixels is performed with a simple gaussian filter of width w_g (set to w), but it only considers pixels that are not tagged as speckles. This helps avoiding creating new speckles from the ones we just removed. Note that this also implies that a pixel (x, y) is reconstructed without using its original value $I(x, y)$.

Setting parameters: This technic is actually not sensitive to small variations of parameters τ, w, r and N_c (respectively set to 2, 5, 10 and 20). We have computed all the presented results using these values, and we suggest them as practical default values.

3 Results

Test setup: Our tests have been performed on a Intel core i7 3.07Ghz, using only a single core and a mildly optimized implementation. Our base HDR images have been obtained from path-tracing with a relatively low number of samples, on scenes where this rendering algorithm exhibits large variance, leading to numerous speckles. All the images in this section have been tonemapped using a simple exposure-based tonemapper, which does not lead to any noise reduction.

Results images: Figure 3 shows the three images before and after processing, as well as a comparison with results obtained using Xu *et al.* method. Close-ups presented in Figures 4 and 5 clearly show the effectiveness of our method. These figures illustrate the efficient removal of both high-frequency/large-amplitude and high-frequency/middle-amplitude noise, leaving only low-amplitude noise. In addition, no visible blur is added on textures, and edges and very thin features are well preserved (see the one-pixel-wide green plant at the bottom-right of the second scene's close-up in Figure 4). Comparatively, Xu *et al.* method adds a lot of blur as well as artefacts due to numerical instability caused by very large pixel intensity values, whatever the set of parameters we tried.

Computation times: Our method has a linear complexity with respect to the number of pixels. The first image in Figure 3 has been reconstructed in 0.7 seconds, while processing the second and third images took 0.5 seconds. Finally, processing the high-resolution image in Figure 1 took two seconds.

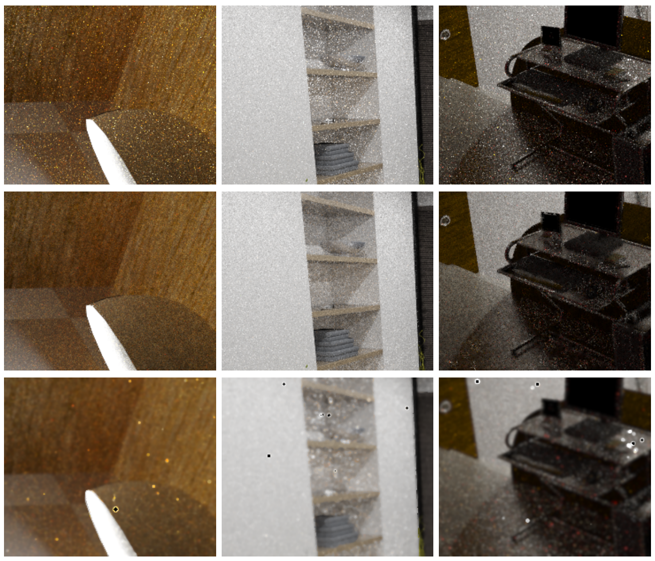

Figure 4: *First row: close-up on original HDR images tonemapped using a larger exposure value to make high-frequency/middle-amplitude noise more visible. Second row: after processing by our method. Third row: after processing by Xu et al. method.*

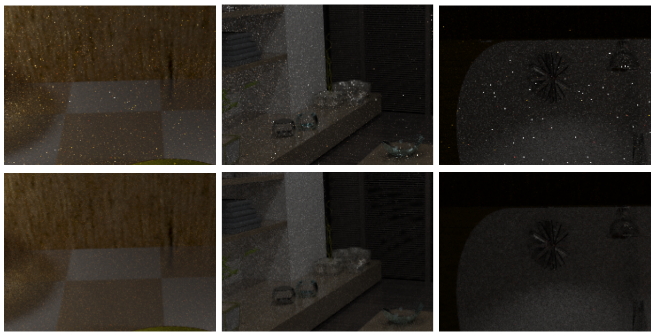

Figure 5: *Top row: close-up on original HDR images, using a low exposure value for the tonemapping to make high-frequency/large-amplitude noise more visible. Bottom row: after processing with our method.*

Handling low-dynamic-range images: Our algorithm can also be used for low-dynamic range images. However, in this case value differences would not be as large as in HDR images, leading to a less robust median test.

Discussion: Although correctly handling all the speckles, our method may tag high-frequency noisy features as speckles (as the not-well-converged large glossy reflection on the table of Figure 2) or small bright features such as some of the specular highlights of the plate in the second scene of Figure 3. In this case, the use of a GUI allowing the user to select regions which should be ignored would be a simple and appropriate solution.

References

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

Pajot, A., Barthe, L., and Paulin, M. 2011. Sample-space bright-spot removal using density estimation. In *GI '11*.

Xu, R., and Pattanaik, S. 2005. A novel monte carlo noise reduction operator. *IEEE Comput. Graph. Appl. 25*, 2, 31–35.