Image Deblurring with Blurred/Noisy Image Pairs

Yohann Salaun

January 24, 2013

Introduction

Taking a picture under low light condition often results in bad quality images. With SLR cameras, options as ISO, exposure time and aperture can be better chosen to fit the bad light conditions. However, even with such gears, the picture is not perfect when taken with hands only. A noise/blur dilemma appears and it is often impossible to avoid at least one of these artifacts. The aim of [3] is to mix up two pictures facing a strong noise for one and a strong blur for the other in order to create a picture that is nor noisy nor blurry.

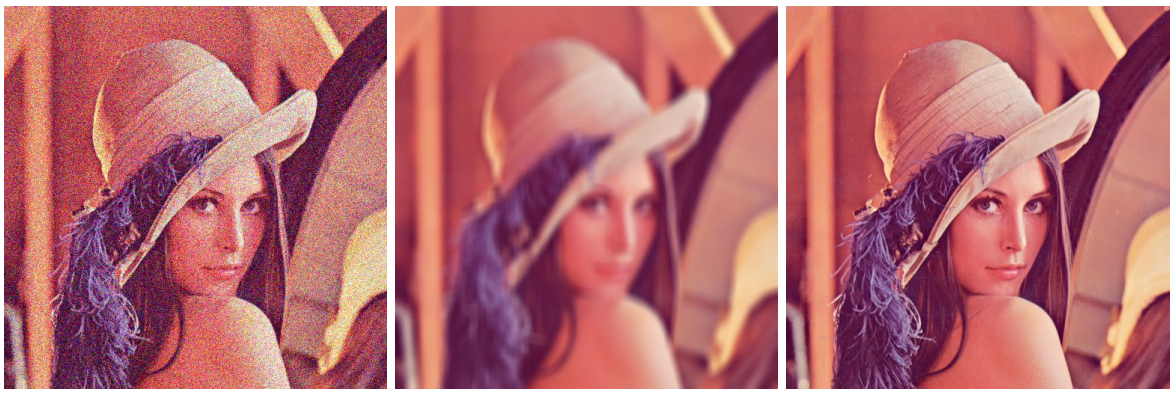

Figure 1: On the left, the noisy picture and on the center the blurry one. Both of these pictures have been virtually altered from the left one.

The original article [3] proposes a complex method that uses deblurring and denoising to improve image quality. In this report only a part of their method will be exposed and implemented.
In order to have a comparison for the denoising, the deblurring and the estimated kernel, the images used as input will be virtually noised/blurred versions of a picture.

1 Algorithm Overview

The general idea of this algorithm is to use the noisy picture in order to find a good estimate of the blur kernel and then use it to deconvolve the blurry picture.

Input : Noisy picture $\mathbf{N}$, Blurry picture $\mathbf{B}$, estimated kernel size $\mathbf{k}_{size}$
Output : Estimated picture $\mathbf{I}$, estimated kernel $\mathbf{k}$
$\mathbf{N}_d$= denoise($\mathbf{N}$)
$\mathbf{I}$= $\mathbf{N}_d$
while *change* $> \epsilon$ **do**
  Estimate kernel $\mathbf{k}$ with $\mathbf{I}$ and $\mathbf{B}$ s.t. $\mathbf{B} = \mathbf{I} \otimes \mathbf{k}$.
  Deconvolute blurred picture $\mathbf{B}$.
  Mix informations to improve estimation $\mathbf{I}$.
  Compute *change* between 2 iterations.
end

Algorithm 1: Main algorithm

2 Theoretical Description

In this section, $\mathbf{I}$ will refer to the current estimate of the good quality picture, $\mathbf{B}$ will be the blurry input picture and $\mathbf{N}$ the roughly denoised version of the noisy input picture.

2.1 Initialization

When taken with cameras, $\mathbf{N}$ and $\mathbf{B}$ have different color intensities for their pixels even when the noise and the blur does not affect them. This is due to the difference in the camera settings when the pictures are taken. In order to have the same intensity scale, $\mathbf{N}$ is pre-multiplied by a ratio $\frac{ISO_{\mathbf{B}}\Delta t_{\mathbf{B}}}{ISO_{\mathbf{N}}\Delta t_{\mathbf{N}}}$ where Δt is the exposure time.
As I used images with virtual blur/noise, I skipped this step of intensity scaling.

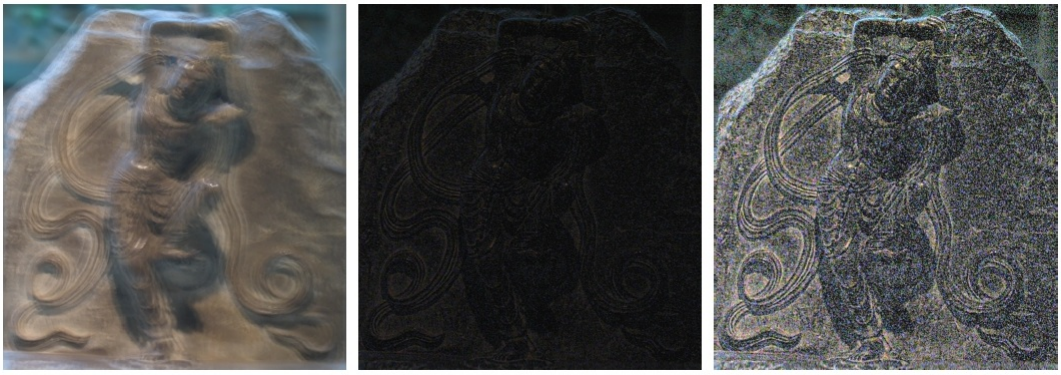

Figure 2: On the left, the blurry picture and on the center the noisy one. The left one is the noisy picture multiplied by the exposure ratio in order to have comparable intensity measures between the blurry and noisy pictures.

Then the noisy picture $\mathbf{N}$ is denoised by an extern process. [3] proposes to use the wavelet based algorithm developed by [1].
However, as this code is no more available, I used a simple bilateral filter to roughly denoise $\mathbf{N}$ into $\mathbf{N}_d$. The denoised picture is used to initialize the estimated picture $\mathbf{I}$.

Figure 3: On the left, the noisy picture and on the right the denoised picture obtained with a simple bilateral filter.

2.2 Kernel Estimation

The aim of this part is to estimate the blur kernel that transformed $\mathbf{I}$ into $\mathbf{B}$. The method exposed in [3] is based on a gradient descent.
The problem is thus formulated as the finding the solution of:

$$\mathbf{B} = \mathbf{I}\mathbf{k} \tag{1}$$

where $\mathbf{k} \in \mathbb{R}^{\mathbf{k}_{size}}$ and $\mathbf{B} \in \mathbb{R}^{\mathbf{n}}$ are written into column-vector form and $\mathbf{I} \in \mathbb{R}^{\mathbf{n} \times \mathbf{k}_{size}}$ is such that $\mathbf{B} = \mathbf{I} \otimes \mathbf{k}$. Solving (1) can be done in a least square sense and the Tikhonov regularization method has been adopted in [3]:

$$\min_{\mathbf{k} \in \mathbb{R}^{\mathbf{k}_{size}}} ||\mathbf{I}\mathbf{k} - \mathbf{B}||_2^2 + \lambda^2 ||\mathbf{k}||_2^2, \quad s.t. \;\; \mathbf{k} \in \mathbb{R}^{+\mathbf{k}_{size}} \;\; and \;\; ||\mathbf{k}||_1 = 1 \tag{2}$$

To find the minimum of (1), a method similar to projected gradient descent is applied:

- $\mathbf{k}^0$ is initialized as the delta function.
- $\mathbf{k}^{n+1} = \mathbf{k}^n + \tau \nabla F(\mathbf{k}^n) = \mathbf{k}^n + 2\tau(\mathbf{I}^T\mathbf{B} - (\mathbf{I}^T\mathbf{I} + \lambda^2 Id)\mathbf{k}^n)$
- $\mathbf{k}^{n+1} = Proj_K(\mathbf{k}^{n+1})$

where F is the functional minimized in (1), τ the step and $K = \{\mathbf{k} \in \mathbb{R}^{+\mathbf{k}_{size}} s.t. ||\mathbf{k}||_1 = 1\}$.
In [3], the projection on K is build this way:

- positive the kernel coefficients $\mathbf{k}_i = max(\mathbf{k}_i, 0)$
- normalize the kernel $\mathbf{k} = \frac{\mathbf{k}}{||\mathbf{k}||}$

This method is a bit questionable since it is not a real projection and some points are projected on the null kernel.

After some unsuccessful try with this algorithm, I finally used the gradient descent algorithm without projection and used a projection at the end only in order to have a correct blur kernel.

2.3 Deconvolution

Once the blur kernel has been estimated, it can be used in order to find a better estimate of the final picture. The deconvolution is then performed with the standard Richardson-Lucy algorithm.
This method is an iterative one that needs the blurred picture $\mathbf{B}$, the estimated kernel $\mathbf{k}$ and an initialization of the picture which here will be $\mathbf{I}$.
The standard form of the algorithm uses the iterative formula:

$$\mathbf{I}_i^{k+1} = \mathbf{I}_i^k \sum_j \frac{\mathbf{B}_j}{(\mathbf{k} \otimes \mathbf{I}^k)_j} \mathbf{k}_j[i]$$

where $\mathbf{k}_j[i]$ is the j^{th} coefficients of the kernel $\mathbf{k}$ centered in pixel i.

In order to have better results for the deconvolution, [3] propose to use the residual pictures :

- $\Delta\mathbf{I} = \mathbf{I} - \mathbf{N}_d$
- $\Delta\mathbf{B} = \Delta\mathbf{I} \otimes \mathbf{k} = \mathbf{B} - \mathbf{N}_d \otimes \mathbf{k}$

The idea that motivates this change is that the RL-deconvolution often creates a ringing effect around the picture and the ringing effect of residual pictures will be at their scale, which is small compared to the whole pictures.
An offset is also added to each residual in order to avoid zero value for color intensities. The offset value is 1 when the color intensities are normalized to range [0, 1].

When I tested the algorithm on the pictures in FIG.1, the deconvolution worked well for the whole pictures but the residual one had really bad results. Thus I could not really apply this part with my own pictures.

2.4 De-ringing

The aim of this part is to avoid the ringing effect and mix the pictures in order to have sharp details in the final picture.
The ringing effect is a consequence of the Gibbs phenomena in Fourier analysis and appear near border. Since, its effect are stronger when the kernel size is larger, it can be an important artifact in the output of such a method. Moreover, as the blur kernel is an unknown parameter, the larger its estimated size is, the better the kernel estimation can be done. Thus, decreasing the ringing effect will allow better estimations of the optimal picture.
In order to avoid ringing effects, the algorithm used is a slight variation of the RL-deconvolution :

$$\mathbf{I}_i^{k+1} = I_{GAIN}[i](\mathbf{I}_i^k \sum_j \frac{\mathbf{B}_j}{(\mathbf{k} \otimes \mathbf{I}^k)_j} \mathbf{k}_j[i])$$

$I_{GAIN}[i]$ is a multiplier factor that is here to decrease the contrast of the residual pictures since the ratio $\sum_j \frac{\mathbf{B}_j}{(\mathbf{k} \otimes \mathbf{I}^k)_j} \mathbf{k}_j[i]$ increase its magnitude at each iteration.

The gain map I_{GAIN} is computed in order to be lower in flat region (i.e. where the ringing effect appears) and higher in edges regions :

$$I_{GAIN} = (1 - \alpha) + \alpha \sum_{l} \|\nabla \mathbf{N}_d{}^l\|$$

The authors of [3] are not very explicit about the computation of this gain map. Moreover, as I only manipulated small pictures and small kernels (due to computation time) I did not really observed the ringing effect. Thus I did not implemented this part of the algorithm.

The algorithm results are now a picture $\mathbf{I}$ with details and ringing effects obtained from residual RL and the "opposite" picture $\mathbf{I}_g$ with bad details but no ringing effect. The idea is then to mix these two pictures in order to have an estimate $\mathbf{I}$ that has nor ringing effects, nor bad details.
The method used is highly inspired from HDR algorithm :

- Compute a detail layer $\mathbf{I}_d = I - F(I)$ where F is a low-pass filter such as the bilateral filter
- Compose the detail layer $\mathbf{I}_d$ and the base layer $\mathbf{I}_g$

Once again, as I did not observed the ringing effects, I could not implemented this part of the algorithm.

3 Results

Since the computation has been very slow, I used only small resolution pictures (200 x 200 pixels) and small kernels (5×5 pixels or 9×9 pixels).
The original pictures, the estimations and the kernel estimations can be seen in FIG. 4 and FIG. 5.

As we can see, the estimated kernel is very influenced by its Dirac initialization. However, lowering the λ coefficient in (2) does not change a lot the result.

FIG. 6 illustrates the results difference obtained with different initialization :

- $\mathbf{I}^0 = 0$ obtains similar result to the classical method for the kernel estimation. However, the ringing artifact supposed to appear with classical method can be seen.
- $\mathbf{I}^0 = \mathbf{N}$ obtains better estimation for the blur kernel but keeps a lot of noise in the final estimation.

This observation seems to confirm that using the noisy picture to estimate the blur kernel is a good idea but maybe denoising it at the early beginning is too soon for a good information on the blur kernel.

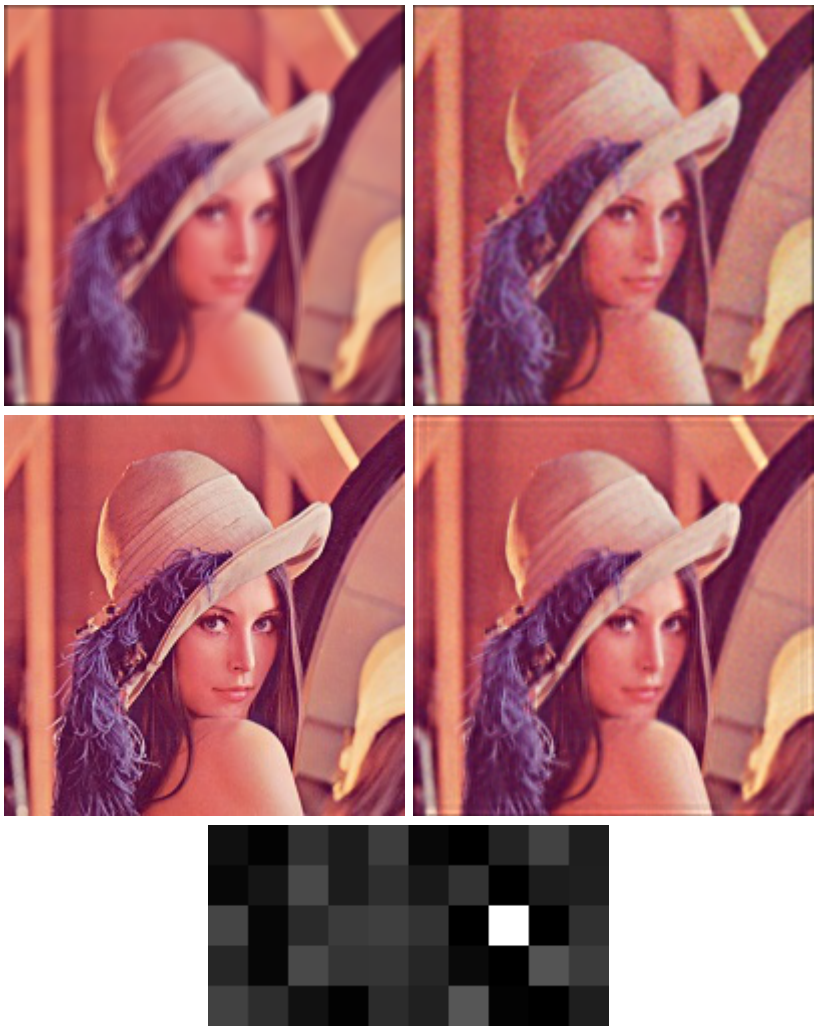

Figure 4: On the top left, the blurred picture obtained with a 5×5 kernel and on top the right the denoised picture obtained with a simple bilateral filter. The bottom left is the original picture and the right one is the estimation obtained with the algorithm. The last picture is a comparison between the original blur kernel on the left and its estimation on the right.

Figure 5: On the top left, the blurred picture obtained with a 9×9 kernel and on top the right the denoised picture obtained with a simple bilateral filter. The bottom left is the original picture and the right one is the estimation obtained with the algorithm.The last picture is a comparison between the original blur kernel on the left and its estimation on the right.

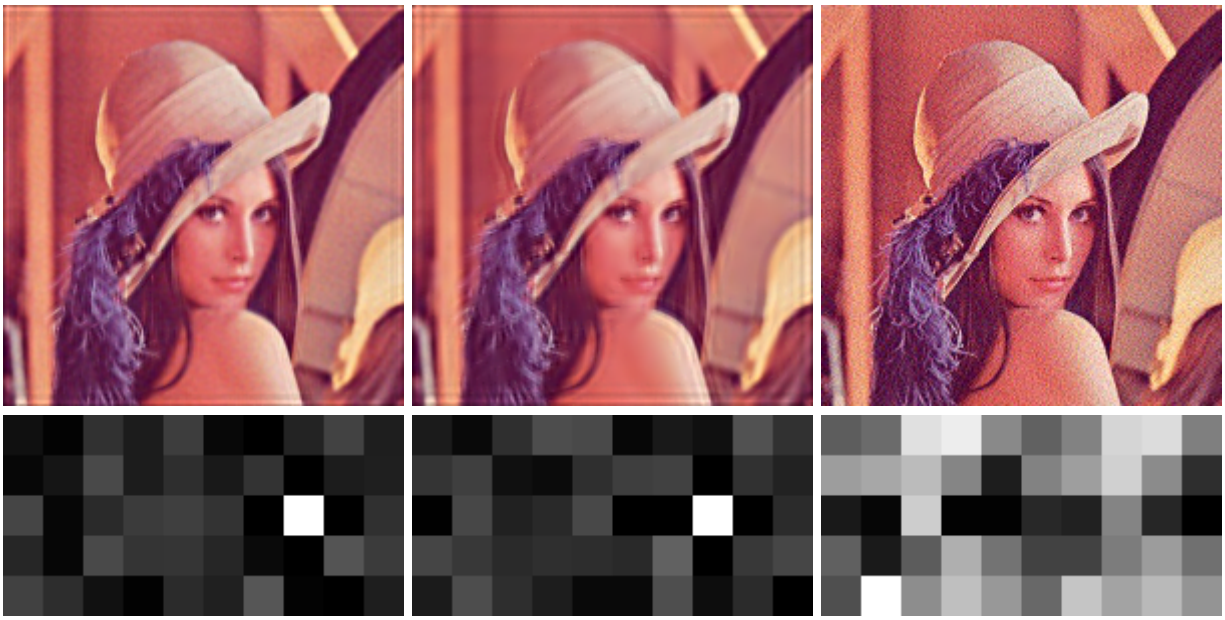

Figure 6: Comparison of results obtained with different initialization for $\mathbf{I}$. From left to right, $\mathbf{I}^0 = \mathbf{N}_d,\ \mathbf{I}^0 = 0,\ \mathbf{I}^0 = \mathbf{N}$

4 Conclusion

The results I obtained with my algorithm cannot really be compared with the results obtained in [3] because only a part of it was really implemented and due to computation time, I used only small scales instead of more realistic one (8 Mpixels pictures and 64×64 kernels).
In fact, the kernel size is a very important parameter because, as the real blur kernel is unknown, the larger the estimation is, the better the results should be. If the estimated blur kernel is bigger than the real one, the estimation is far better than in the opposite case (see FIG.7).
Moreover, the cases I used were virtual pictures while for real pictures, the blur in $\mathbf{B}$ is caused by the long exposure time but also by the camera movement between the first shoot $\mathbf{N}$ and the second one $\mathbf{B}$.
Finally, I think that mixing informations coming from a blurry picture and a noisy picture if the same scene is a great idea and [3] seems to show that it works greatly. However, this article is always very explicit and I could not implement and observe all of what they did.

References

[1] J. Portilla, V. Strela, M. Wainwright, and E.P. Simoncelli *Image denoising using scale mixtures of gaussians in the wavelet domain.* IEEE Trans. on Image Processing 12, 11, 1338–1351, 2003.

[2] H. Richardson *Bayesian-based iterative method of image restoration.* JOSA, A 62, 1, 55–59. 1972.

[3] L. Yuan, J. Sun, L. Quan, and H-Y. Shum *Image Deblurring with Blurred/Noisy Image Pairs.* ACM SIGGRAPH, 2007.

Figure 7: Leftward pictures are blurry inputs and rightward ones are estimations. Comparison of results obtained with differences between real kernel size and estimated kernel size. Up pictures are obtained with a real kernel size of 15 pixels and an estimated one of 5 pixels. Bottom pictures are obtained with a real kernel size of 3 pixels and an estimated one of 9 pixels.