Project Theme : Monte Carlo Denoising

Aim :

The goal of this project is to implement the adaptive sampling algorithm described in the paper ***"Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms"***- *Kalantari Et. al.* The paper also details an image denoising strategy which is employed once the image is rendered as a post processing step. For this project, I focussed on implementing the adaptive sampling strategy detailed in the paper.

For evaluating the performance of the algorithm, I rendered the scenes we have already used with some changes in a few of them. I also extended my path tracer to implement Depth of Field so as to evaluate the algorithm for such scenes too.

Adaptive Sampling :

The algorithm proceeds iteratively using a portion of the samples in each iteration based on an *importance map* calculated from the intermediate rendered image. Initially, 2 samples are shot through each pixel. Using this rendered image, an adaptive sampling strategy is employed to distribute *1/7*th of the remaining samples across the image. This rendered image is then used to distribute *1/3*rd of the remaining samples and then in the final iteration all the remaining samples are distributed.

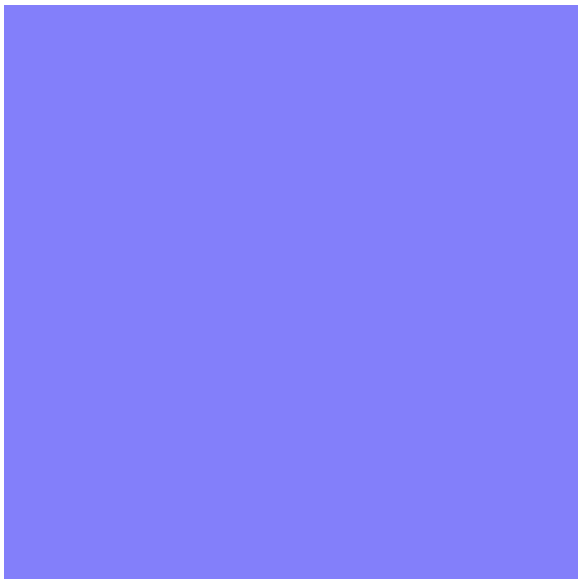
Image is divided into non-overlapping blocks of 8x8 and an *importance* is calculated for each block as $\mathcal{M}_p = \sqrt{\gamma_{s(w(p))}\tilde{\sigma}^2_{w(p)}}$. where gamma is the contrast map and sigma is the Noise map.

Contrast Map : Contrast map [Hachisuka et al.] employs a simple averaging of all samples over each color channel in a window of size 8x8. This was pretty straight-forward to calculate.

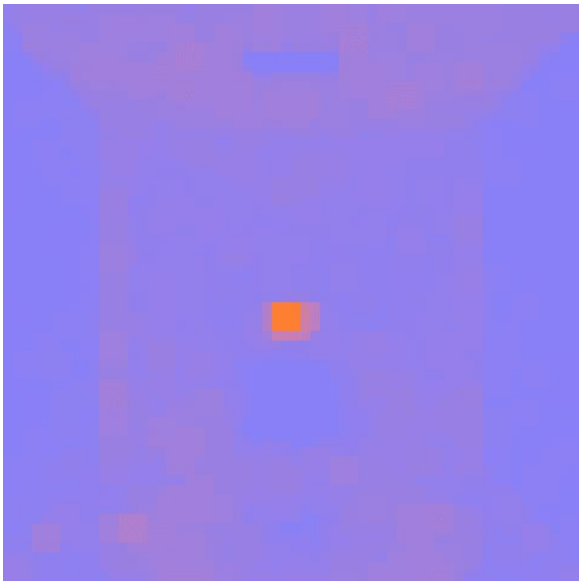
Noise Map : This is essentially the Median Absolute Deviation [Donoho et al.] and is calculated as $\tilde{\sigma}_{w(p)} = \frac{\text{median}(|\mathcal{D}_0^1|)}{0.6745}$ where D_0 is the diagonal detail coefficient of the finest level of wavelet transform performed on each block (for each channel) containing the pixel color across samples (averaged).

The samples are divided (in ratio) according to the importance map calculated. [Equal samples for all pixels in the same block]. Following is the algorithm in action with sample distribution dumped as heat map after each iteration. [Blue least, red highest].

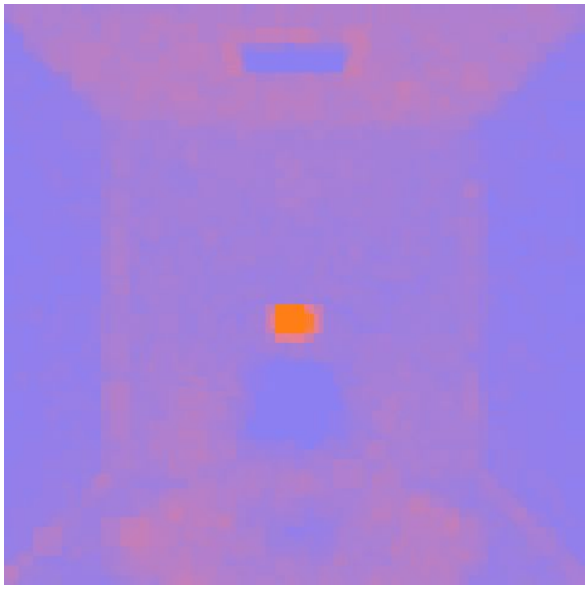
Iteration 0 : [2 samples per pixel]



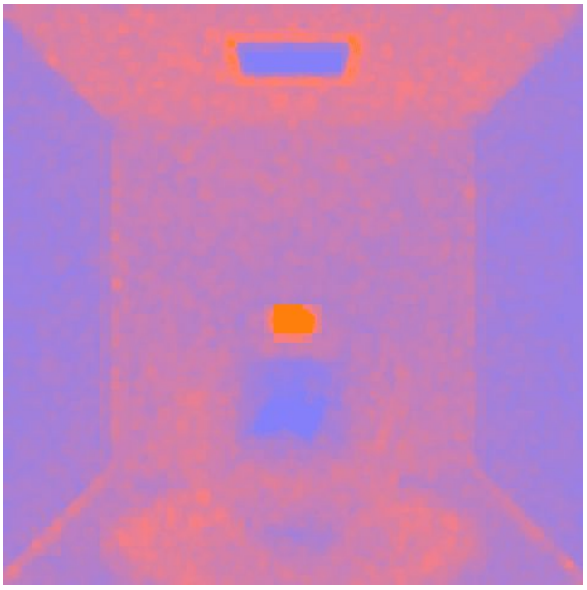
Iteration 1 [Adaptively sampled]



Iteration 2 [Adaptively Sampled]

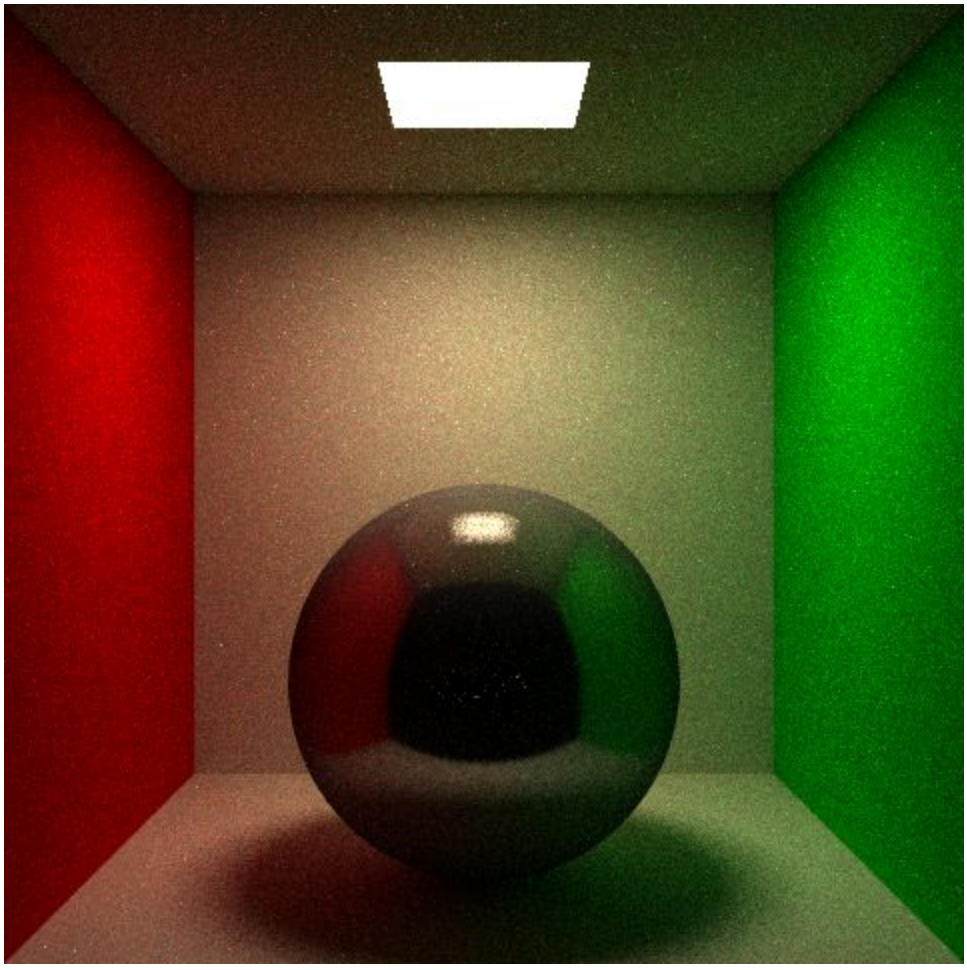


Iteration 3 [Adaptively Sampled]



The algorithm reserves more samples for the specular highlight of the sphere and its shadow and relatively fewer for the side walls.

Rendered Image [Total samples 64 * 480 * 480]:

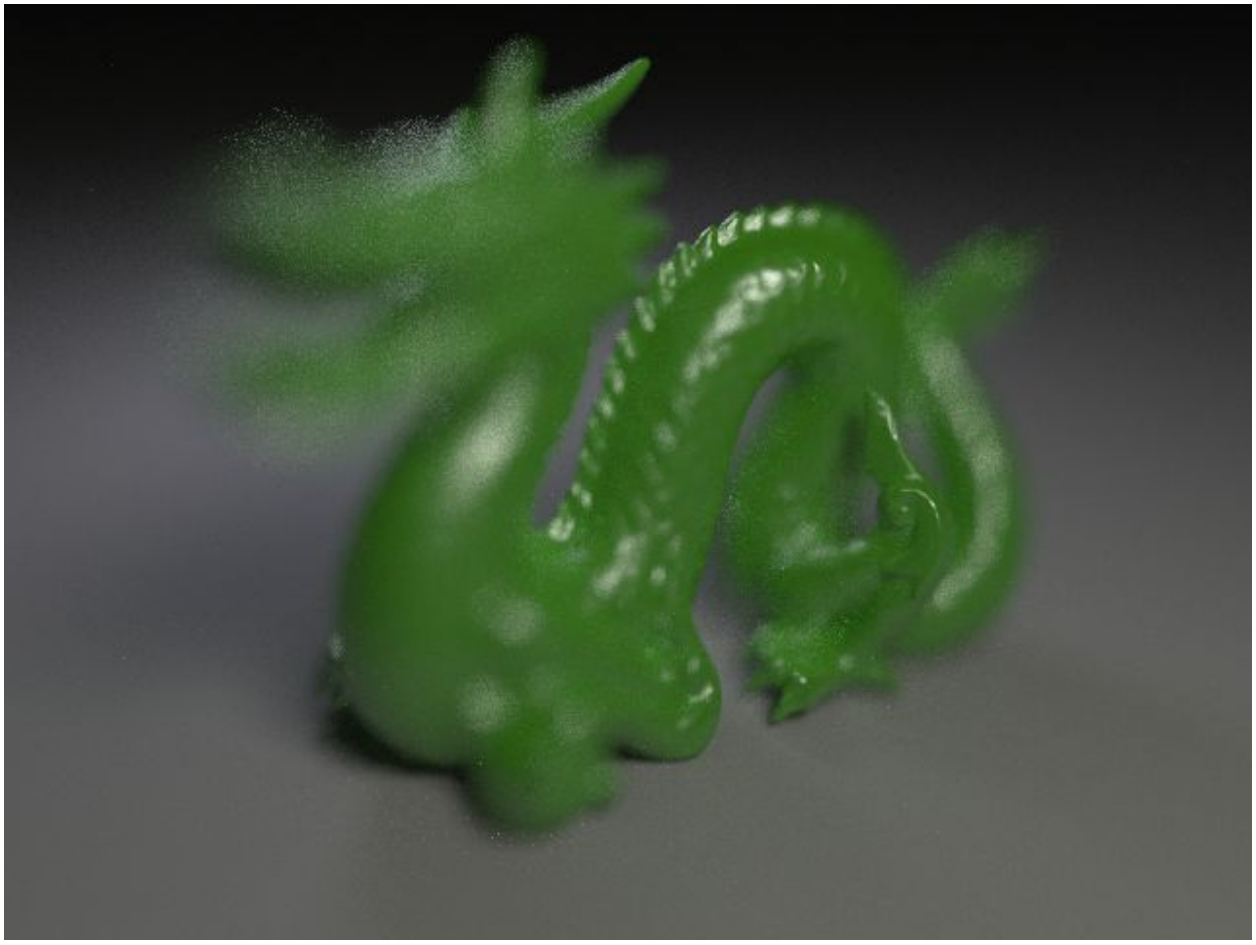


CornellMIS.test rendered with adaptive sampling.

Depth of Field :

Implementing DOF effect was pretty straightforward. I extended my path tracer to work with lens (circular) parameters: aperture and focus. I followed this post for implementing DOF. Following is a result:

Dragon.test rendered with focal length: 18 , aperture : 3



Results :

The following pages show some of the scenes rendered using adaptive sampling. Some of the scenes have been changed a little to better bring out the results of the algorithm. All the changes have been in the viewing angle or lens. Rest remains consistent with the test files provided.

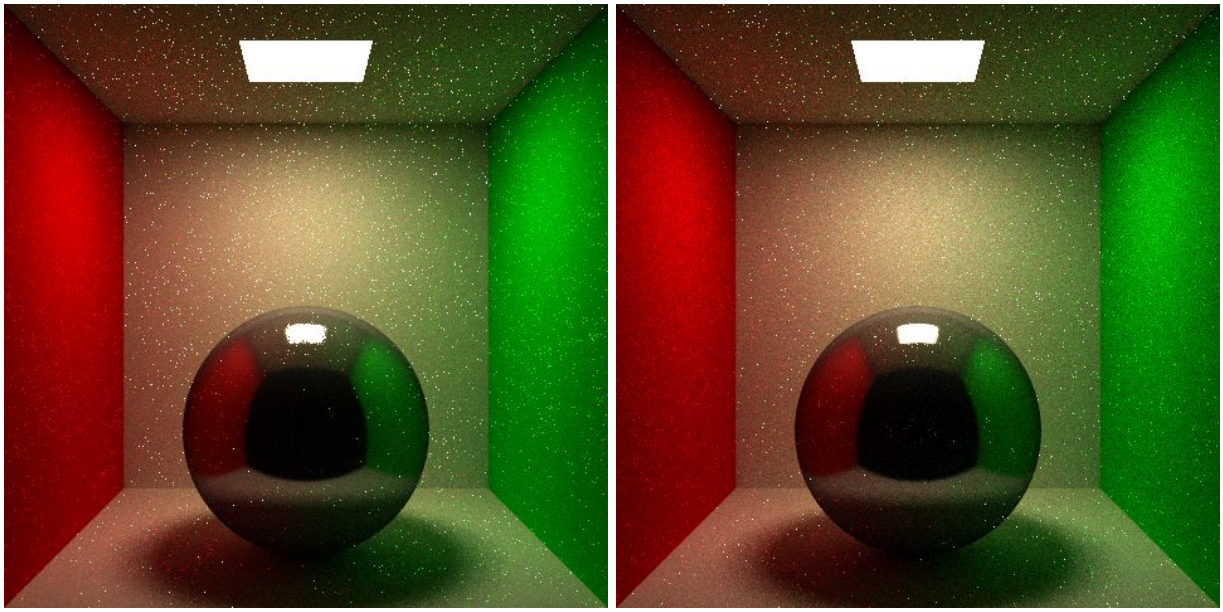
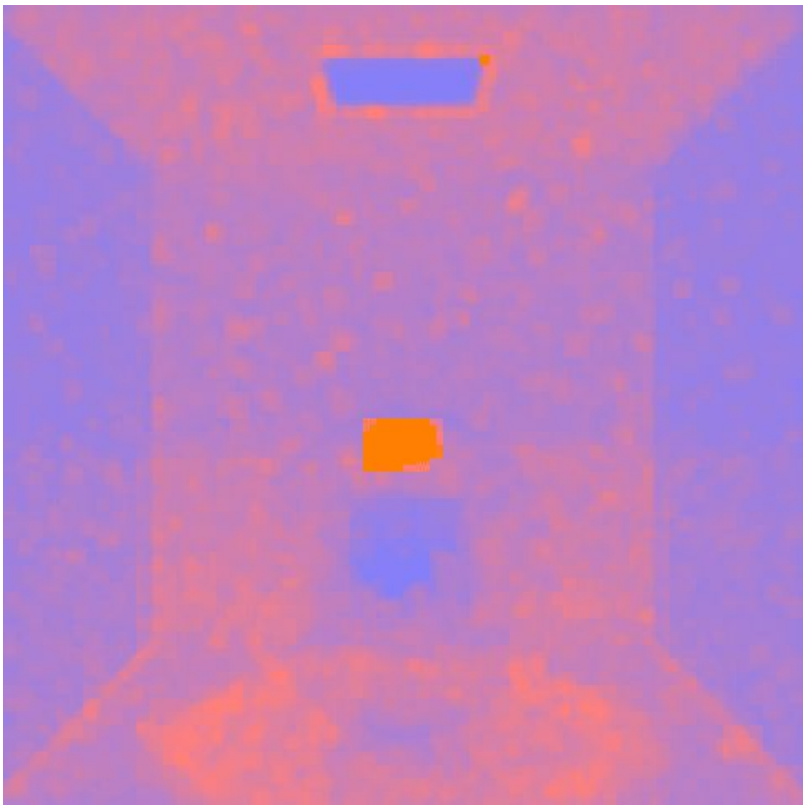


Image on the left is cornellBRDF.test rendered and on the right is cornellBRDF.test rendered with adaptive sampling. Notice a finer highlight on sphere



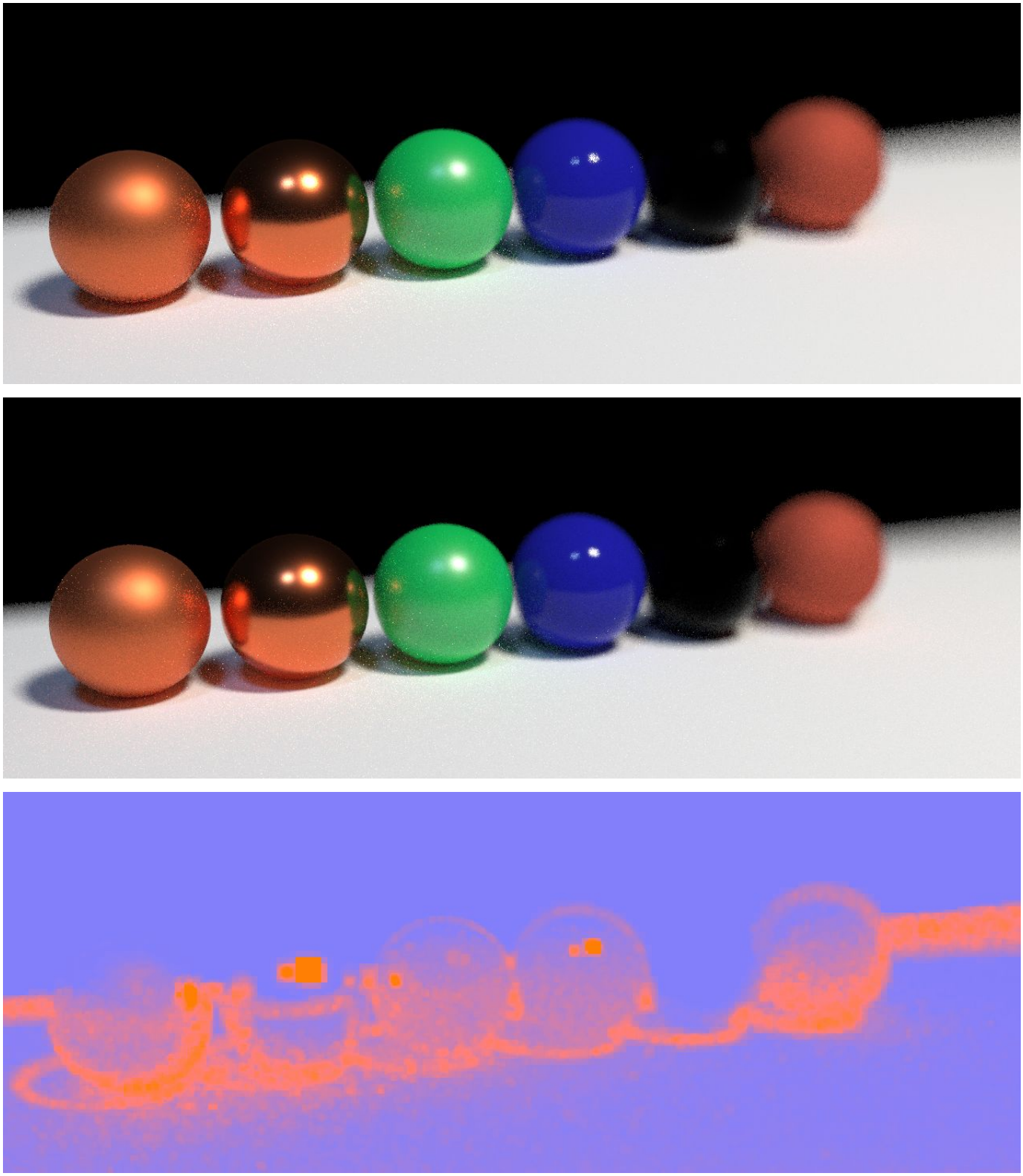
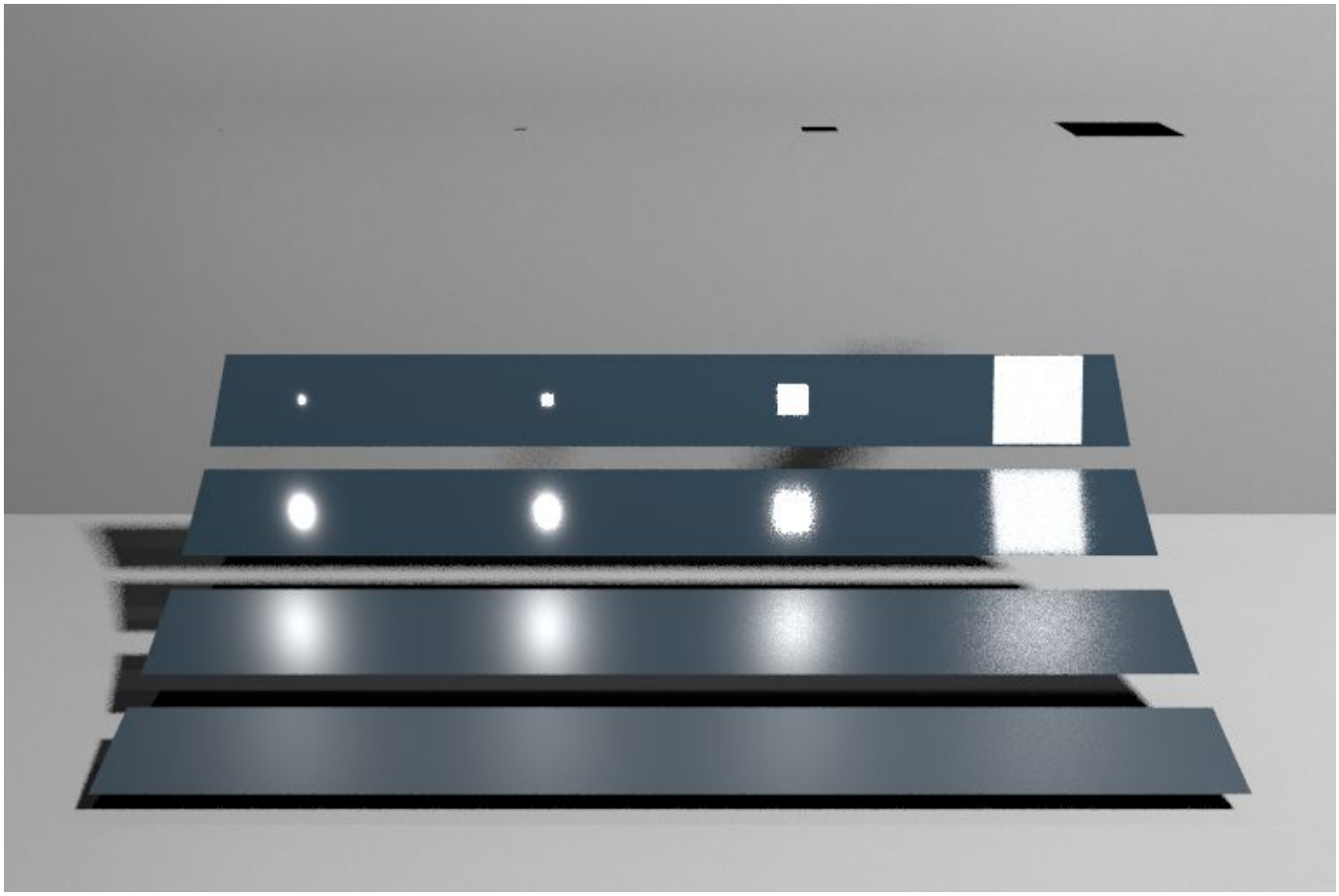
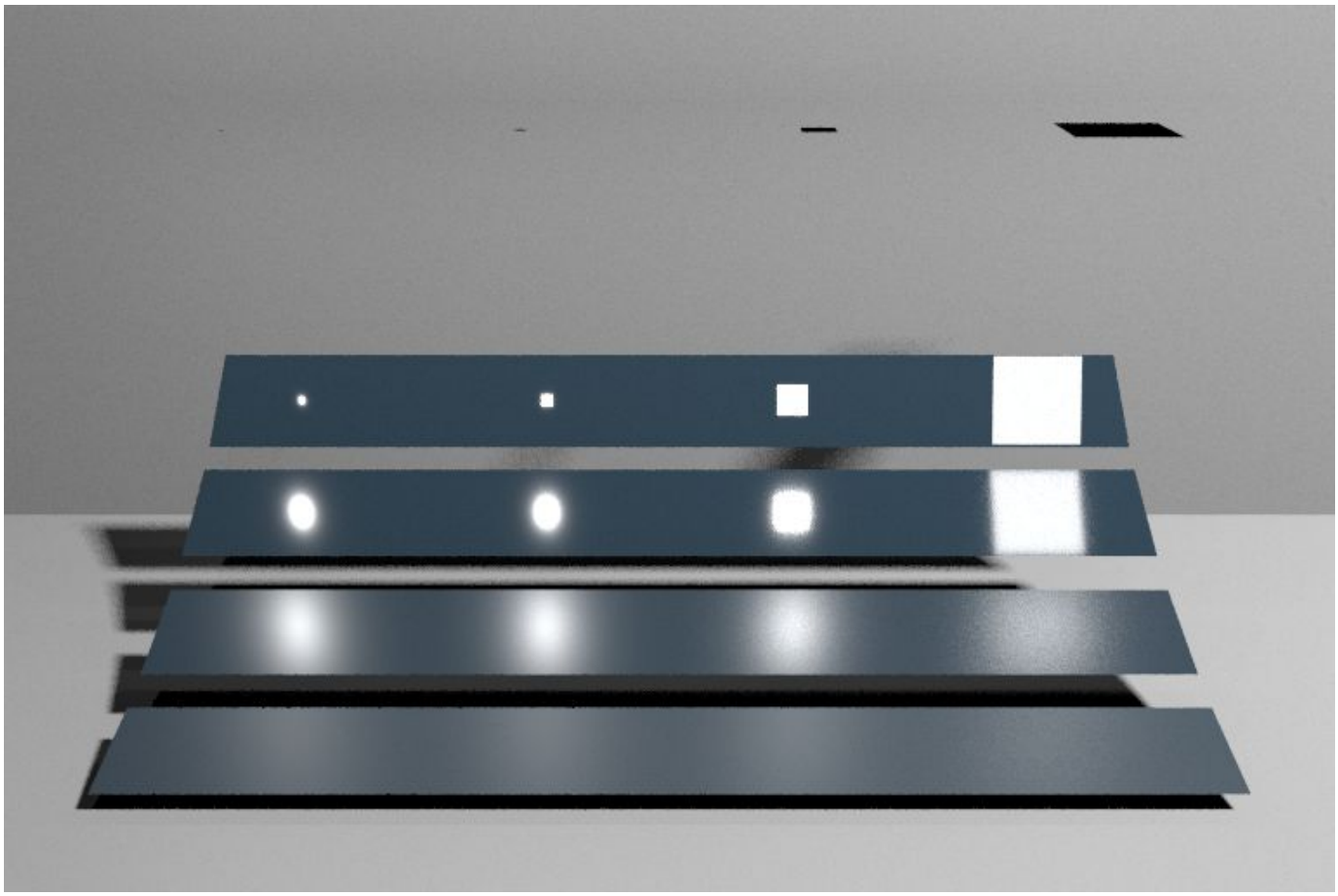
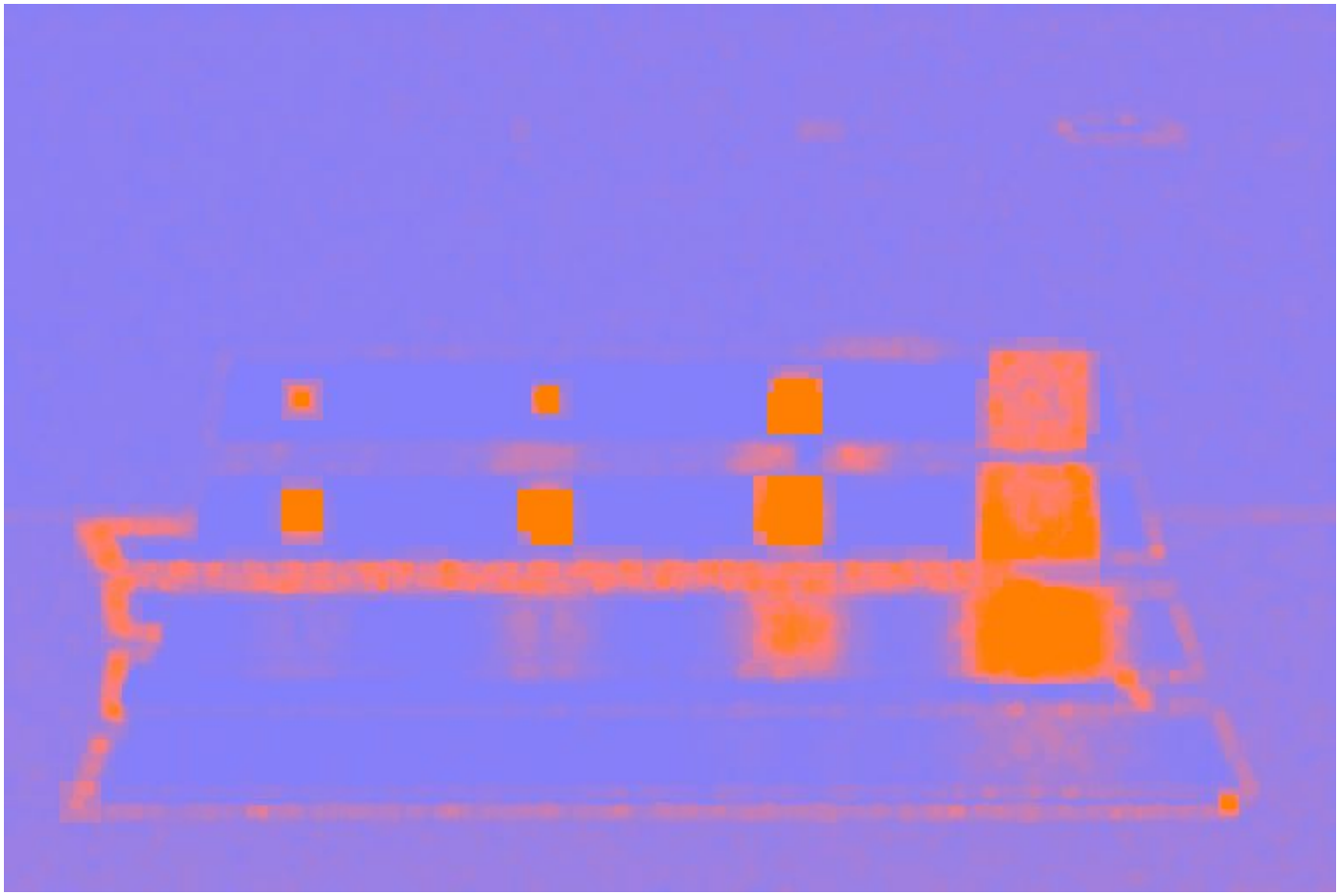


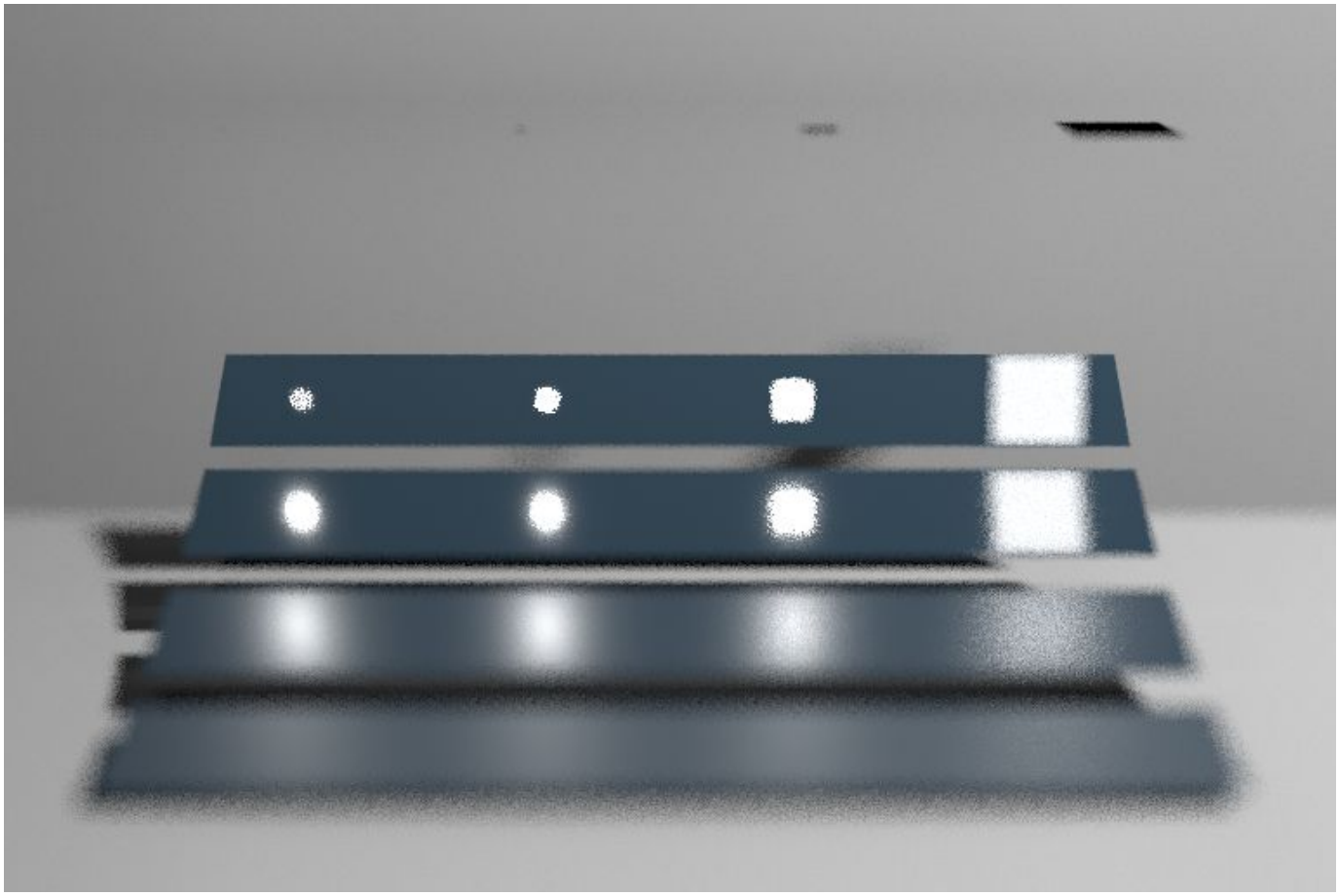
Image on the top is ggx.test rendered with a change in camera position and lens (15, 0.8) [no back wall]
Image in the middle is rendered with adaptive sampling. Notice the much smoother boundary of the first sphere and smooth boundaries of the out-of-focus sphere (last) and wall.

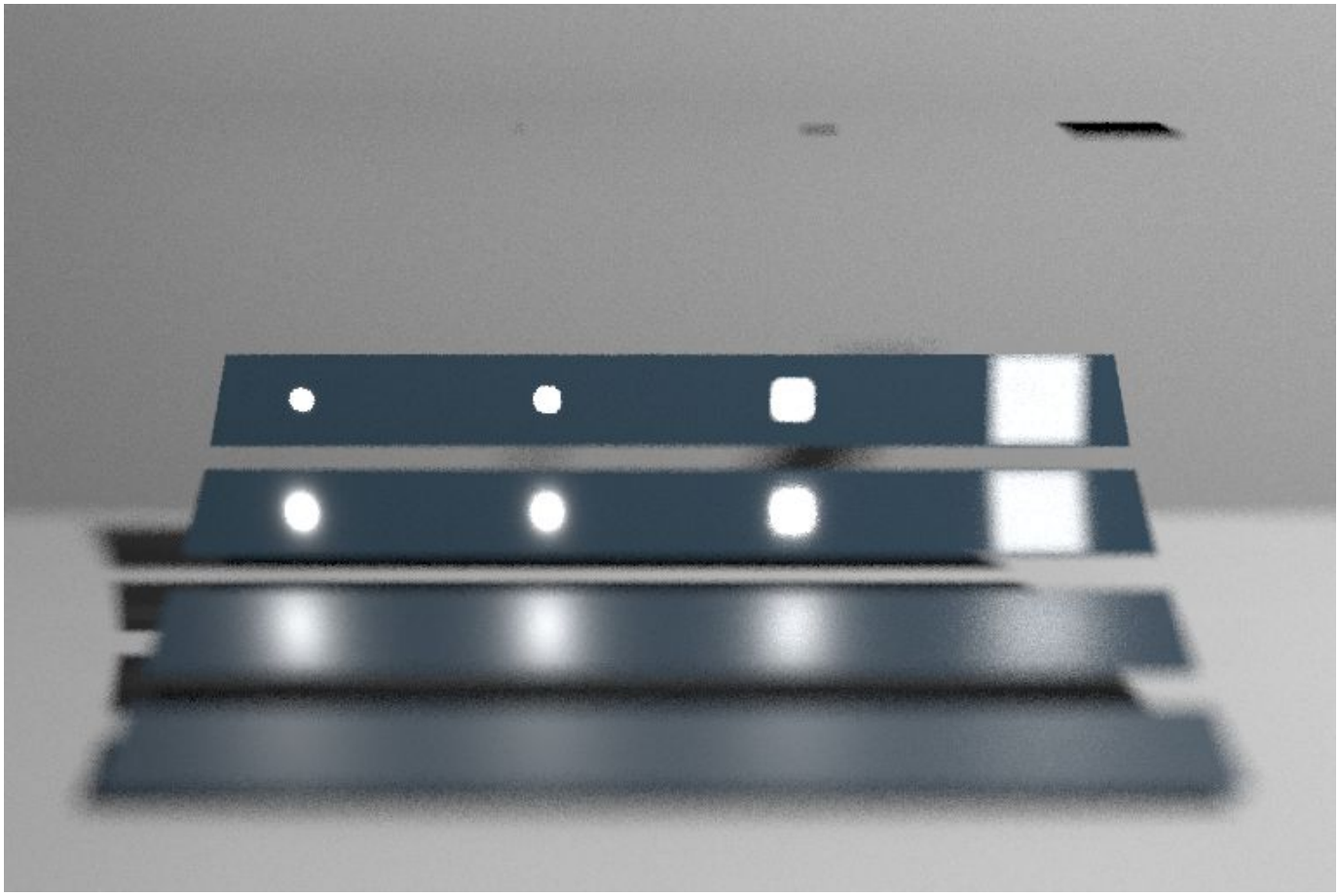


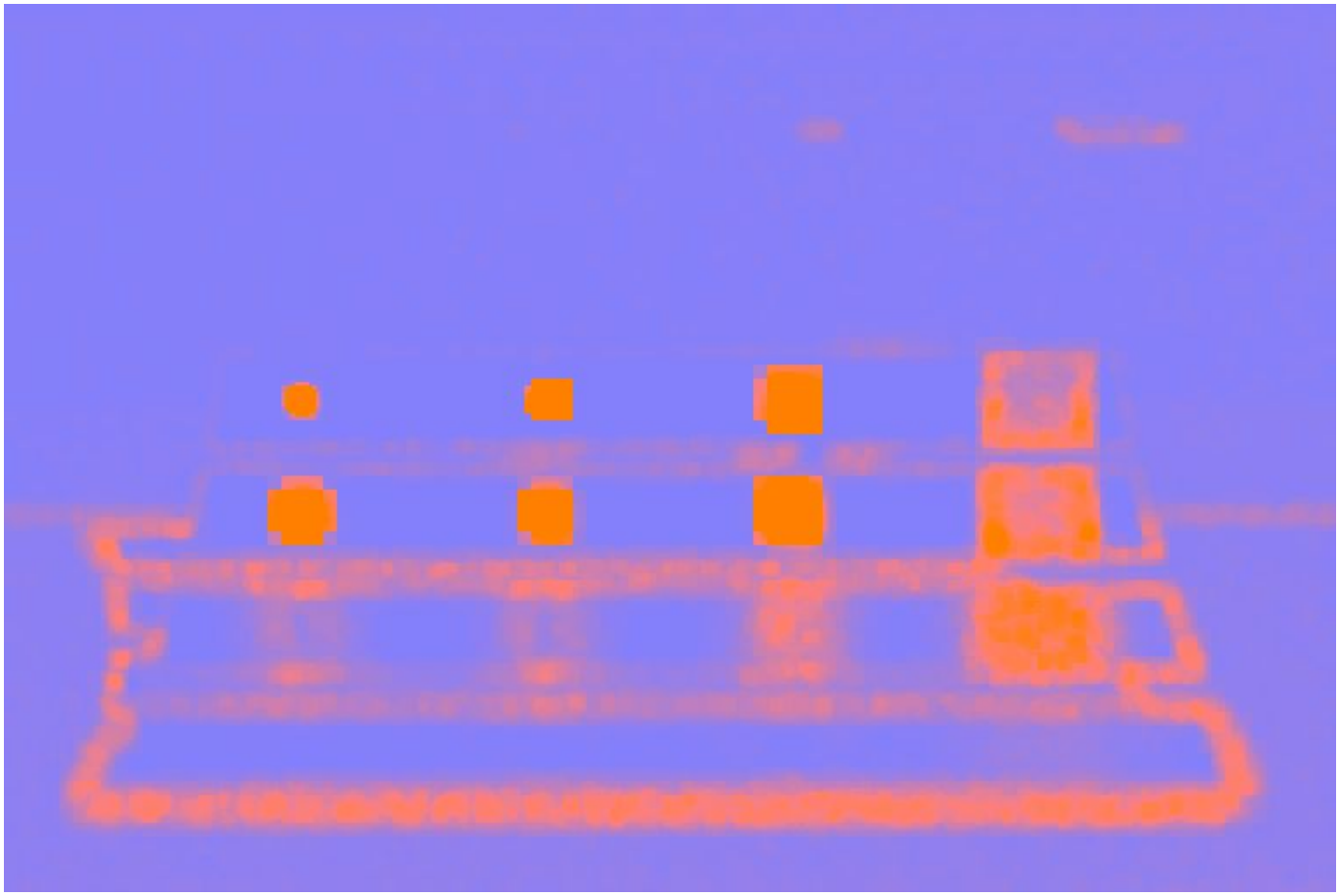




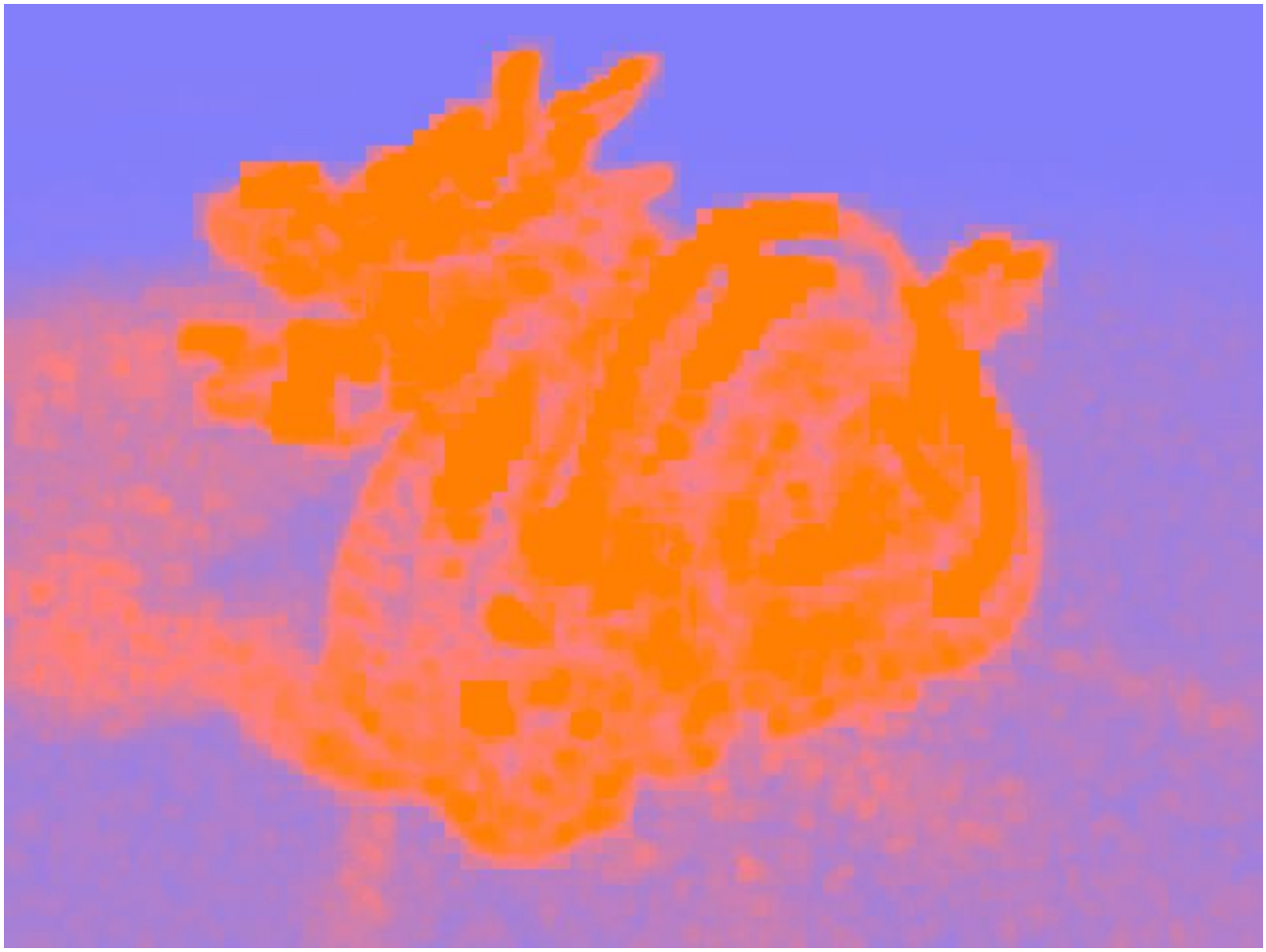
First image is mis.test rendered.
Middle image is rendered using adaptive sampling. Notice the much sharper specular highlights in the first two panes and a much smoother glossy appearance on the third pane. As can be seen from the heat-map, majority of the samples are dedicated to the highlights!



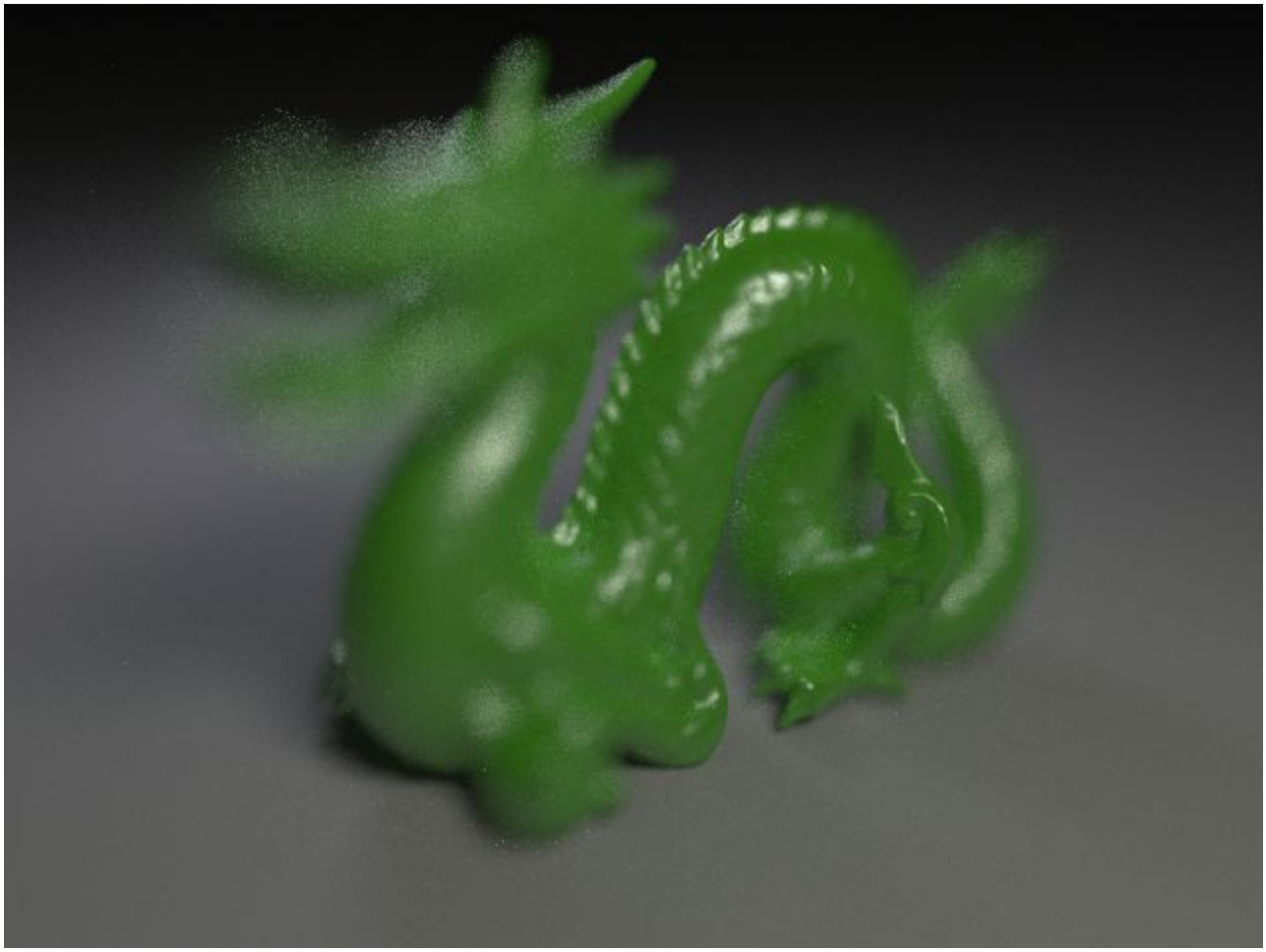


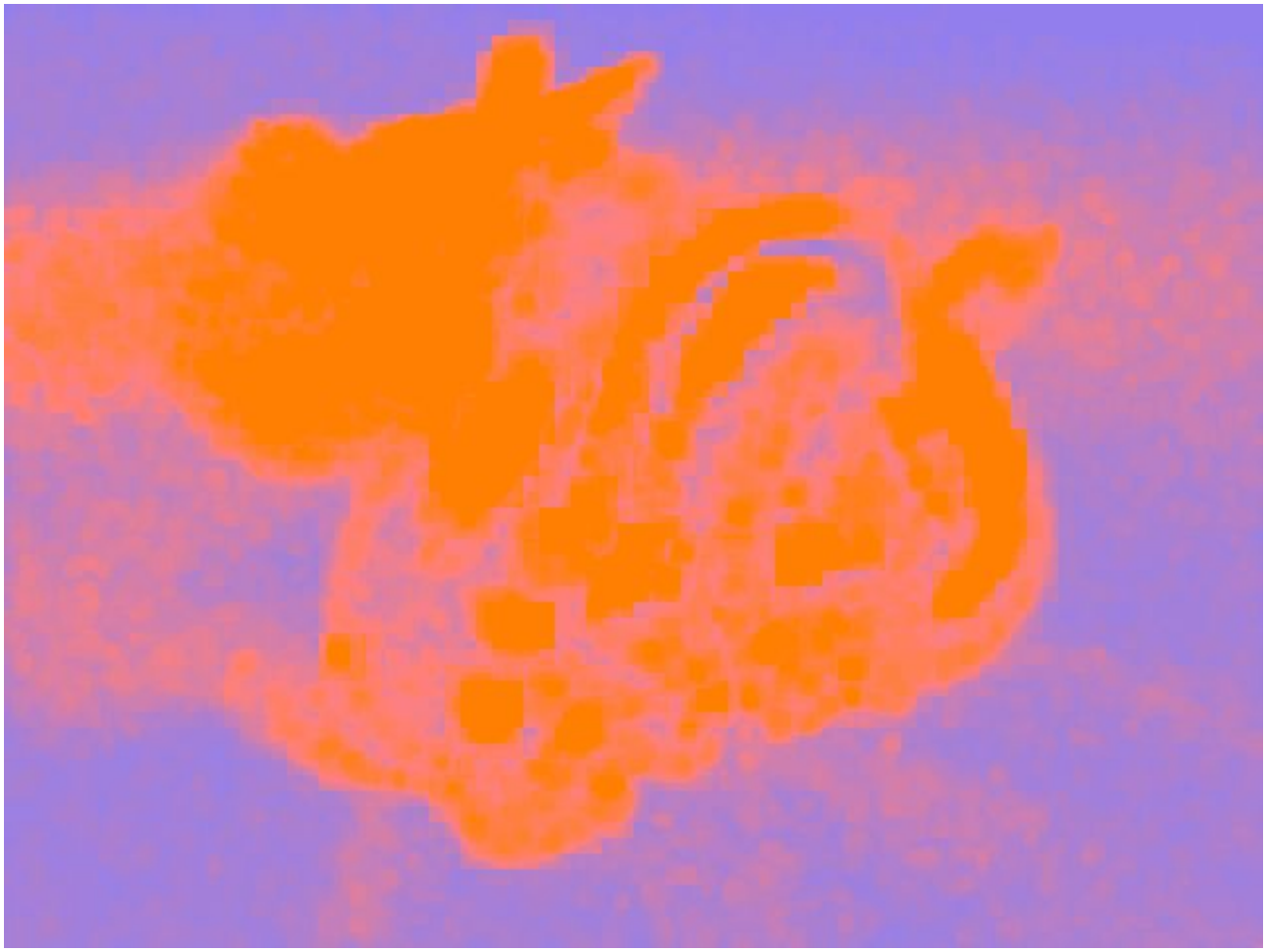


First image is mis.test rendered with a lens (16.3, 0.8).
Second image is rendered using adaptive sampling. Notice the much smoother highlights

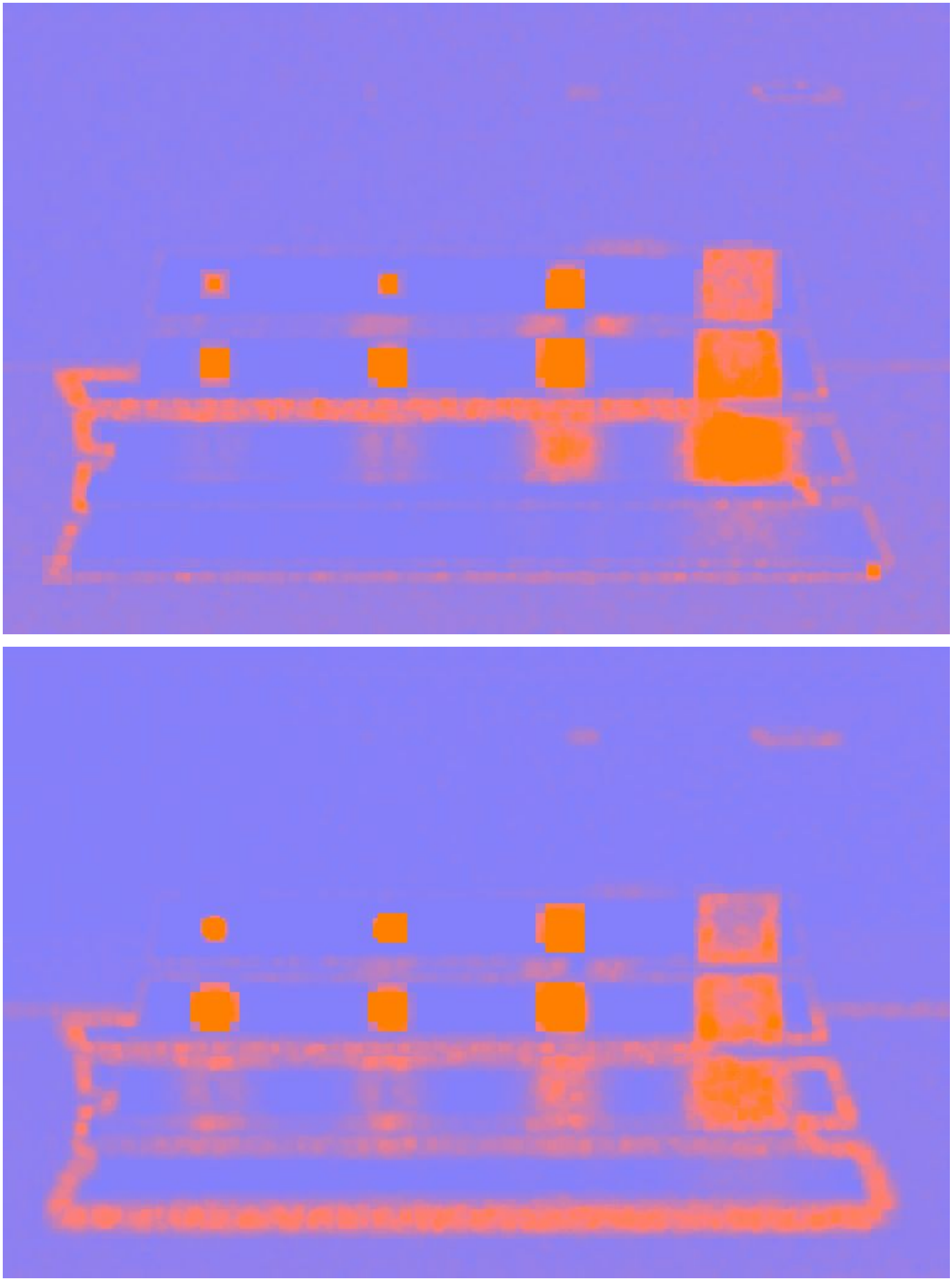


First image is dragon.test rendered.
Second image is rendered using adaptive sampling. It is interesting to note here that since majority of the samples have been thrown at the dragon, the floor appears noisy when compared to the first image.

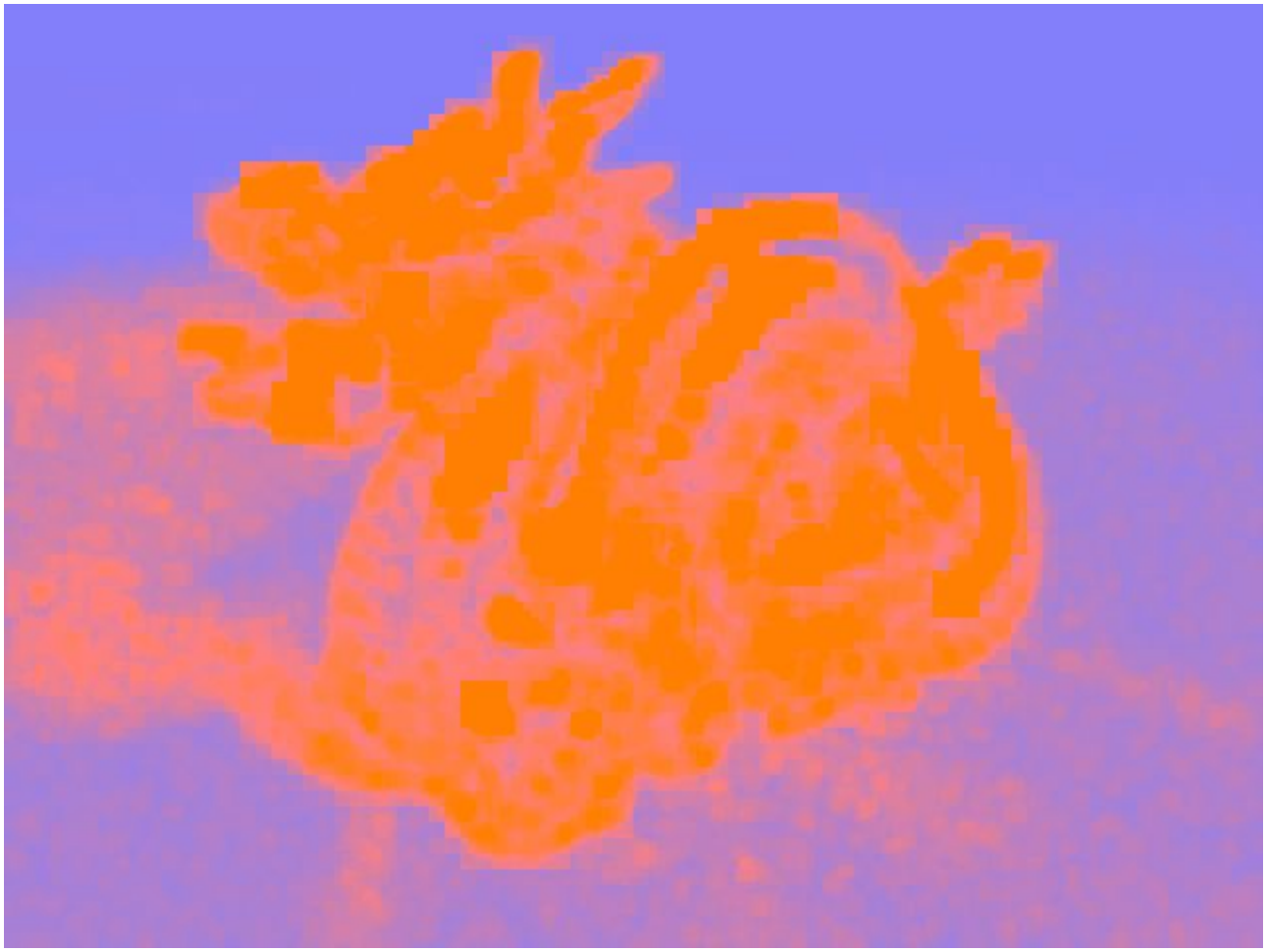


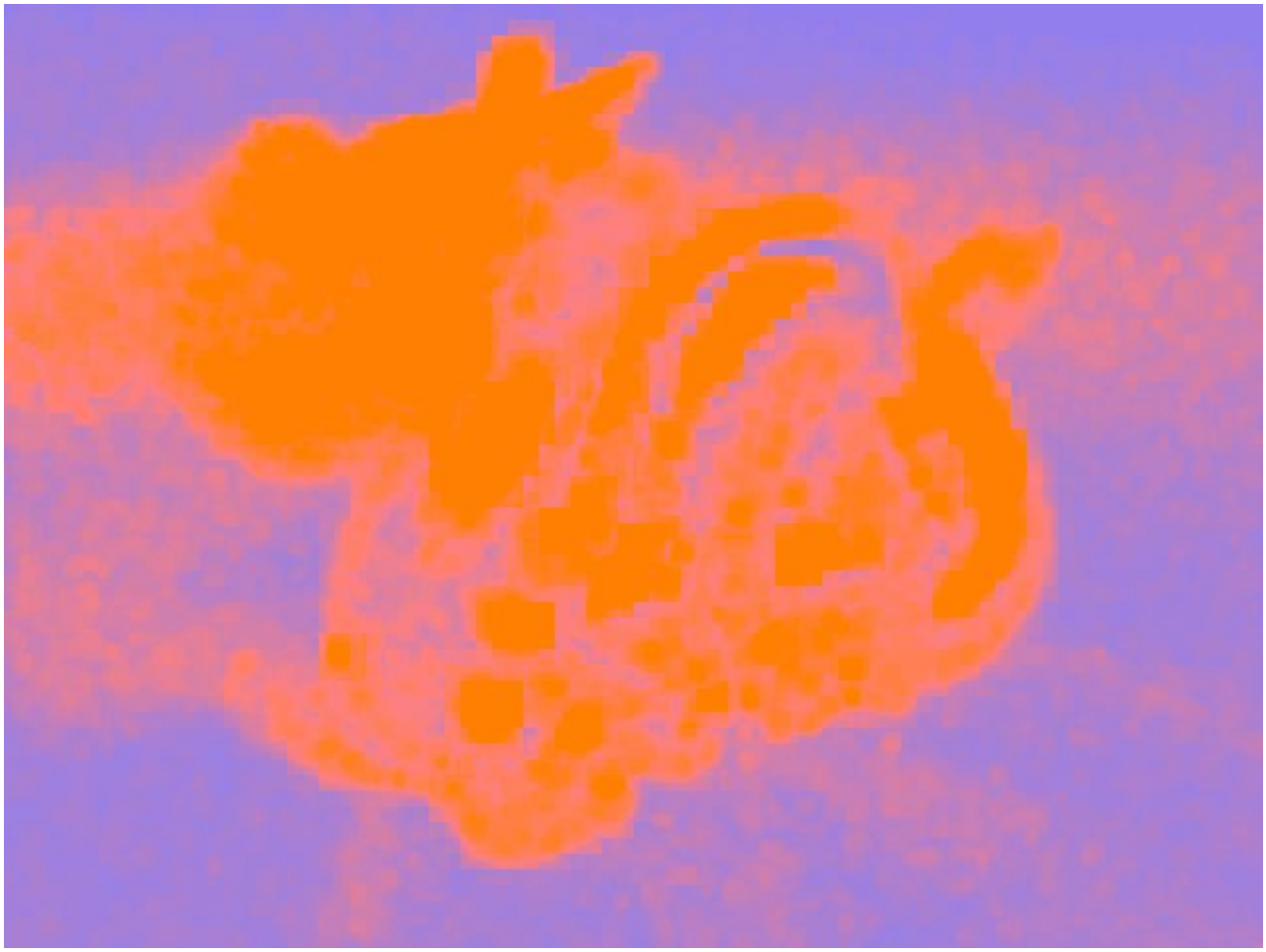


The first image is dragon.test rendered with a lens (18, 3).
The second image is rendered using adaptive sampling. Notice that a huge number of samples are thrown at the head and the tail and they therefore appear smoother and visually appealing.



Comparison of sample distribution for a pinhole vs a lens





Comparison of sample distribution for a pinhole vs a lens

References :

[1] Depth of Field
[2] Removing Noise in MC Rendering
[3] Wavelet Sampling