Effective Despeckling of HDR Images

Anthony Pajot* - Loïc Barthe* - Mathias Paulin*

SIGGRAPH Asia 2011 - December 15th, 2011

*IRIT-CNRS - Université de Toulouse, France

Outline

The goal

remove speckles from **HDR** images.

Example of result

before

after

1600 $\times$ 900, processing time : 0.2s on a core i7.

Details conservation

Lighted borders of the table, shadows edges, details of the keyboard, textures patterns, ...

Why *HDR* Images ?

Allows for easier post-processing :

- No need to despeckle the LDR version each time processing is changed
- More flexibility for post-processing : no need to use speckles-robust methods

For rendering : average multiple HDR images for more accurate results.

- high-frequency/High-amplitude noise removed
- low-frequency/low-amplitude noise as is, properly removed by averaging

State of the art

Image-space methods in the rendering community :

Bilateral filtering (for instance Xu *et al.* 05), leads to blur or smears.

Base of the Method

Key points of the method

- Reconstruct only speckled pixels $\Rightarrow$ detection
- Instead of detecting speckles, detect *non-speckled* pixels $\Rightarrow$ more general characterization
- Avoid using speckled pixels in reconstruction

Advantages of selectivity for reconstruction

no blur, no smears

Outline

Step 1, Detection : Characterization

Non-speckled pixel : intuitive characterization

part of a **coherent region**.

Coherent region

- Spatial coherency
- Color-space similarity

Implemented characterization

For each pixel p, build a set S_p of *contiguous* pixels *similar* to p. If $|S_p| \geq N_c$, p is not a speckle.

Step 1, Detection : Similar Pixels

$q \in S_p$ if :

- Spatial coherency : q is contiguous to a pixel in S_p
- Chromacity similarity : $\sqrt{(a^*(p) - a^*(q))^2 + (b^*(p) - b^*(q))^2} < d$
- p is not too bright : $\frac{L^*(p)}{L^*(q)} < r$

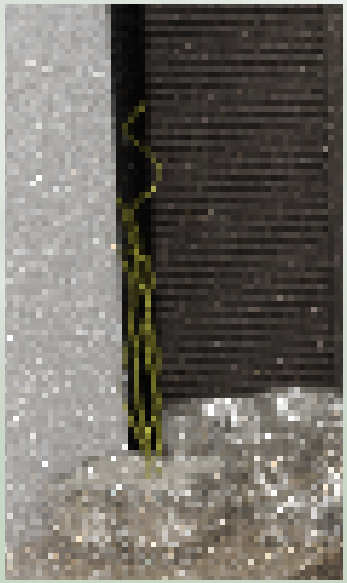
Non-commutativity

$$q \in S_p \not\Rightarrow p \in S_q$$

Step 1, Detection : Implementation

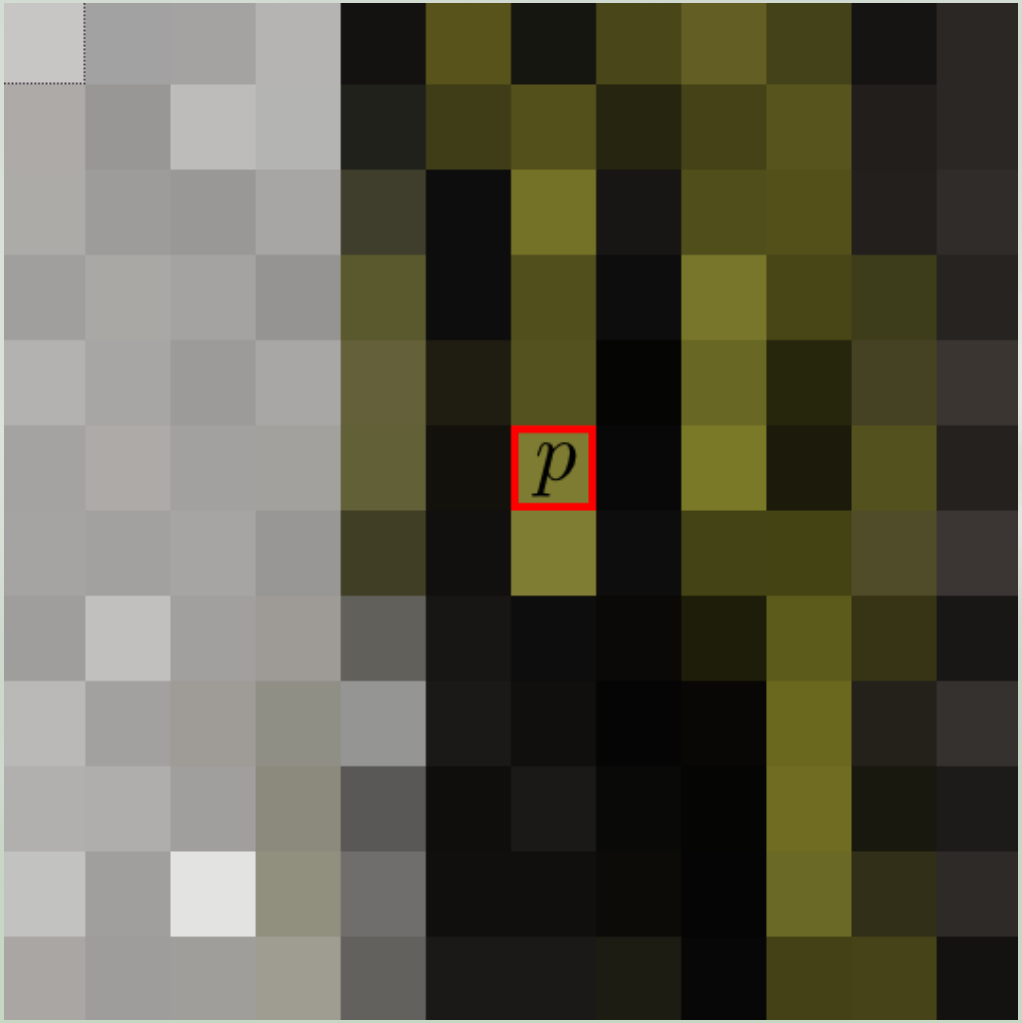
For each pixel p, build S_p by flood-filling until $|S_p| \geq N_c$.

Illustration : Base HDR image, tonemapped for display.



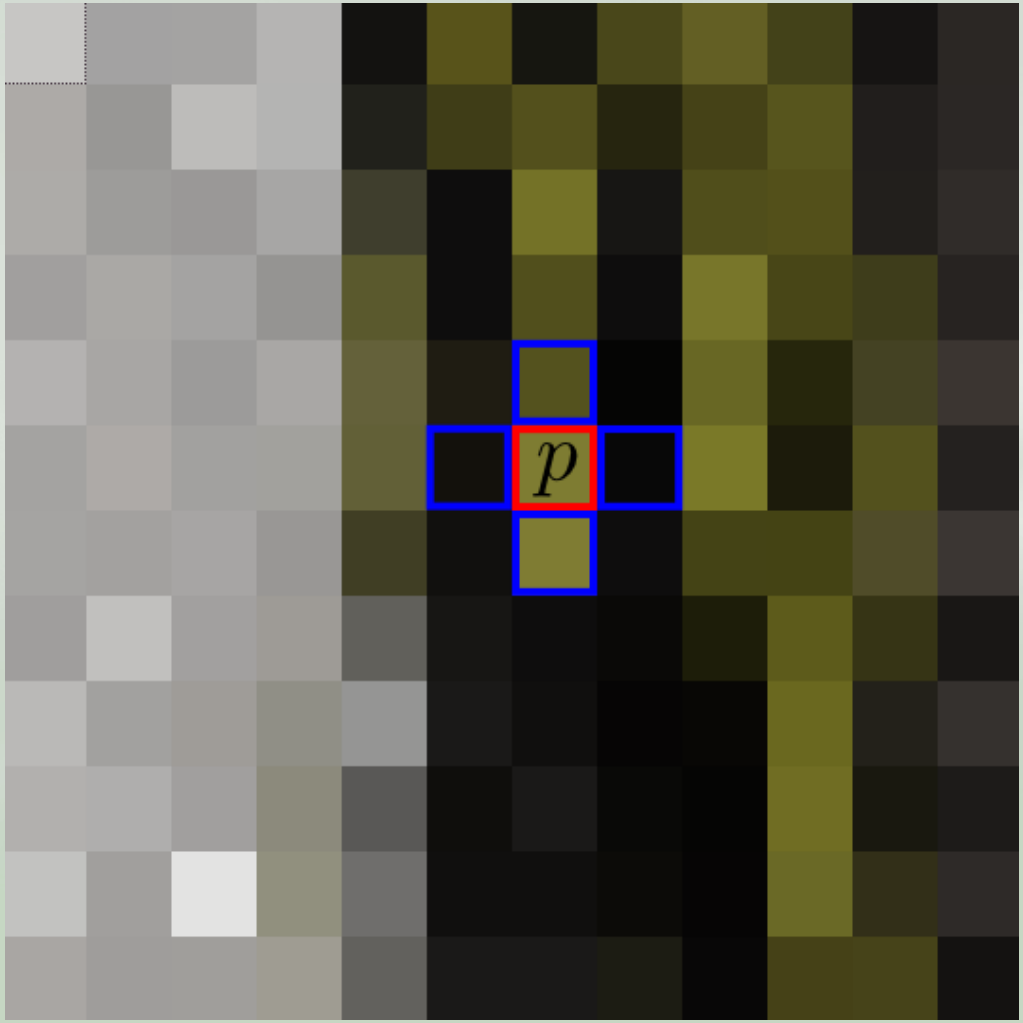
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 1$$



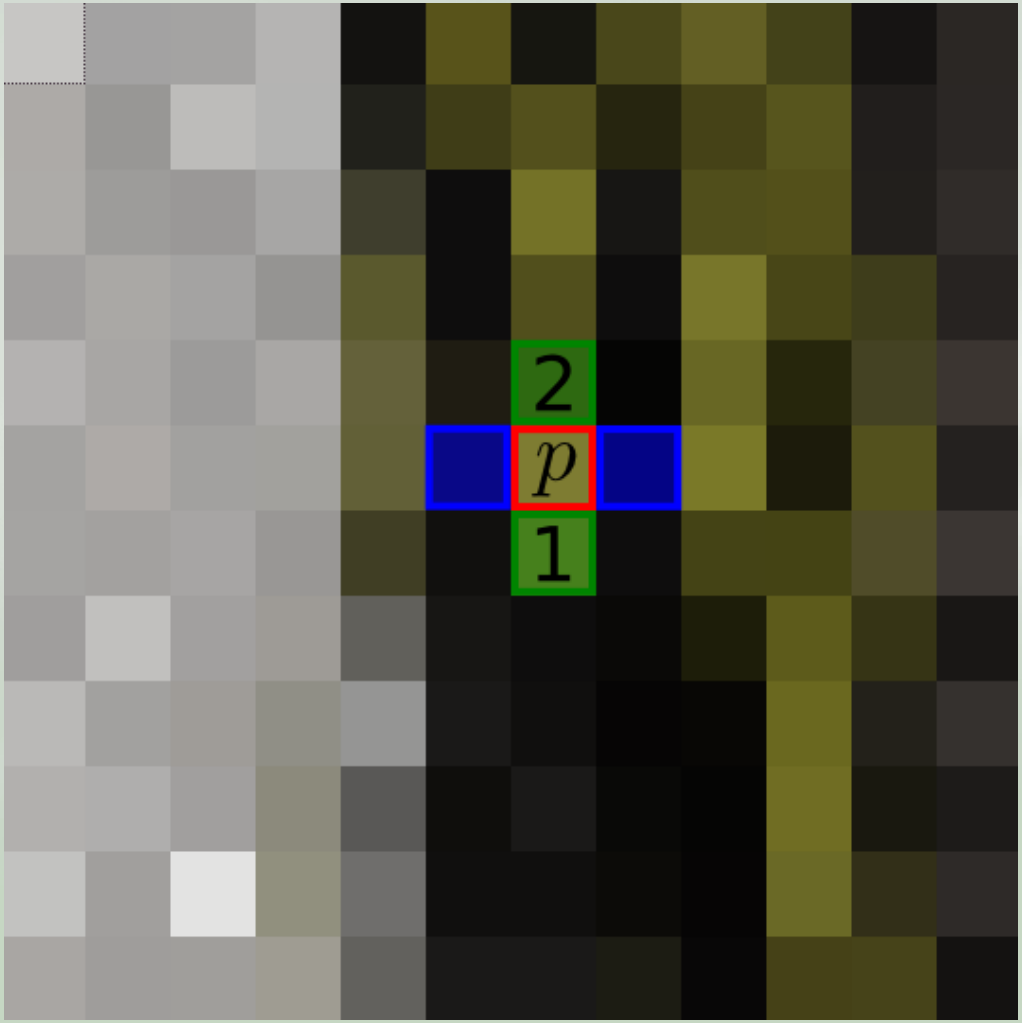
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 1$$



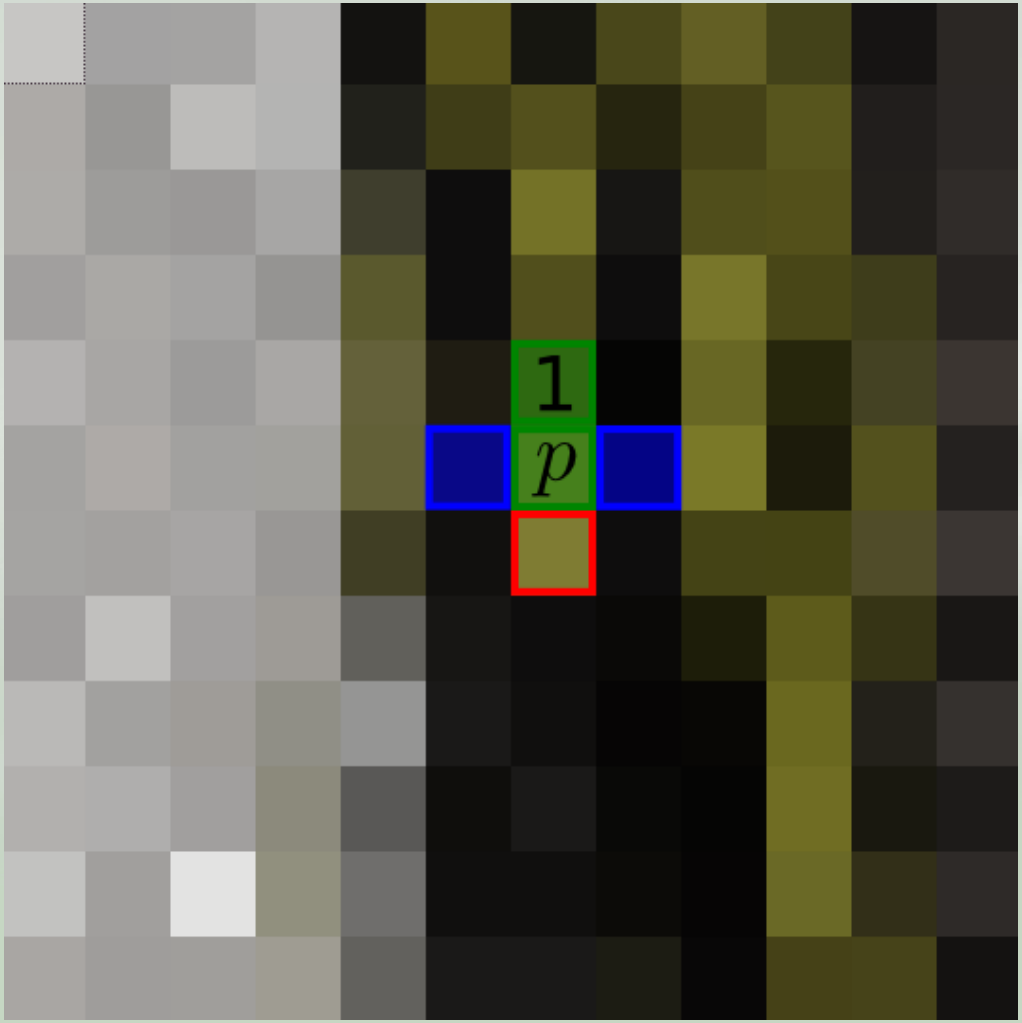
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 3$$



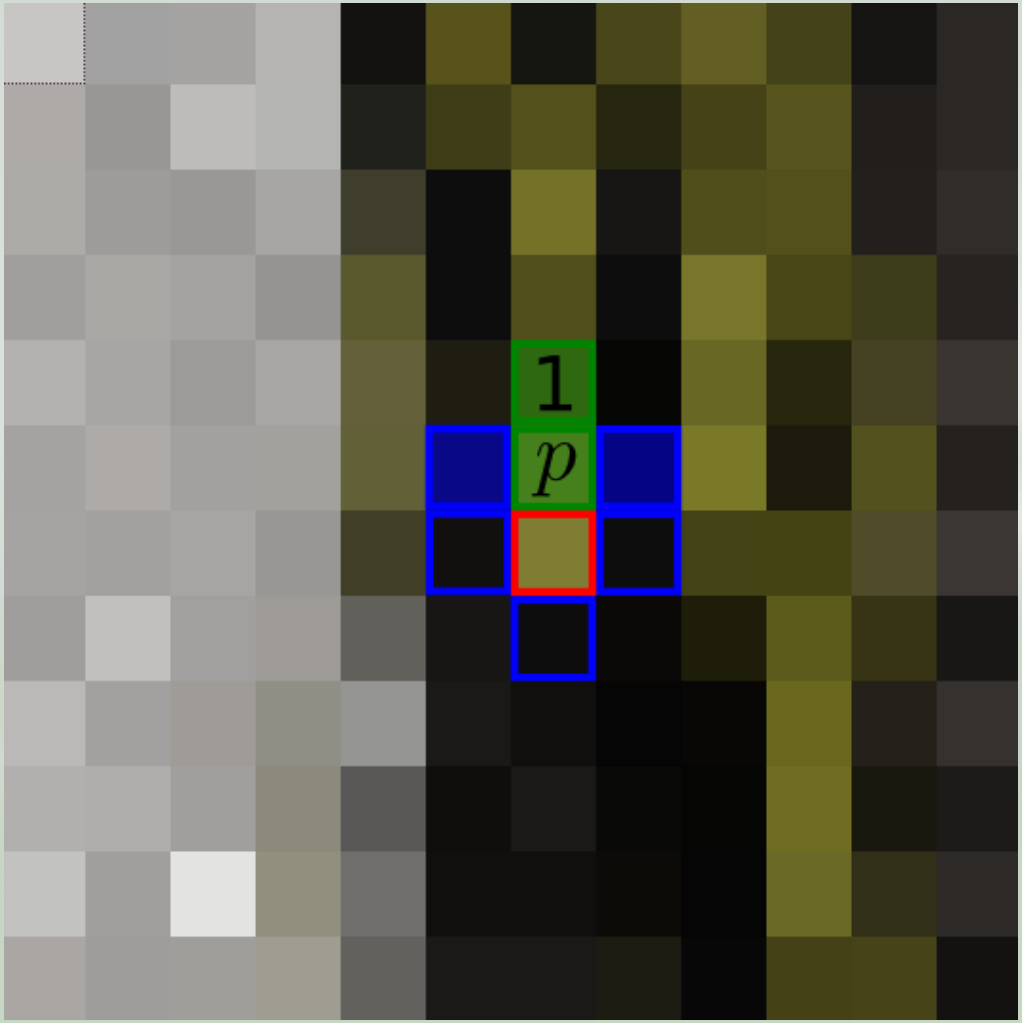
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 3$$



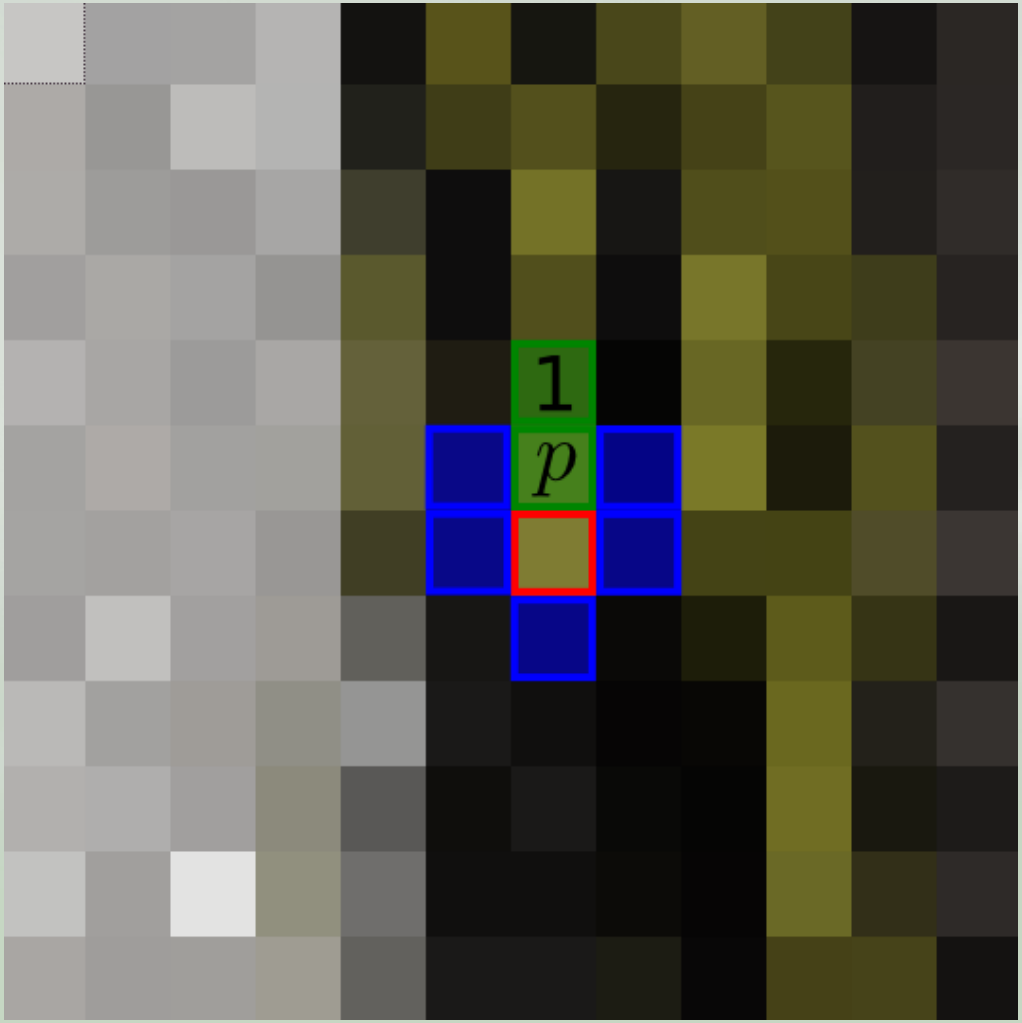
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 3$$



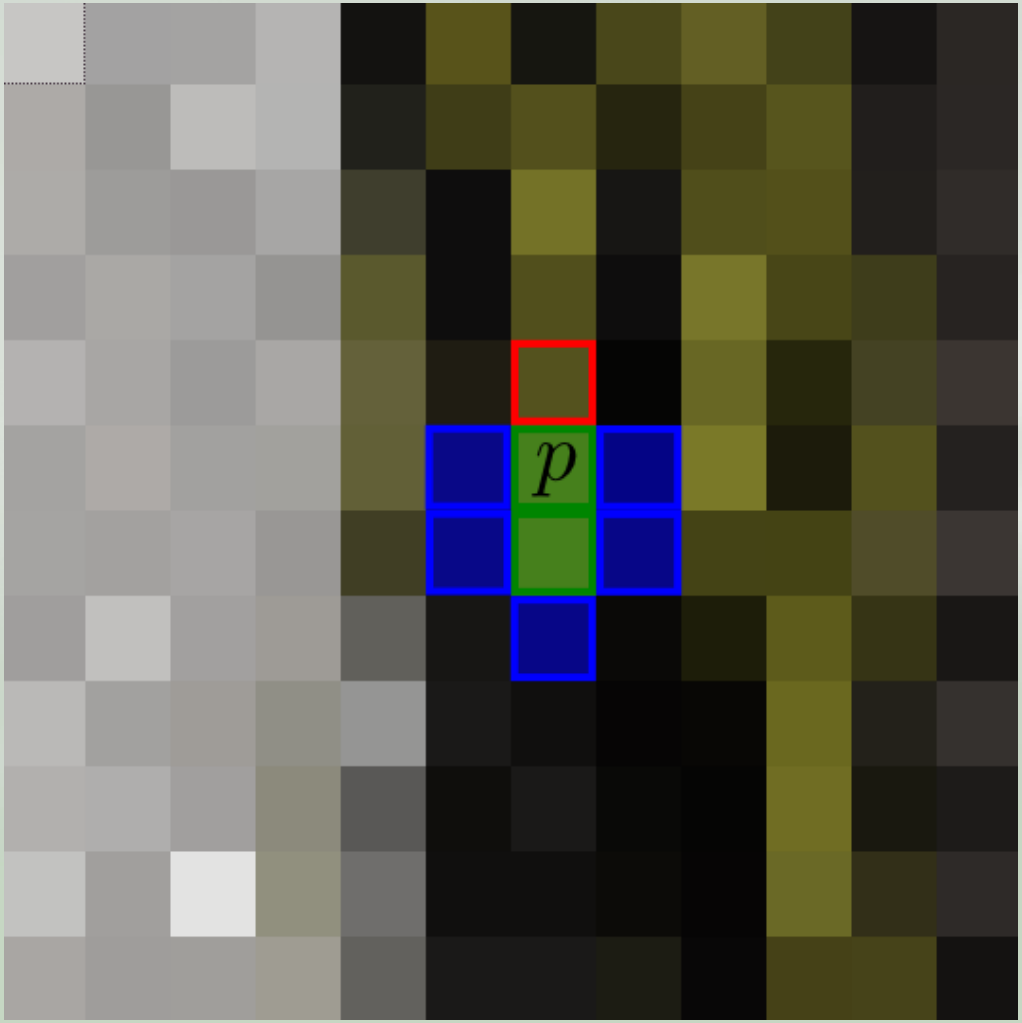
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 3$$



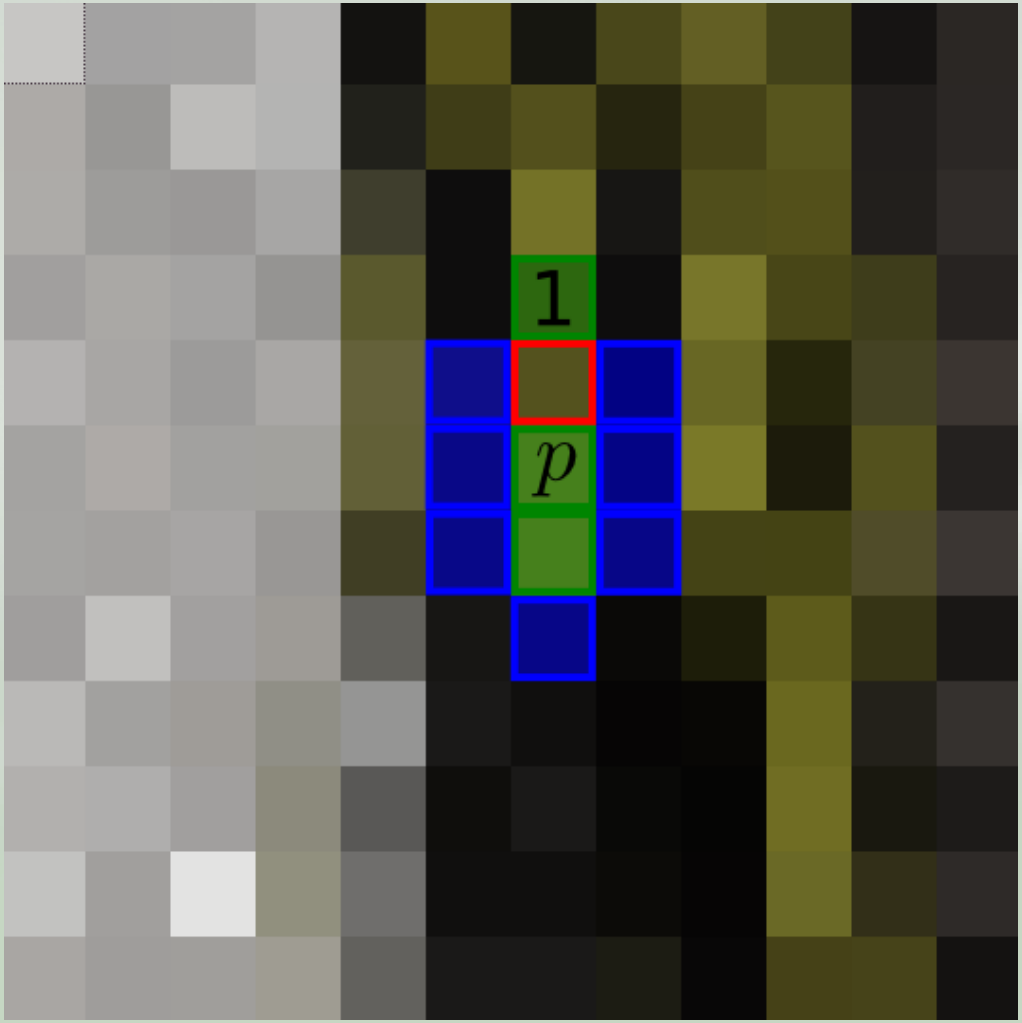
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 3$$



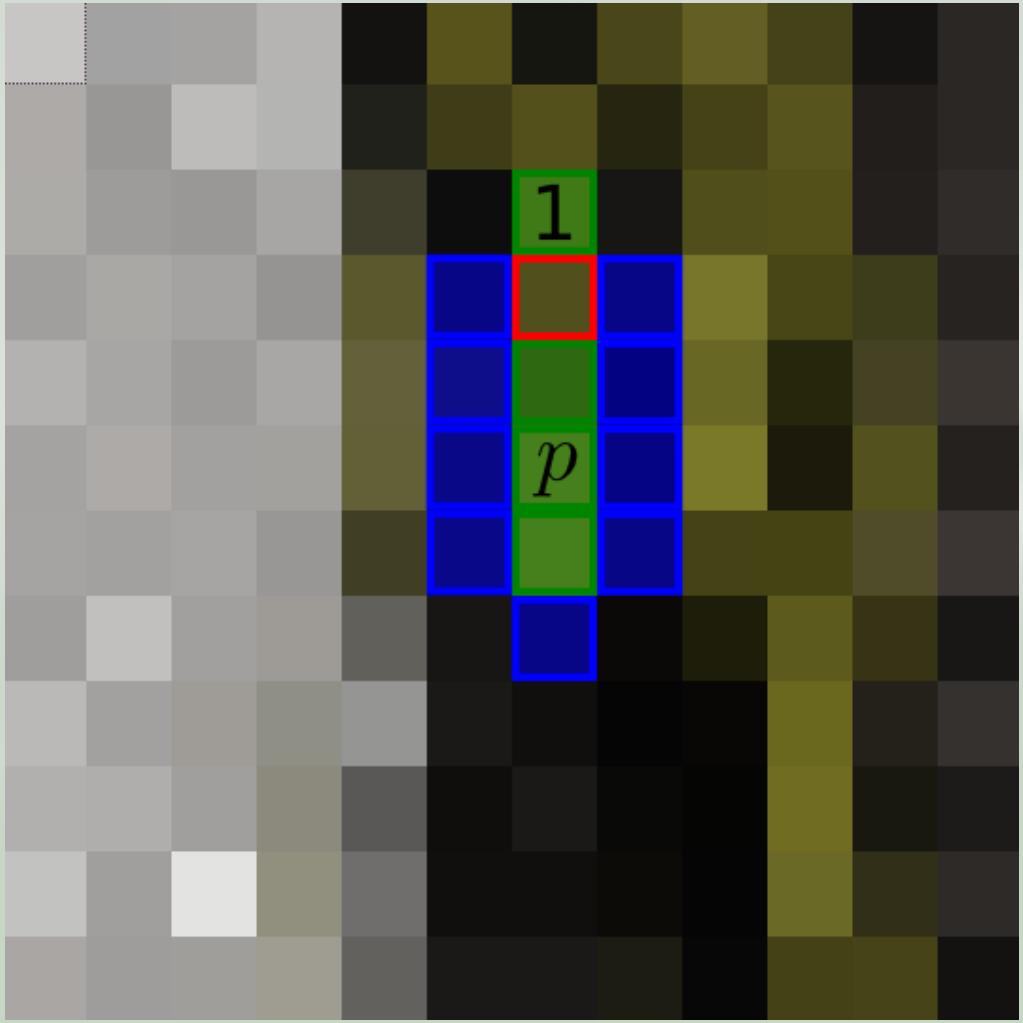
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 4$$



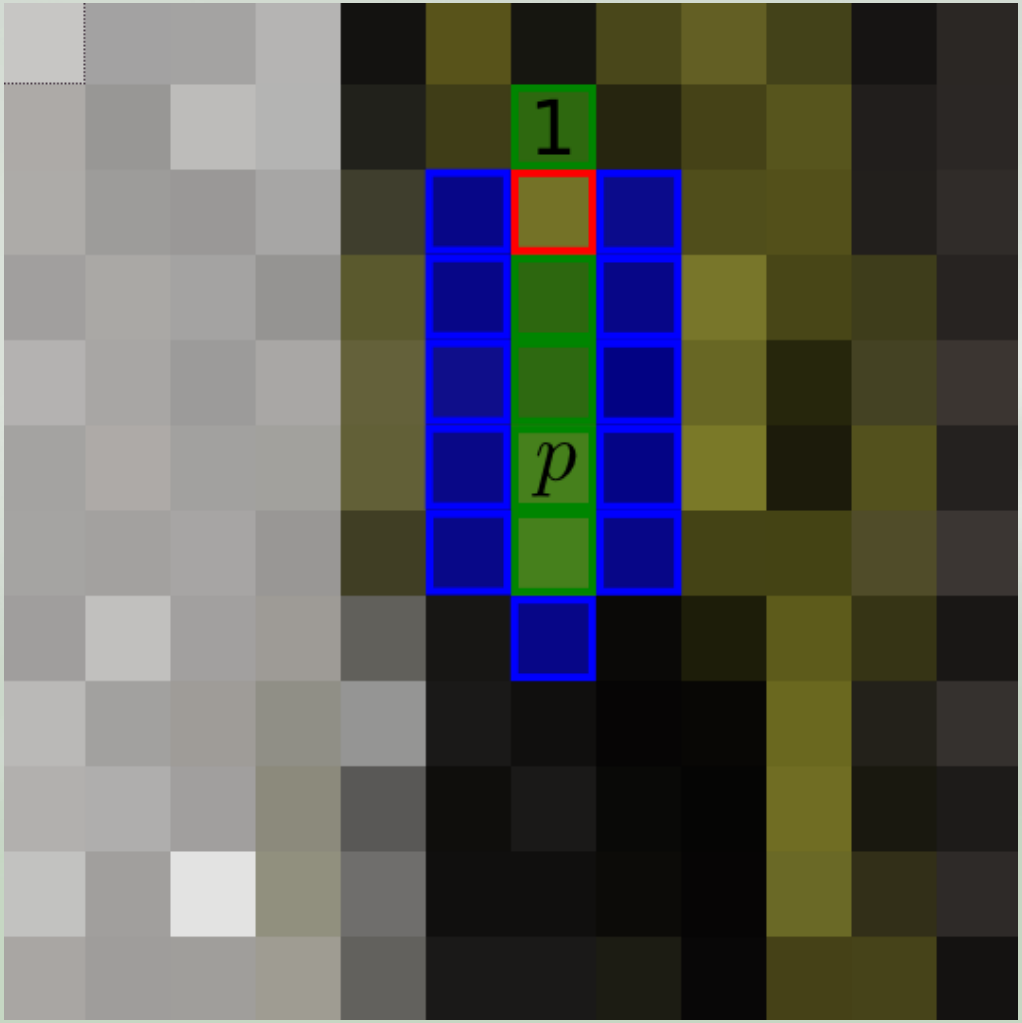
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 5$$



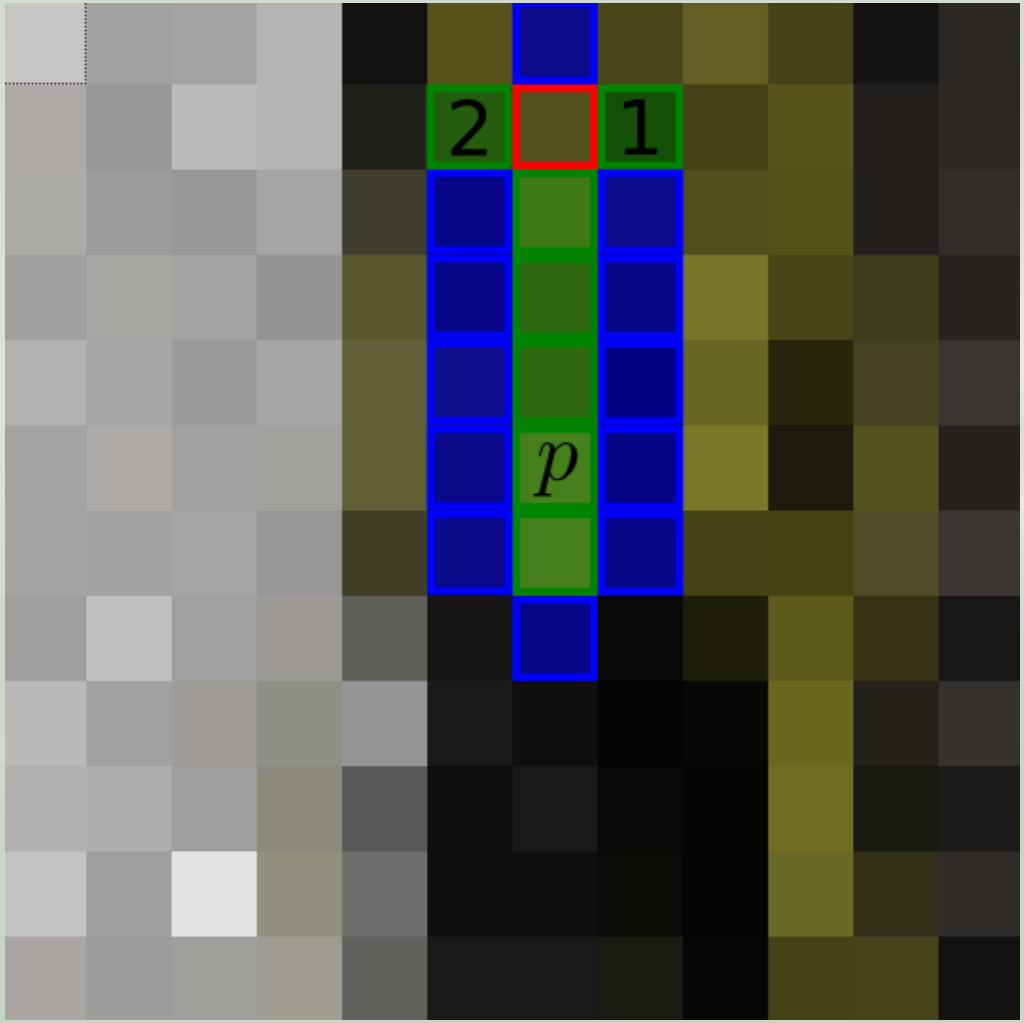
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 6$$



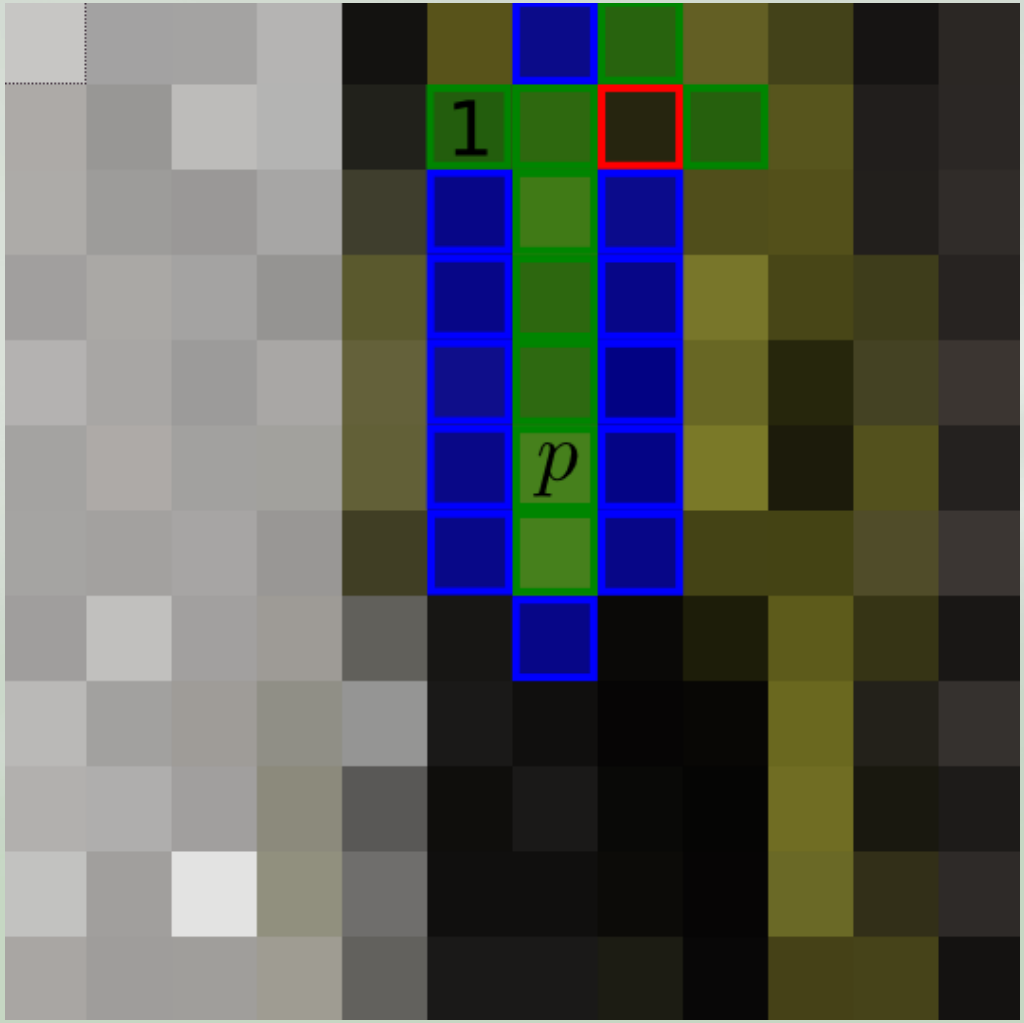
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = 8$$



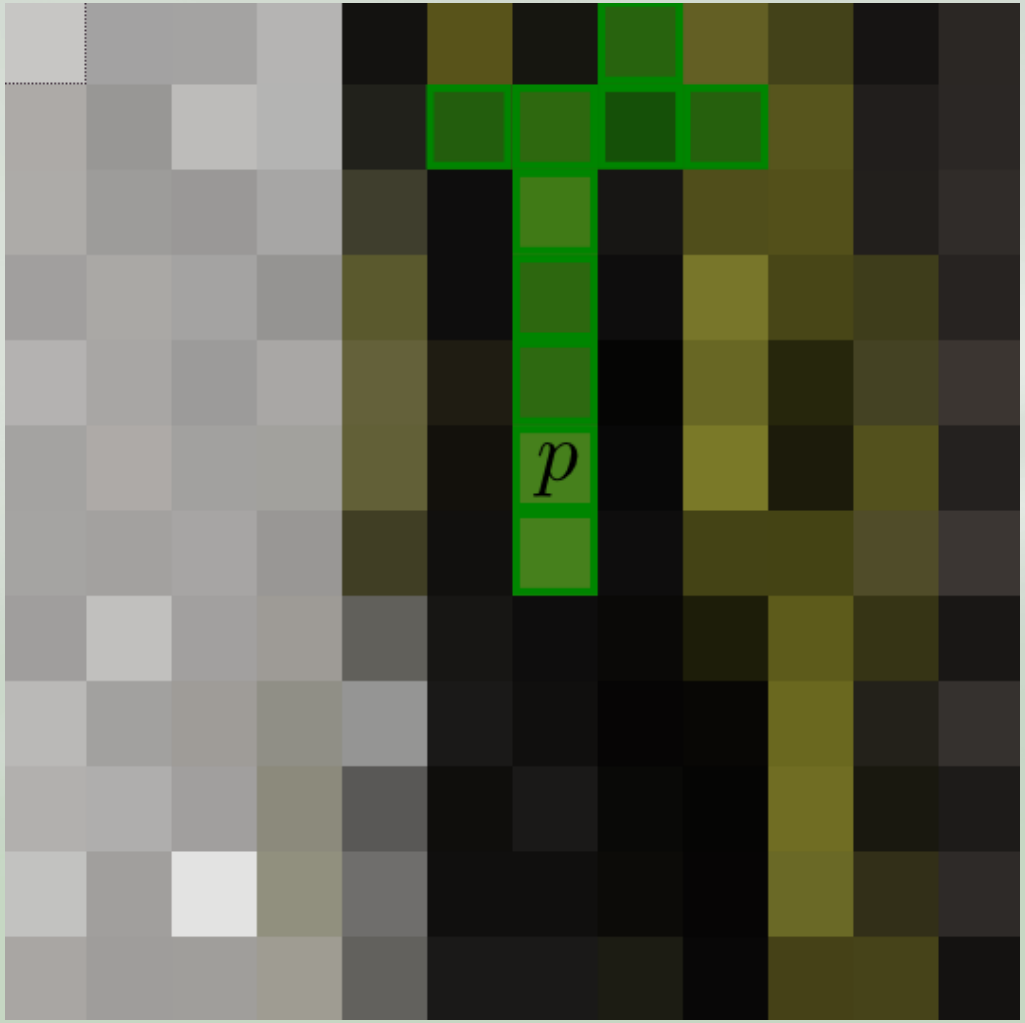
Step 1, Detection : Implementation

$$N_c = 10, |S_p| = \mathbf{10}$$



Step 1, Detection : Implementation

$|S_p| \geq N_c$, p is not a speckle



Clustering versus window-based detection

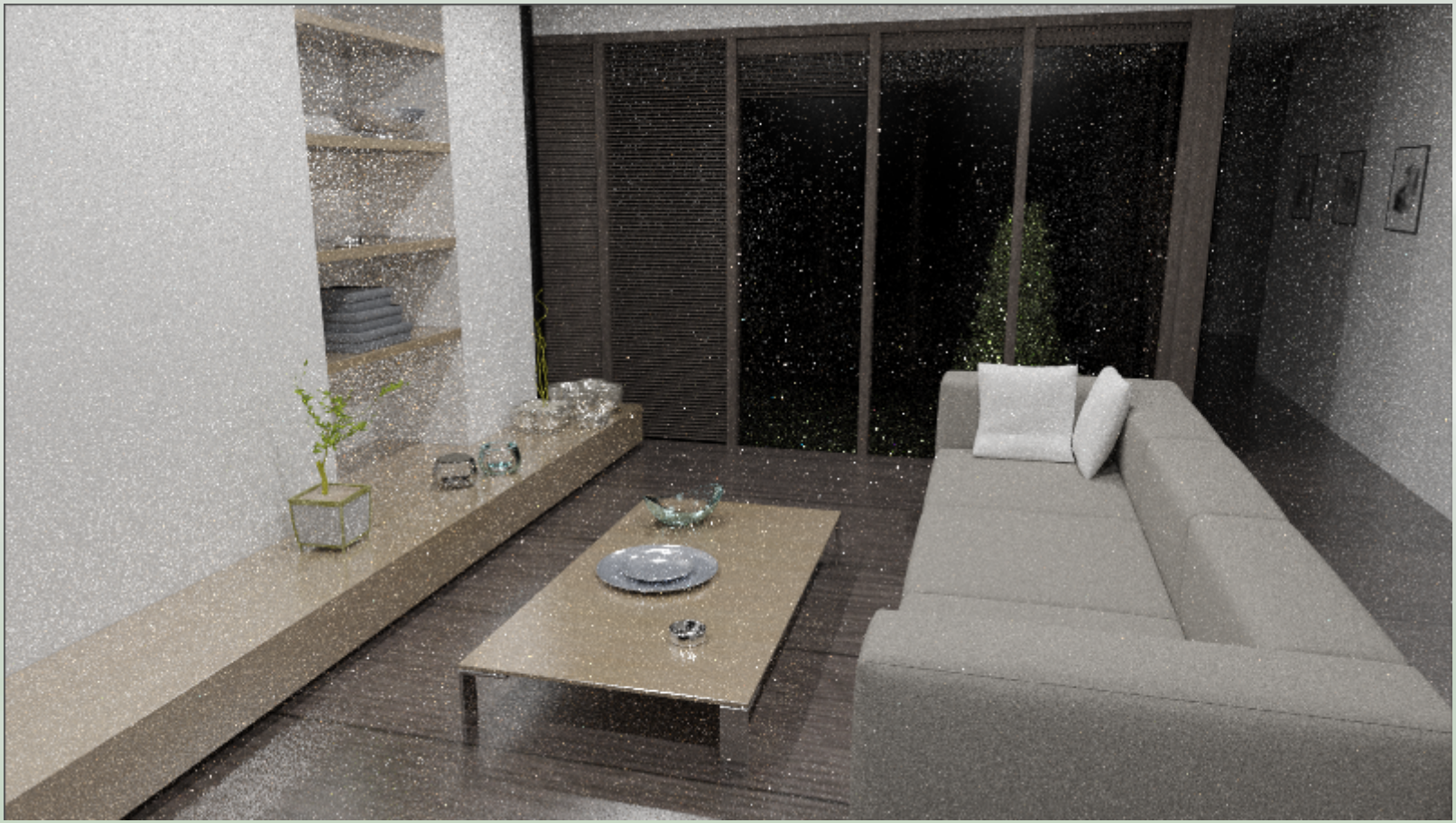
- Advantage : Independent of the shape of the features.
- Drawback : Parallel implementation requires per-thread pixel-tagging.

Step 1, Detection : Parameters

- N_c : size of feature, depends on the resolution. We use $N_c = 10$.
- d : chromacity distance in $L^*a^*b^*$ space, the higher the less restrictive. We use $d = 20$.
- r : lightness ratio, the higher the less restrictive, should be ≥ 1. We use $r = 1.1 - 1.5$.

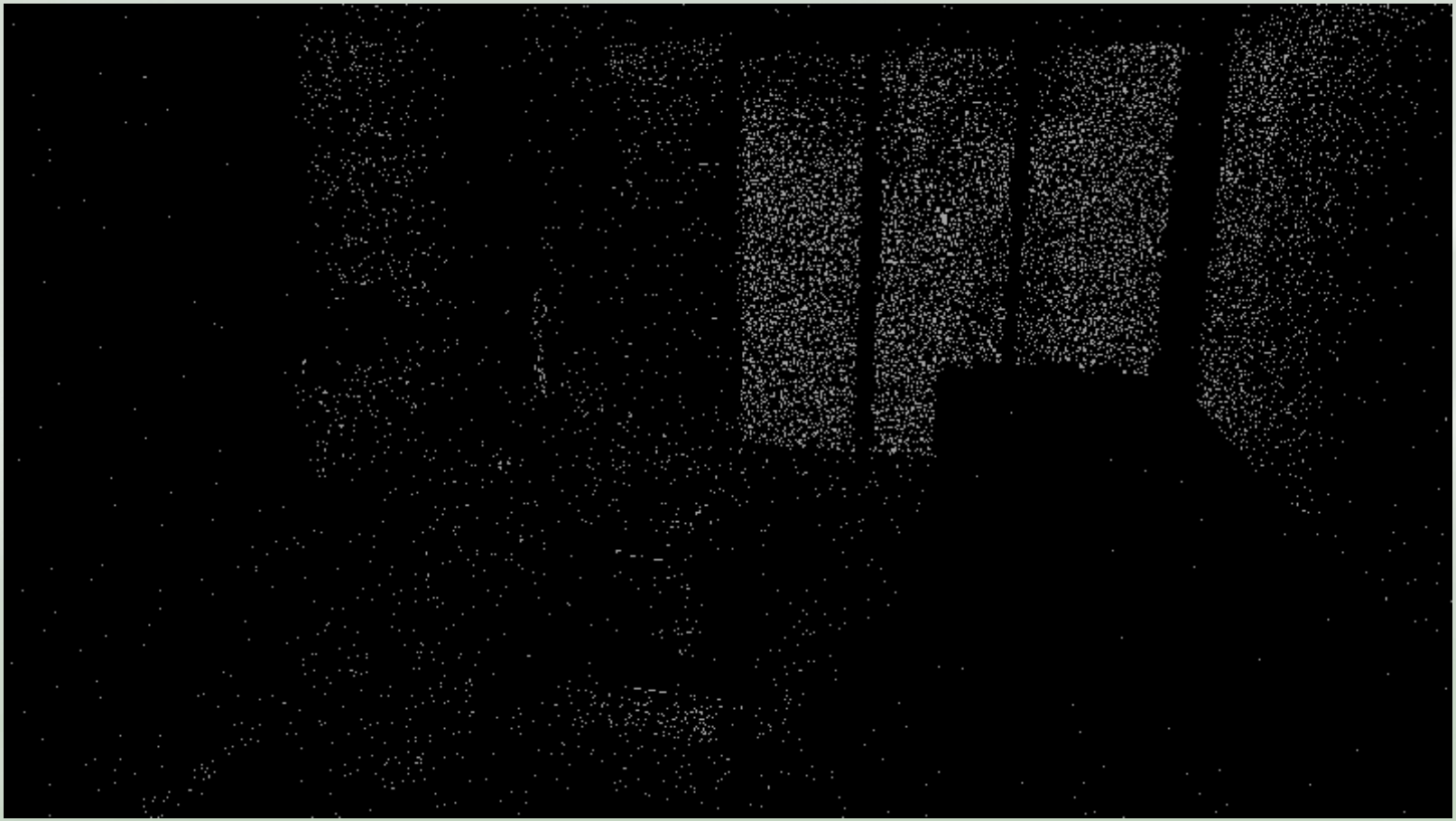
Step 1, Detection : Result

Base HDR image, tonemapped for display



Step 1, Detection : Result

Detected **speckles**



$N_c = 10, d = 20, r = 1.5$

Step 2, Reconstruction of the Speckles

Gaussian reconstruction on a square neighborhood of size w_g

- of the speckled pixels *only*
- ignoring speckles in the gaussian filter

Can be seen as a bilateral filter with binary range filter :

$$I_p(x,y) = \frac{\sum_{\mathcal{N}(w_g)} I(x',y')G(x',y',w_g)H(x',y')}{\sum_{\mathcal{N}(w_g)} G(x',y',w_g)H(x',y')}$$

where the range filter $H(x',y') = 0$ if (x',y') is a speckle, 1 otherwise.

Efficient Implementation

1. Precompute the $L^*a^*b^*$ version of the image
2. Clustering for all pixels
3. Reconstruction of all speckled pixels

Each step can be made in parallel on all pixels.

Outline

Extreme case (synthetic)

1600 × 900

Extreme case (synthetic) : detected speckles

$N_c = 10, d = 20, r = 1.2, 1600 \times 900$

Extreme case (synthetic)

$N_c = 10, d = 20, r = 1.2, 1600 \times 900$

processing time : 0.2s on a core i7

Extreme case (synthetic)

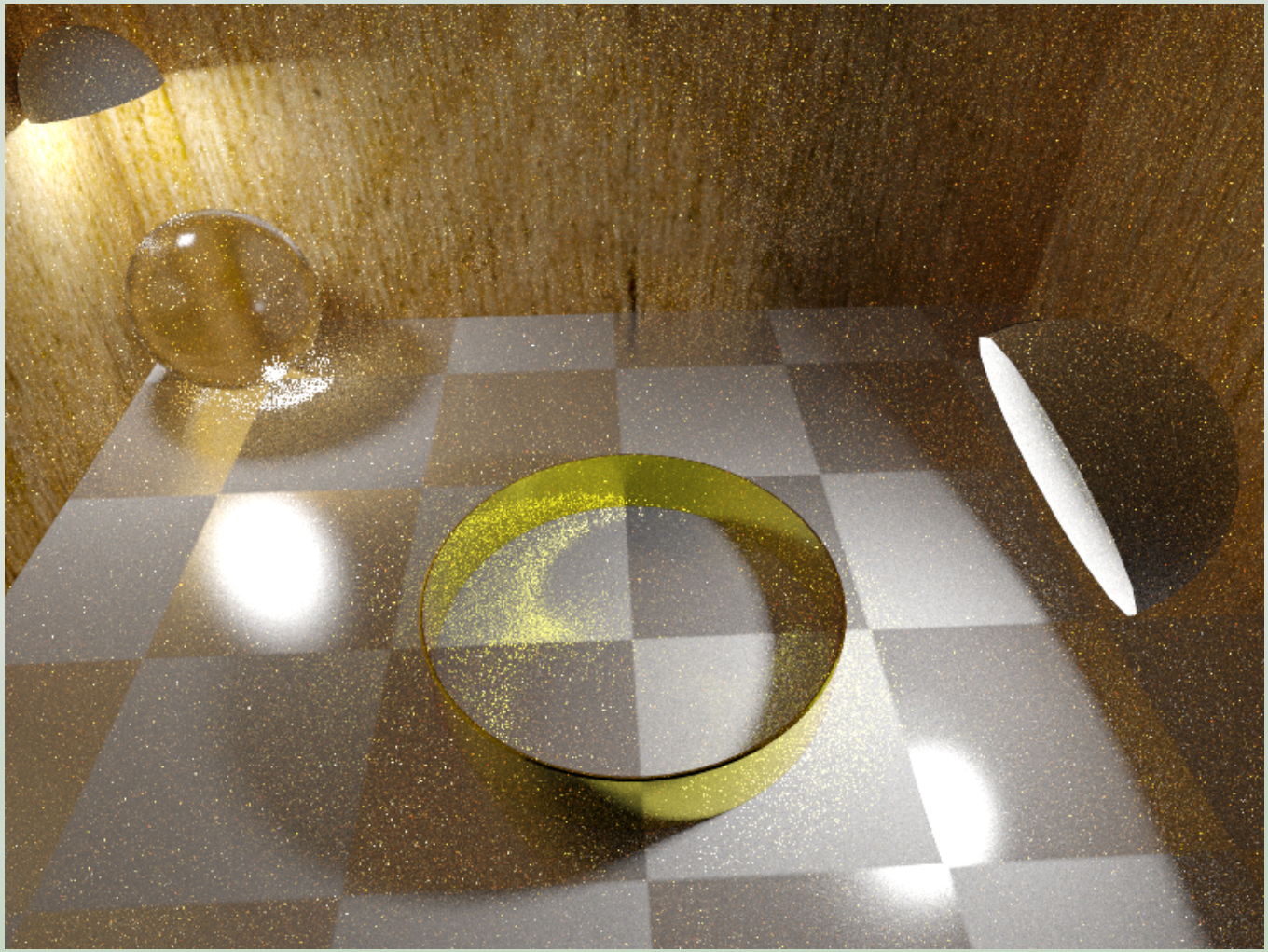
before

after

$N_c = 10, d = 20, r = 1.2, 1600 \times 900$

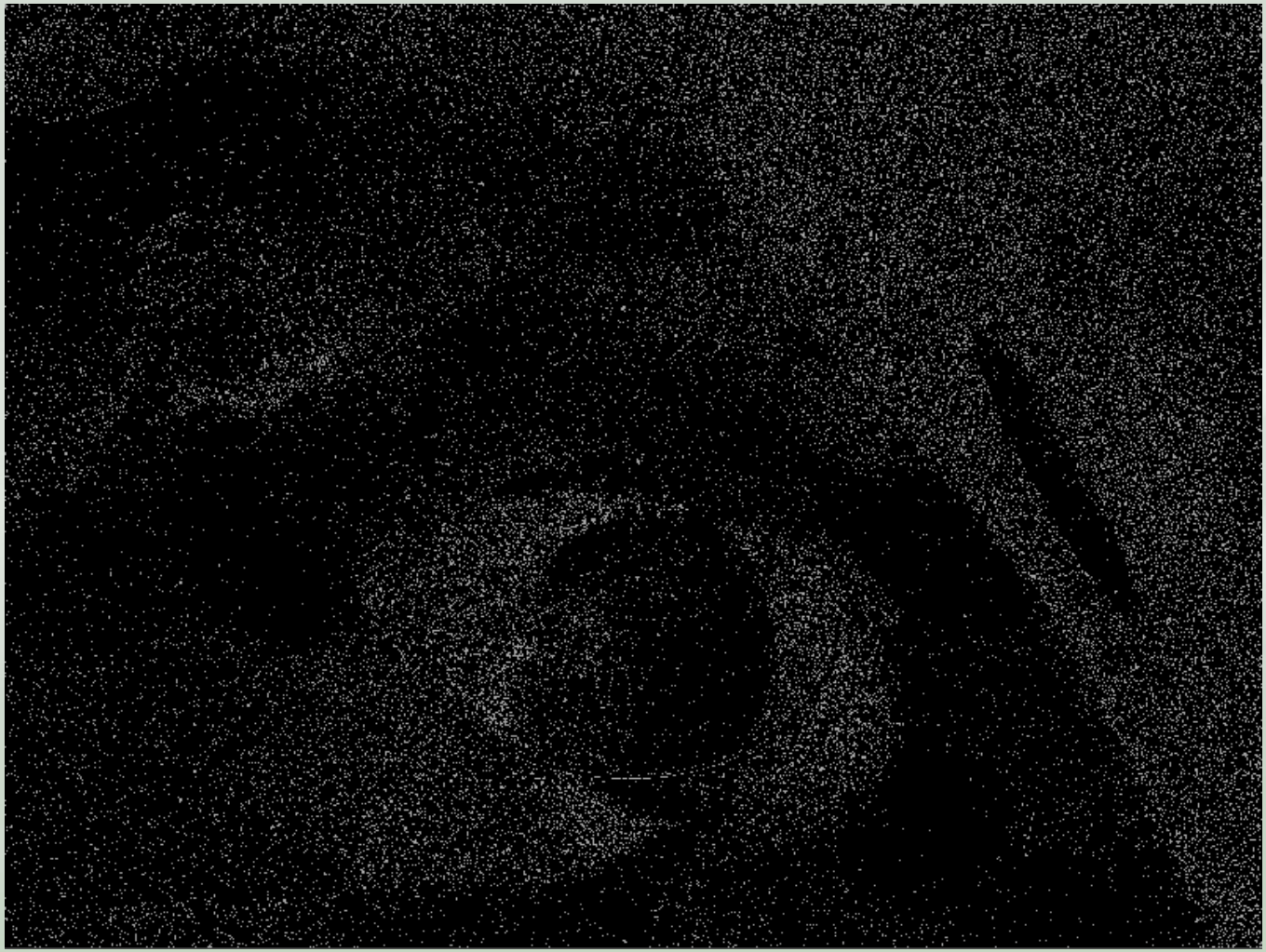
processing time : 0.2s on a core i7

Real world case #1 (path-tracing)



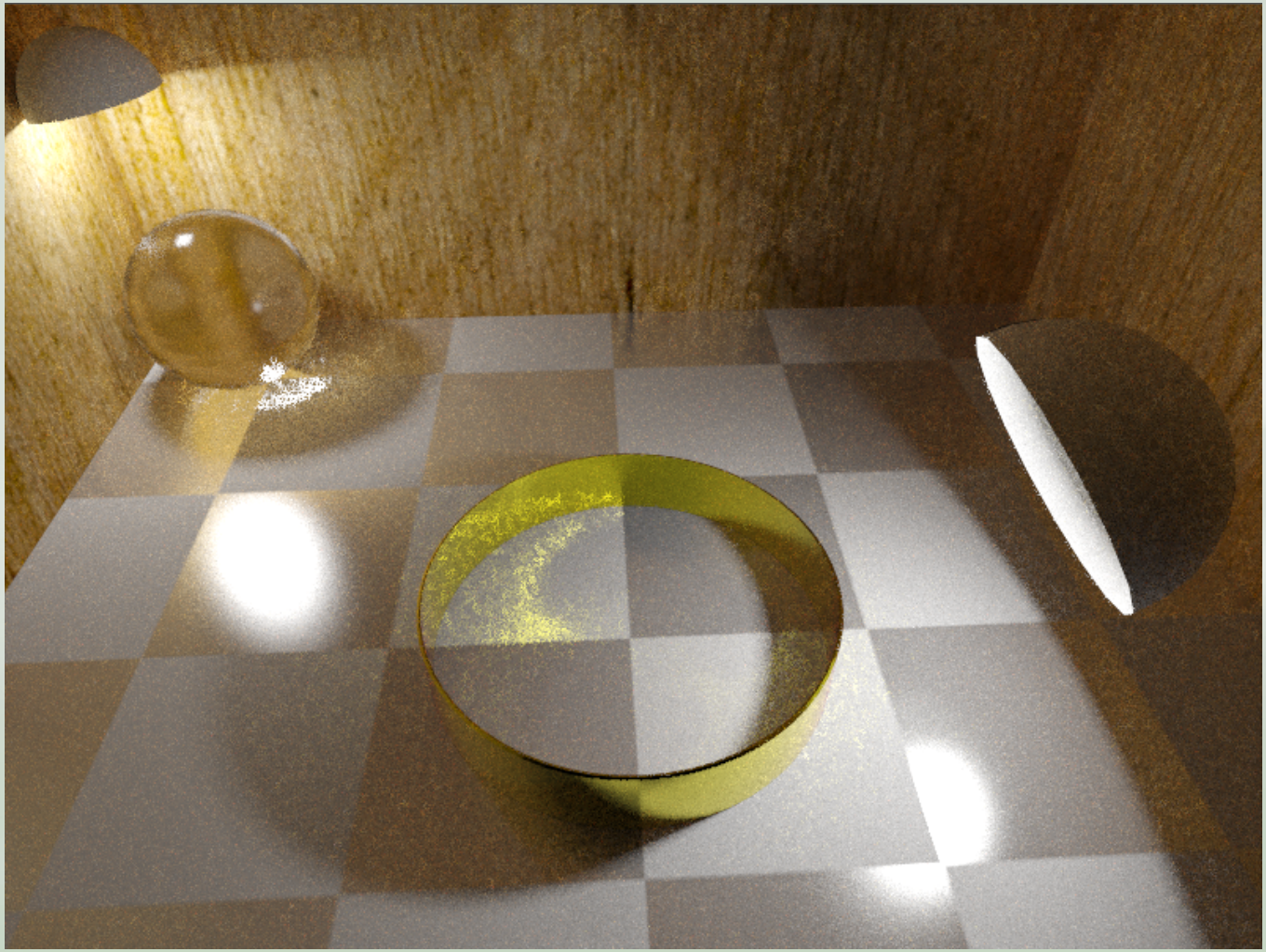
800 × 600

Real world case #1 : detected speckles



$N_c = 10, d = 20, r = 1.2, 800 \times 600$

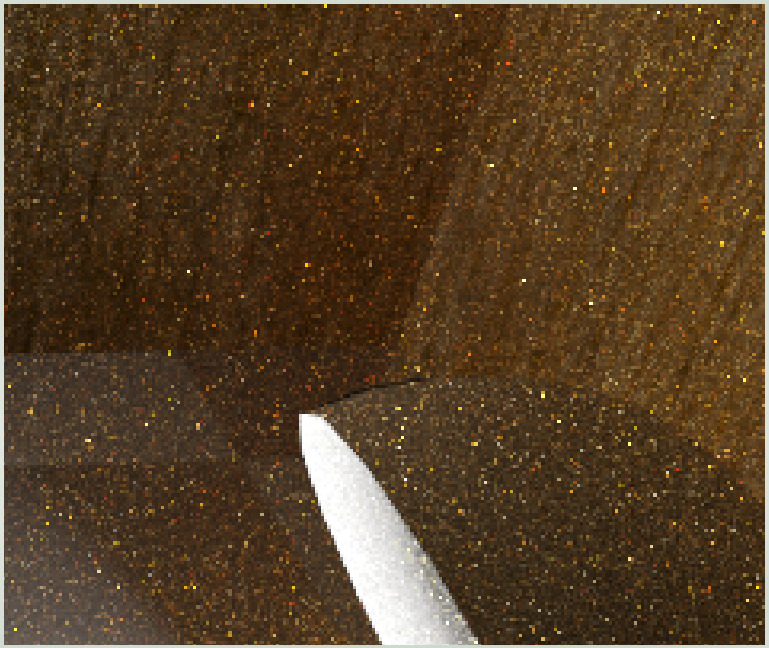
Real world case #1 (path-tracing)



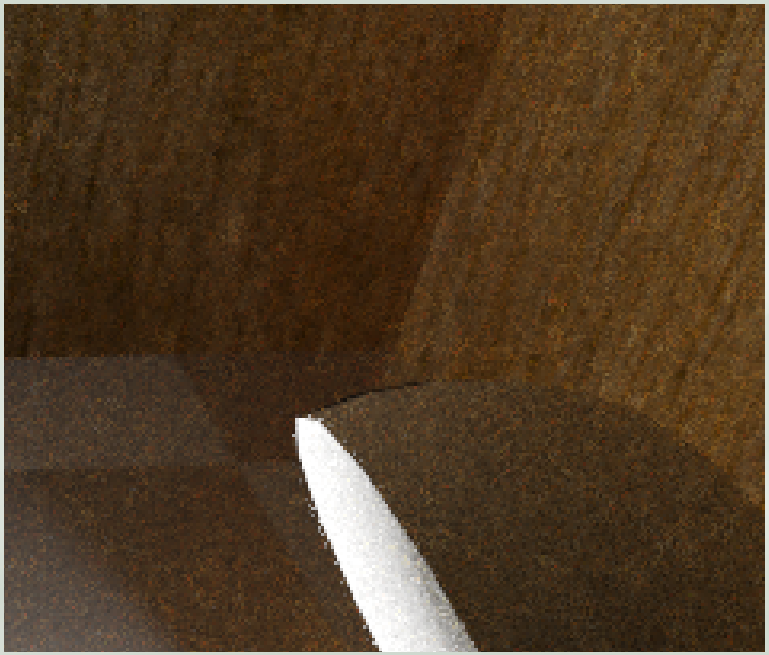
$N_c = 10, d = 20, r = 1.2, 800 \times 600$

processing time : 0.07s on a core i7

Real world case #1 (path-tracing)



before

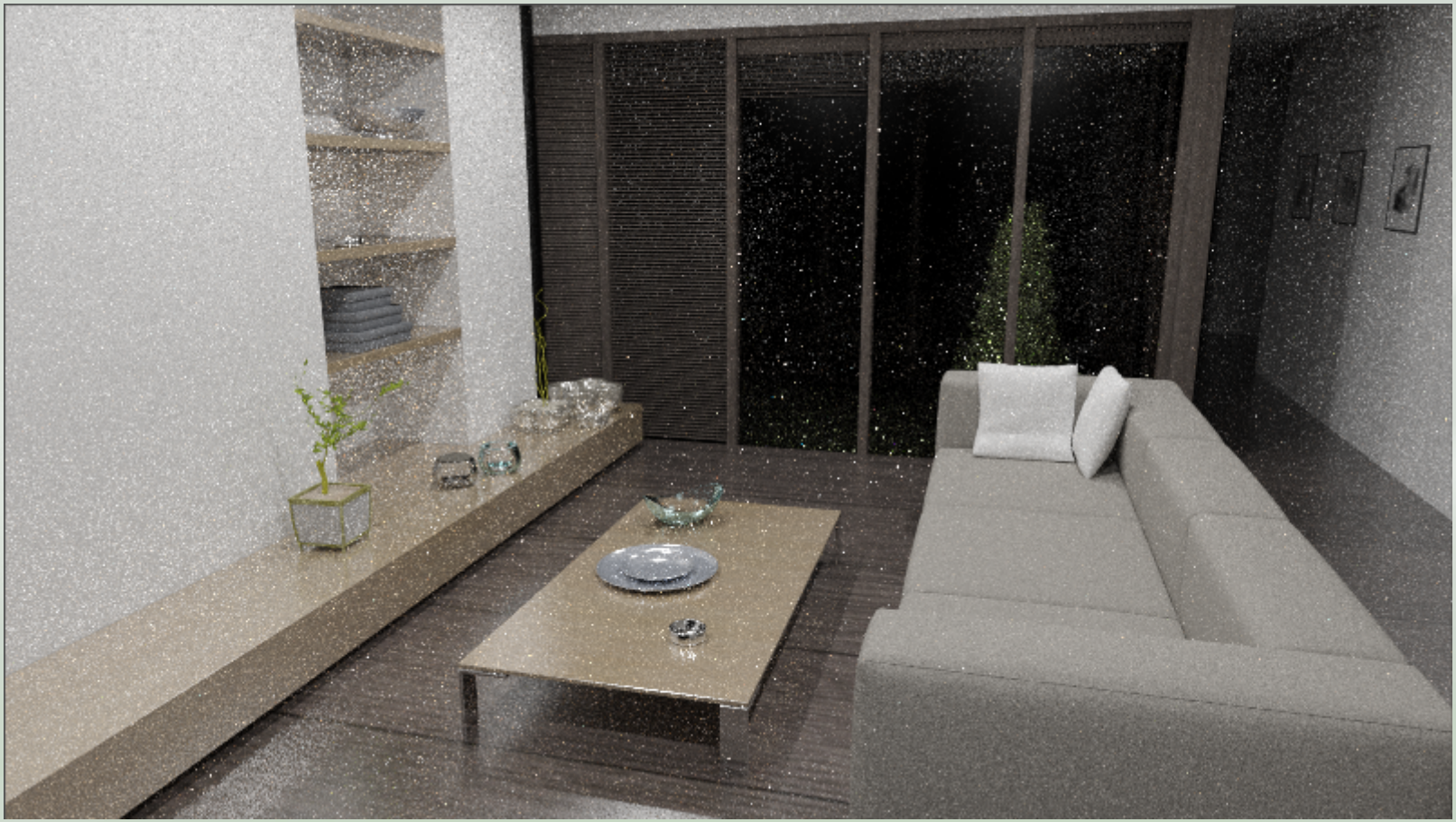


after

$N_c = 10, d = 20, r = 1.2, 800 \times 600$

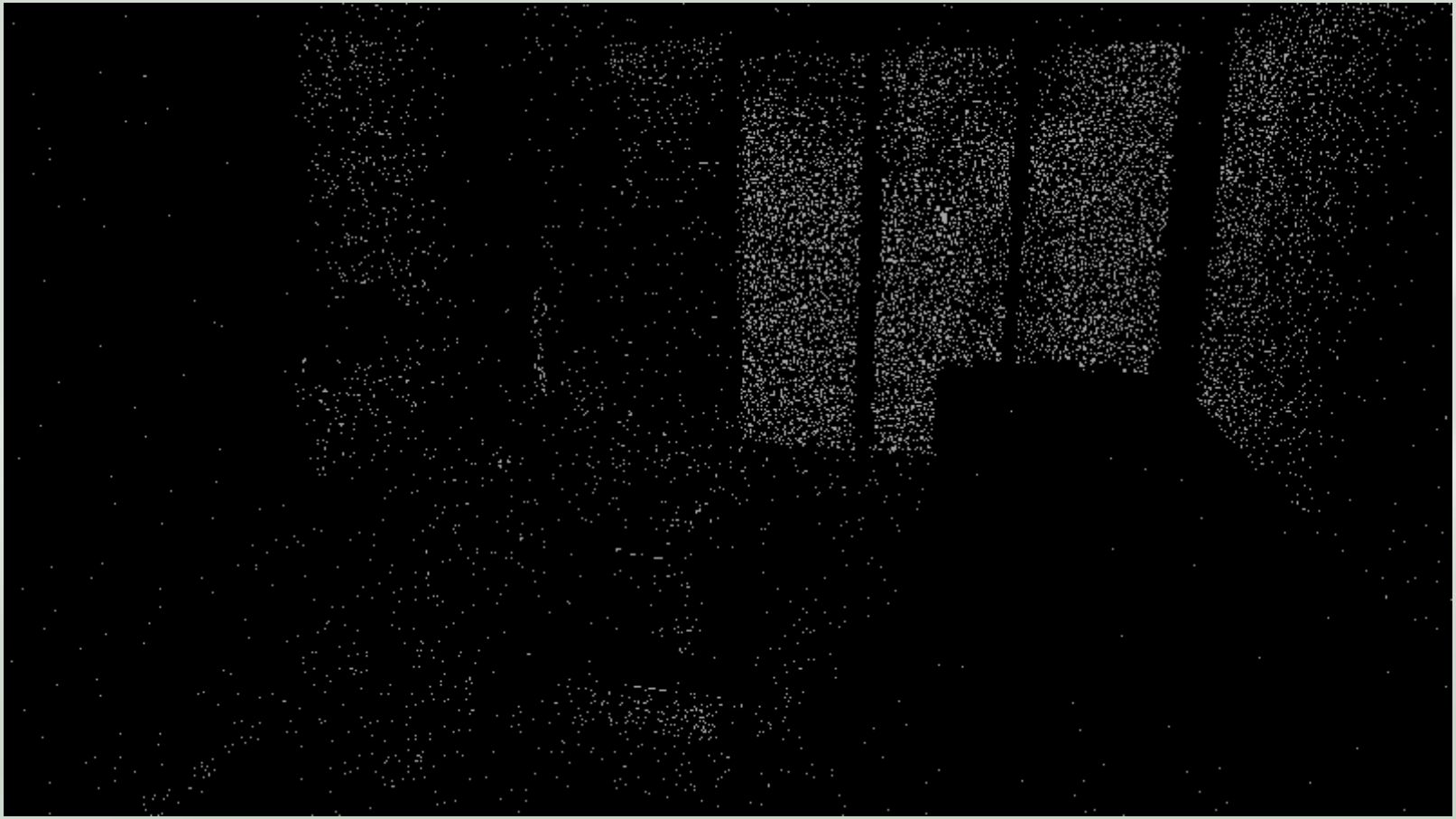
processing time : 0.07s on a core i7

Real world case #2 (path-tracing)



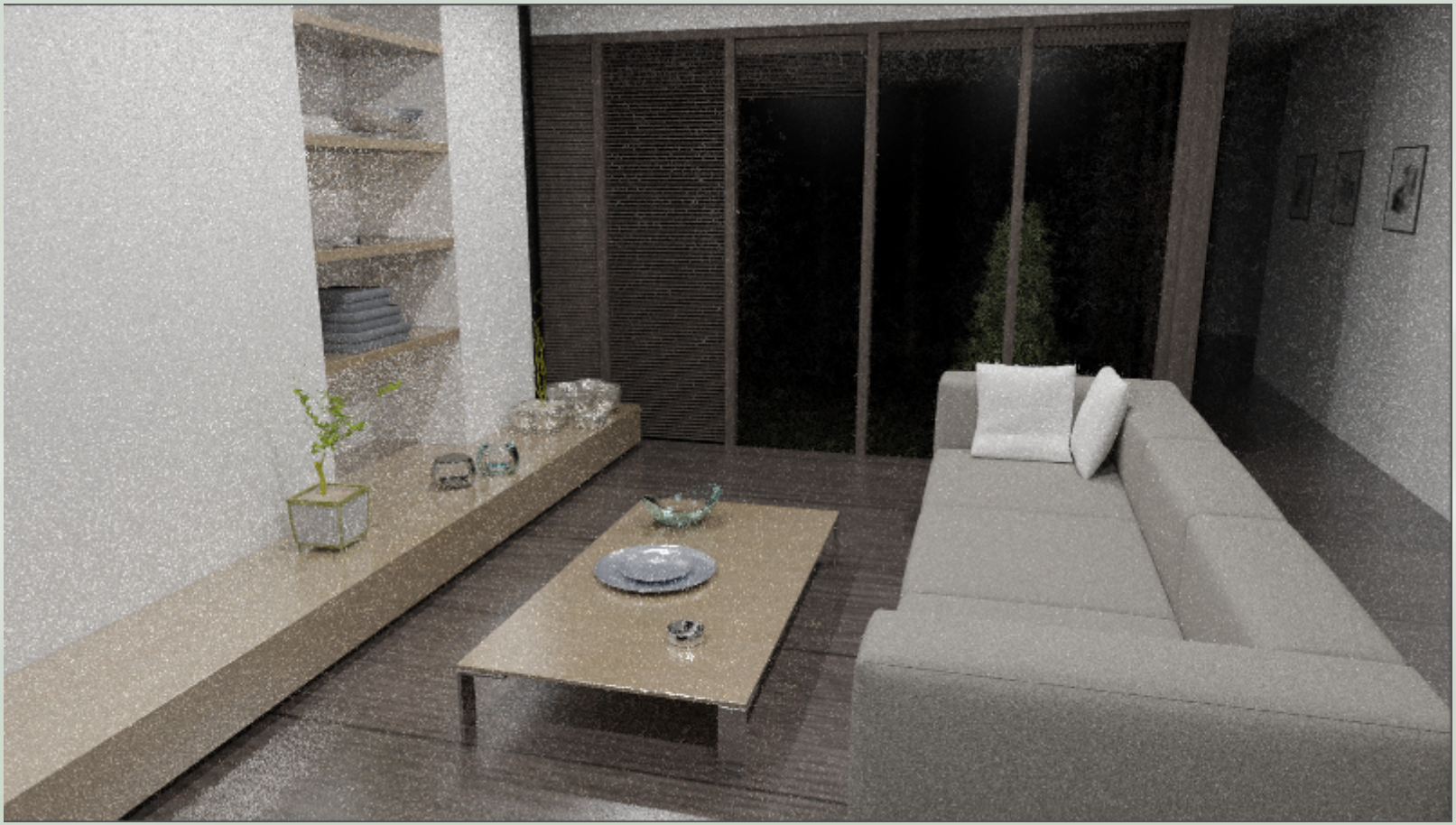
800 × 450

Real world case #2 : detected speckles



$N_c = 10, d = 20, r = 1.5, 800 \times 450$

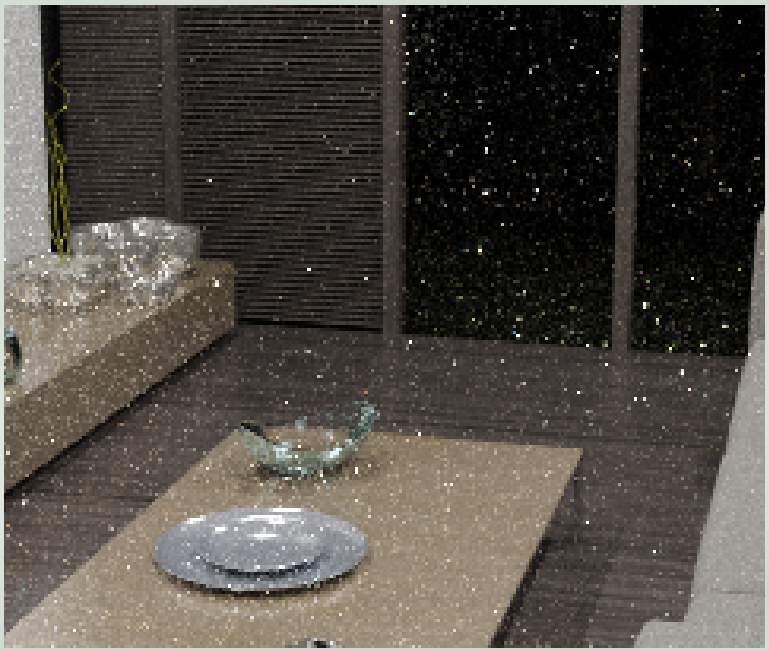
Real world case #2 (path-tracing)

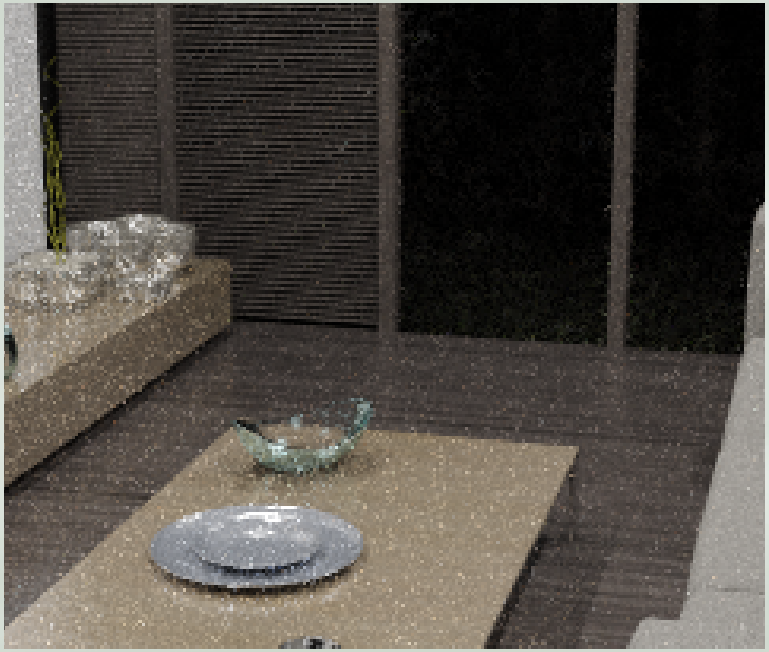


$N_c = 10, d = 20, r = 1.5, 800 \times 450$

processing time : 0.05s on a core i7

Real world case #2 (path-tracing)





before after

$N_c = 10, d = 20, r = 1.5, 800 \times 450$

processing time : 0.05s on a core i7

Conclusion

Advantages

- Simple and effective method
- Relies on a characterization of non-speckled pixels
- Easily parallelized for efficient implementation

Drawbacks - future works

- Not completely robust (more global characterization ?)
- Reconstruction when all pixels in a neighborhood are speckles ?
- How to set the parameters ? (Automatic computation ?)
- ...and certainly others we are not aware of

That's all, Folks !

Thanks for your attention !