Fast Deconvolution using Hyper-Laplacian Priors

Placido Mora

2024 (update)

1 Introduction

This report briefly documents my Python implementation to the algorithm presented in the paper

> D. Krishnan, R. Fergus: "Fast Image Deconvolution using Hyper-Laplacian Priors". Proceedings of NIPS 2009.

The algorithm presented in the paper gives a solution to the image deconvolution problem

$$(\mathbf{x} \circledast \mathbf{k}) + \epsilon = \mathbf{y} \tag{1}$$

where $\mathbf{x}$ =clean image, $\circledast$ is 2-dimensional convolution, $\mathbf{k}$ =kernel, ϵ =random noise, and $\mathbf{y}$ =blurred noisy image.

In the paper, the solution to this problem is described as resulting from the minimization shown on Equation 2 (not shown here, refer to the paper), which is split in two parts:

- **x sub-problem**, which introduces parameters β and λ. See description below.
- **w sub-problem**, which introduces parameter α. This implementation is specifically for $\alpha = 1/2$ as instructed. See description below.

x sub-problem

The solution for $\mathbf{x}$ is shown to be

$$\mathbf{x} = \mathscr{F}^{-1}\left(\frac{\mathscr{F}(F^1)^* \circ \mathscr{F}(\mathbf{w}^1) + \mathscr{F}(F^2)^* \circ \mathscr{F}(\mathbf{w}^2) + (\lambda/\beta)\mathscr{F}(K)^* \circ \mathscr{F}(\mathbf{y})}{\mathscr{F}(F^1)^* \circ \mathscr{F}(F^1) + \mathscr{F}(F^2)^* \circ \mathscr{F}(F^2) + (\lambda/\beta)\mathscr{F}(K)^* \circ \mathscr{F}(K)}\right) \tag{2}$$

where $*$ is the complex conjugate, $\circ$ denotes component-wise multiplication, and

- $F_i^1\mathbf{x} = (\mathbf{x} \circledast f_1)_i, F_i^2\mathbf{x} = (\mathbf{x} \circledast f_2)_i \;\forall\, pixel\, i$. Filters: $f_1 = [1, -1], f_2 = [1, -1]^T$
- K: optical transfer function for kernel k.

- λ: weighting term that controls the strength of the regularization.
- β: weight that varies during optimization. As $\beta \to \infty$ the solution converges.
- $\mathbf{w}^1, \mathbf{w}^2$ are auxiliary variables

In order to simplify the notation in the Python code, the solution is rewritten as

$$\mathbf{x} = \mathscr{F}^{-1}\left(\frac{N_1 + \xi N_2}{D_1 + \xi D_2}\right) \tag{3}$$

where: $N_1 = \mathscr{F}(K)^* \circ \mathscr{F}(\mathbf{y})$, $N_2 = \mathscr{F}(F^1)^* \circ \mathscr{F}(\mathbf{w}^1) + \mathscr{F}(F^2)^* \circ \mathscr{F}(\mathbf{w}^2)$, $D_1 = \mathscr{F}(K)^* \circ \mathscr{F}(K)$, $D_2 = (\beta/\lambda)[\mathscr{F}(F^1)^* \circ \mathscr{F}(F^1) + \mathscr{F}(F^2)^* \circ \mathscr{F}(F^2)]$, and $\xi = \beta/\lambda$. Note that $N2$ is the only term depending on $\mathbf{w}^1$ and $\mathbf{w}^2$.

w sub-problem

This sub-problem starts in the paper with the equation

$$w^* = arg\, min_w\, |w|^\alpha + \frac{\beta}{2}(w - v)^2 \tag{4}$$

Solving for **w** can be done via look-up table (LUT), or via analytical solutions, for the cases $\alpha = 1/2$ or $\alpha = 2/3$ only. As explained, this implementation is for $\alpha = 1/2$, and in such case we find the roots for the cubic polynomial

$$w^3 - 2vw^2 + v^2 w - sign(v)/4\beta^2 = 0 \tag{5}$$

2 Implementation

Based on the description above, the user will find the following in the Python file:

- `solve eq3`: function for the *x sub-problem*. This is basically used to solve equation 3 of this report.
- `solve eq5 alpha12`: function for the *w sub-problem*. This is basically used to solve equation 5 of this report.
- `fastd`: main function for the deconvolution algorithm. It calls other functions (see below).
- `main`: featuring 5 steps
  - specify test kernel
  - input test images (directly from the web at runtime)
  - blur test image
  - process blurred test image (run `fastd`)
  - display output

In the following 2 pages some test samples are shown.

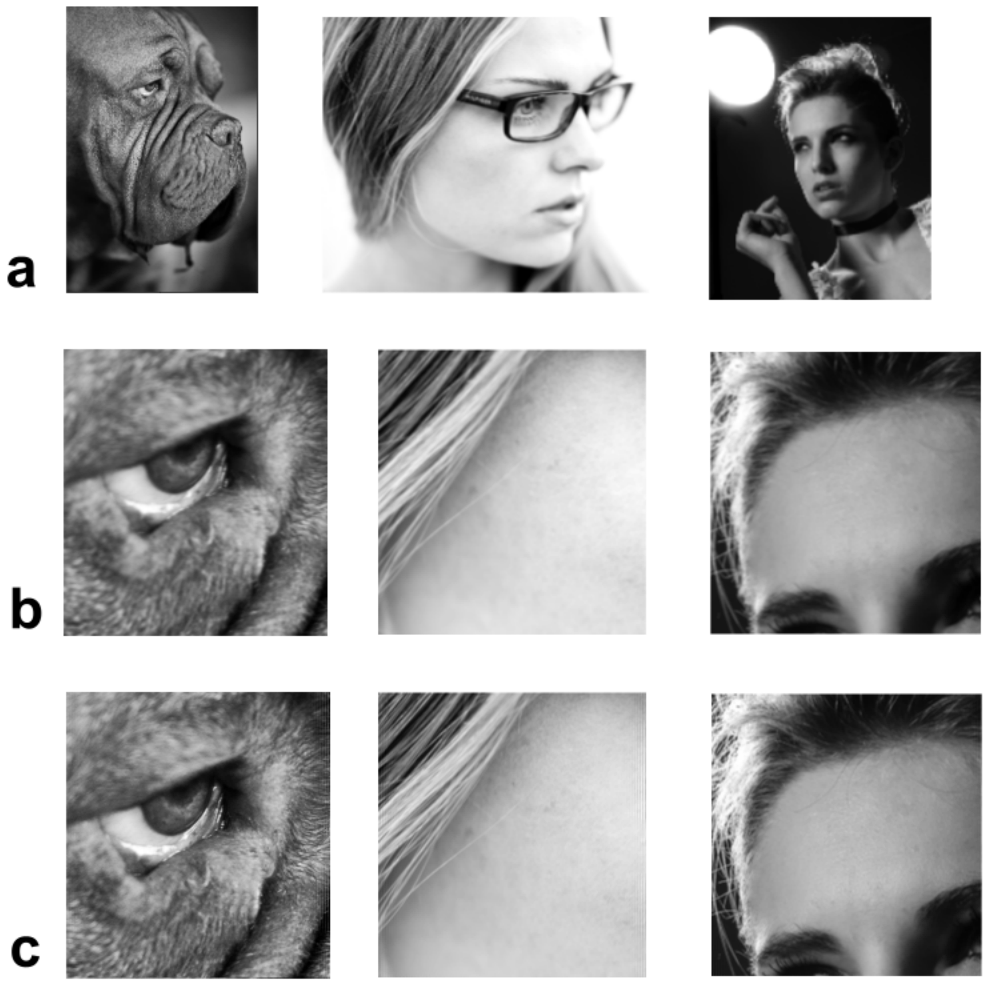

Figure 1: Test pictures. (a) original, (b) blurred, (c) deblurred.

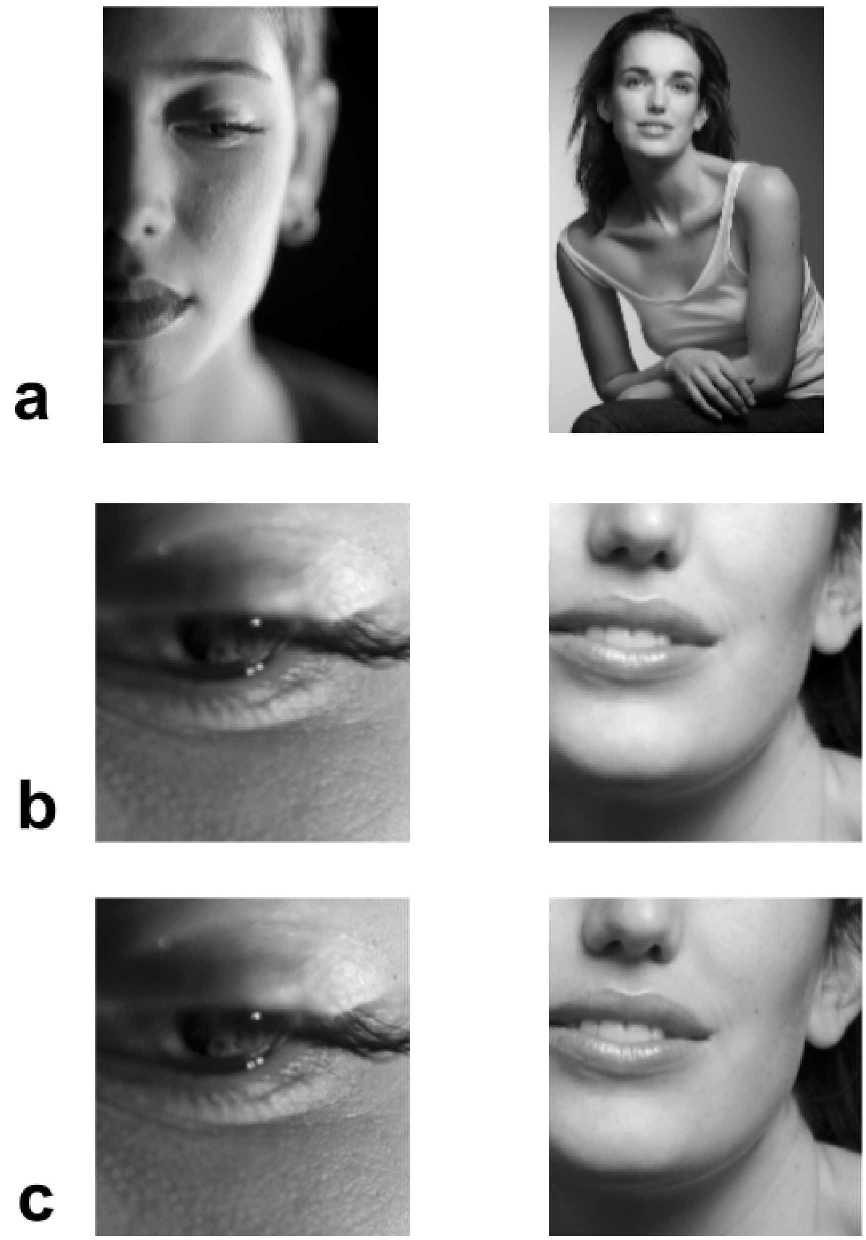

Figure 2: Test pictures. (a) original, (b) blurred, (c) deblurred.