Image denoising: cours 4

Experimental report

MVA 2021-2022
ENS Paris-Saclay
Elías Masquil
eliasmasquil@gmail.com - elias.masquil@ens-paris-saclay.fr

EPLL: An Image Denoising Method Using a Gaussian Mixture Model Learned on a Large Set of Patches

Let's begin by experimenting with different values for the parameters, and then compare the performance of EPLL against BM3D. For all the experiments we'll consider two settings: small noise (sigma = 10) and high noise (sigma = 50). As suggested, I'm using 50% of the rank of the covariance matrices for performance considerations. Based on the experiments, this algorithm is far slower than others studied during this course.

Figure 1: Original "Traffic" image, used in all the experiments

Small noise image (std=10) Big noise image (std=50)

Figure 2: Noisy images, used in all the experiments

Step size

Augmenting the step size for extracting and aggregating patches causes to have less estimates for each pixel. This significantly reduces the running time of the algorithm.

Small noise

Big step size (s=8) PSNR = 29.80 Small step size (s=3) PSNR = 29.60

Figure 3: Single scale denoised image varying step size (small noise)

In the small noise context, the algorithm does well at both step sizes. Both denoised images look the same. Curiously the one with a bigger step size has a slightly higher PSNR (despite having less estimates per pixel). In this case the best idea is to use the bigger step size because of an improvement on the running time without having an impact on the results.

Big noise

Big step size (s=8) PSNR = 23.53 Small step size (s=3) PSNR = 24.79

Figure 4: Single scale denoised image varying step size (big noise)

With bigger noise the situation is completely different. When using the bigger step size, the algorithm is not able to denoise the image. The use of a smaller step size indeed improves the performance, both perceptually and in terms of PSNR. In this case we cannot choose to speed up the algorithm by increasing the step size because the decrease in the performance is significant.

Number of scales

Single scale PSNR = 29.81 3-Scales PSNR = 26.26

Figure 5: Multi-scale denoised image (small noise)

Surprisingly, the usage of the multi-scale approach doesn't really work. The single scale denoised image is good but the 3-scale denoised one presents some heavy blur. Both perceptually and in terms of PSNR, the 3-scale denoising is worse than the single scale one.

Single scale PSNR = 24.75 3-Scales PSNR = 23.69

Figure 6: Multi-scale denoised image (big noise)

The same previous behavior is observed, although the difference between both images is smaller in this case. It can be argued that with higher noise, the multi-scale approach does better than single-scale on the regular parts of the image (such as the sky), because it's able to remove more of the low frequency noise. However, on the rest of the image some artifacts show up, degrading the results. Both visually and in PSNR, the overall results are clearly better for the single-scale algorithm.

Comparison against BM3D

Based on the previous results, we'll now compare the performance of EPLL against BM3D, using the parameters which provided better denoising capabilities:

- Single scale
- Smallest step size for big noise, and highest step size for small noise

Small noise

EPLL PSNR = 29.79 BM3D PSNR = 34.526

Figure 7: EPLL - BM3D comparison (small noise)

Although both algorithms returned acceptable results. BM3D is vastly better at preserving details: the leaves, the bus' folding part, and the arc on the top right of the image. The PSNR is conclusive as well.

Big noise

EPLL PSNR = 24.82 BM3D PSNR = 26.4448

Figure 8: EPLL - BM3D comparison (big noise)

With bigger noise, BM3D results are, again, much better. EPLL manages to denoise the image but its reconstruction is too smooth. All the details are almost lost, while BM3D manages to preserve most of them.
In conclusion, BM3D is consistently better than EPLL across these experiments.

Zoran-Weiss Gaussians Mixture Model

For this paper, the authors proposed and adjusted a GMM to a massive set of patches extracted from noiseless images. This model allows us to represent any noiseless patch (and any image) as the result of a mixture of Gaussians with known parameters.
Let's make a brief interpretation of some of the Gaussians and the expressiveness of this model.
From the results we can see that there's a dominant Gaussian, in the sense that its mixing coefficient is far higher than the rest.

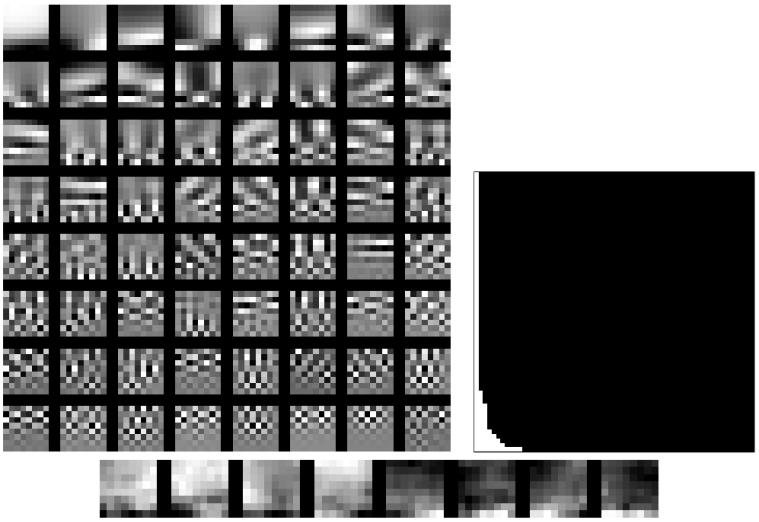

Figure 9: Zoran-Weiss Gaussian with highest probability (0.00363). Top, Eigenvectors (left) and eigenvalues. Bottom, simulated patches.

Seems reasonable that the most likely Gaussian, corresponds to one which models mostly flat patches. This can be interpreted as that in natural images, the most likely patches (the most common) are flat or regular areas. This vector has a sparse representation, concentrating most of its variance around a few eigenvectors.

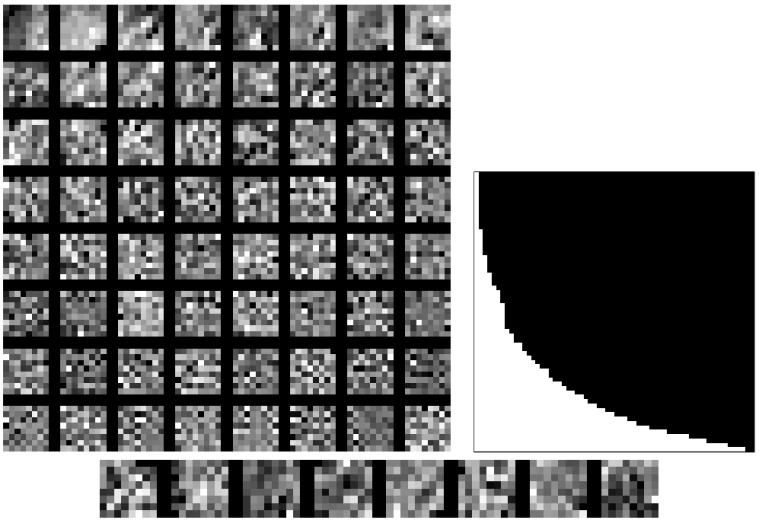

Figure 10: Zoran-Weiss Gaussian (probability 0.00019) Top, Eigenvectors (left) and eigenvalues of Gaussian 71 . Bottom, simulated patches.

On the other hand, this vector is not sparse. This Gaussian allows the model to handle more complex textures, such as the ones represented by the simulated patches. Again, it seems reasonable that this specific kind of texture has a much lower probability than a flat patch.

By inspecting more Gaussians from the model, it is curious to notice that there are many patterns that look really similar to some vectors from the DCT basis. This wide range of

different Gaussians show that the model seems to be general enough to represent all possible natural patches from an image.