Image deblurring as an inverse problem

Oleg Makhnin

February 12, 2010

Can you read this?

- **Image:** F = a matrix of grayscale intensities (real numbers in a bounded interval), generally assumed unknown
- **Image blurring:** each pixel is spread out according to some matrix B, at the maximum of m pixels across and m pixels up/down.
- **Data:** a blurred image $G = \mathcal{K}_B F$ with the operator $\mathcal{K}_B$ based on the pixel blurring operation B applied to each pixel, and the results added up.
- **Deblurring problem:** Based on the given blurred image G (and, maybe, B), restore "as well as possible" the original image F

Matrix form: if F and G are turned into vectors f, g then $g = K_B f$ with some matrix $K = K_B$.

However, the matrix K is usually not invertible. Also, there's often noise involved:

$$g = Kf + e$$

Loss of information from blurring; some recovery is possible because the image F is redundant

I. Bayesian approach

Bayesian regularization

The image is treated as *Gauss-Markov Random Field* with the values at any site/pixel $s = (i, j)$ dependent on its neighbors.

Thus, the *prior* distribution for the image is

$$p(F) \;\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{\texttt{sites}\ s\sim t}(F(t)-F(s))^2\right\}$$

The *likelihood* of the data G given the image F

$$p(G|F) \;\propto \exp\left\{-\frac{1}{2\sigma_e^2}\sum_{\texttt{sites}\ s}\left[G(s)-\mathcal{K}F(\texttt{neighb. of}\ s)\right]^2\right\}$$

the parameters σ and σ_e control the degree of smoothness and overall sharpness of the posterior distribution

Posterior probability

Makes the Bayesian inversion, that is

$$\texttt{Posterior} \propto \texttt{Prior} \times \texttt{Likelihood}$$

$$\log p(F|G) = const - \frac{1}{2\sigma_e^2} \sum_{\texttt{sites } s \sim t} (F(t) - F(s))^2 -$$

$$-\frac{1}{2\sigma_e^2} \sum_{\texttt{sites } s} [G(s) - \mathcal{K}F(\texttt{neighb. of } s)]^2$$

Based on this (for example, finding its maximum, or maybe mean) we can produce an estimate of F

The posterior $p(F|G)$ is a multi-dimensional distribution that's hard to compute. Instead, we can *sample* its values (i.e. get the "typical" values of F).

Gibbs sampler:

To set up a Gibbs sampler, we start with random values for the parameters (in our case, values of F). Then, sample from the *full conditional posteriors*

$$P(\texttt{a parameter} | \texttt{ all other parameters, data } G, \sigma, \sigma_e)$$

In a simple single-site updating scheme, update F(s) based on the values of its neighbors.

Pressures to:

1) conform to its neighbors
2) conform to the data G

An evolution scheme which eventually migrates to "likely" F.

Metropolis algorithm

In order to draw from $P(F(s)|F(\texttt{neighbor sites}), G)$, we use **Metropolis algorithm**:

1. Draw a new candidate $f_{\texttt{new}}$ for $F(s)$
2. Compare the posteriors $post = p(F|G)$ with the old and the new F.
3. If the new value has higher posterior, then accept it.
   If lower, then accept it with probability $= post_{new}/post_{old}$

It's nice because the knowledge of proportionality constant in $p(F|G)$ is not required!

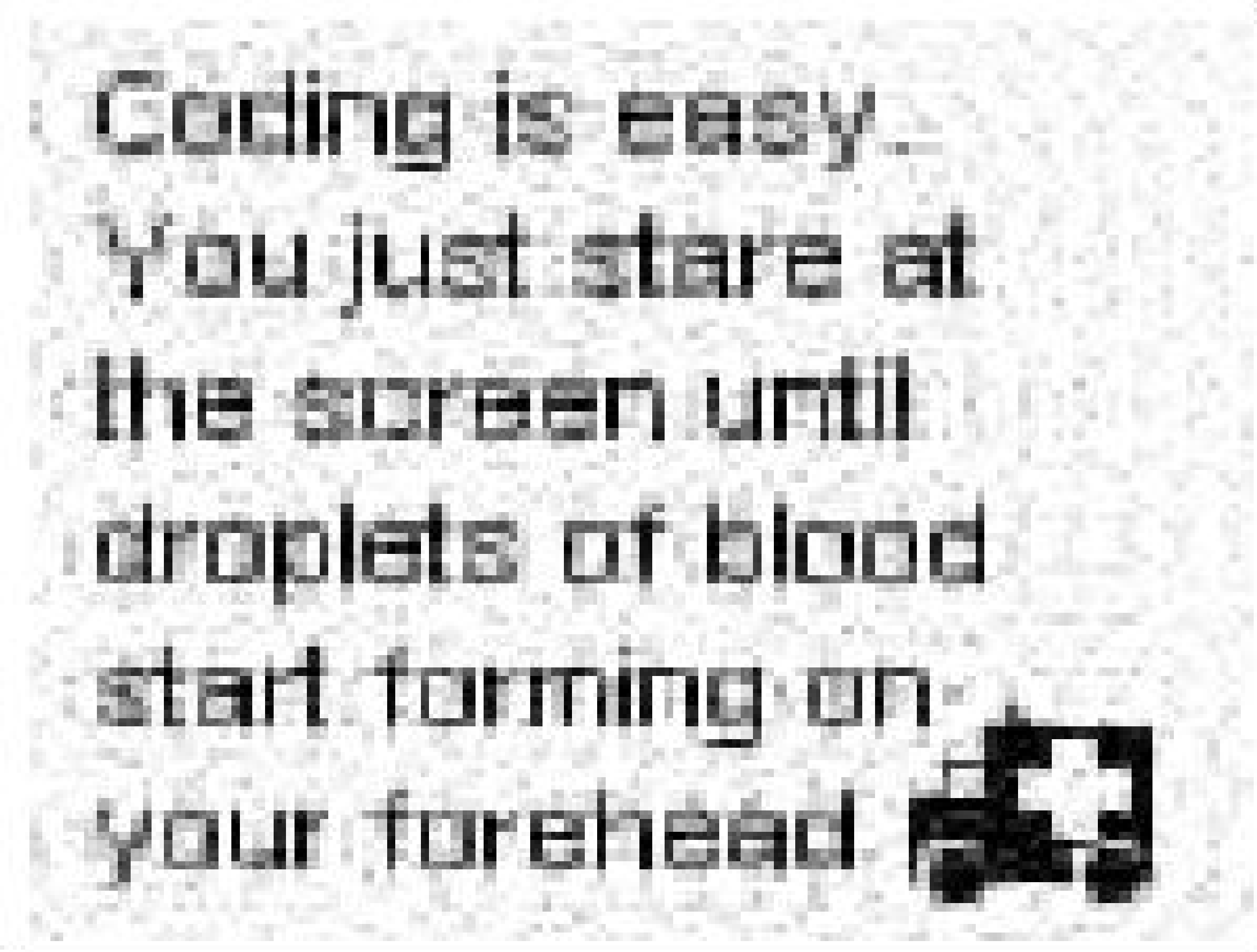
Furthermore, one can use *simulated annealing* based on Metropolis algorithm to find the "most likely" F.

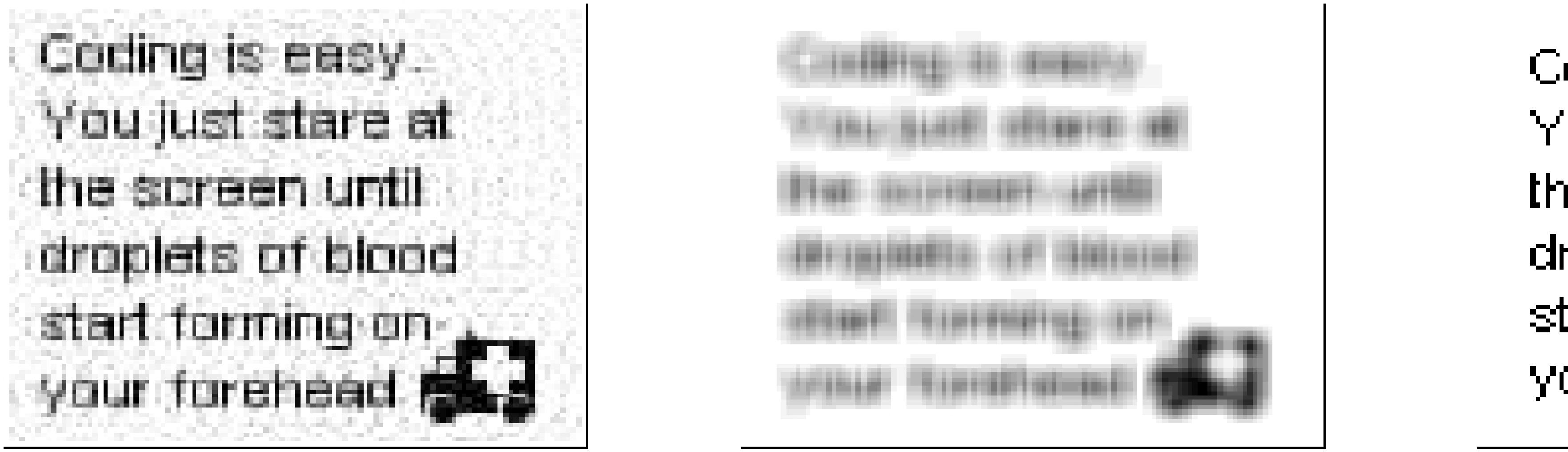
[Demo of Gibbs here]

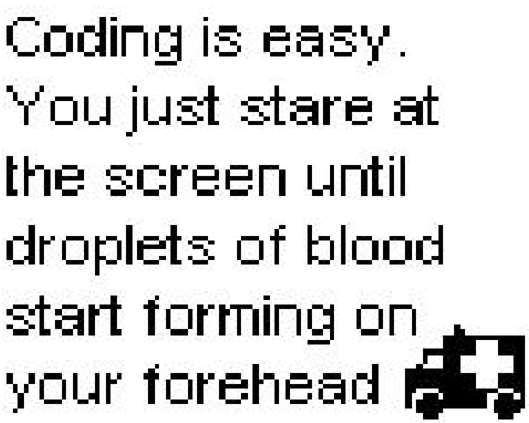
Some results: $\hat{F}, G, \mathcal{K}\hat{F}$



Some results:




Coding is easy.
You just stare at
the screen until
droplets of blood
start forming on
your forehead


Coding is easy.
You just stare at
the screen until
droplets of blood
start forming on
your forehead

II. Regularization approach

Regularization approach:

Instead of solving $\min_f ||Kf - g||$, let's solve

$$\min_f ||Kf - g||^2 + \alpha ||f||^2$$

i.e. penalize large values of f

Compute the *singular value decomposition (SVD)* of K:

$$K = UDV'$$

where U, V are orthogonal matrices, and $D = diag\{d_1, d_2, ..., d_N\}$ consists of *singular values* of K.

Let K^+ be a regularized inverse of K (see below) then, the deblurred image $\hat{F}$ is

$$\hat{F} = K^+ G = V D^+ U' G$$

where the inverse D^+ is taken using

$$d_i^+ = \frac{d_i}{\alpha^2 + d_i^2}$$

However, K is huge...

Note: if K is a *circulant* matrix,

$$K = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_3 & c_2 \\ & & \ddots & & & \vdots \\ & & & \ddots & & \vdots \\ \cdots & & & & \ddots & \vdots \\ c_{n-1} & c_{n-2} & \cdots & & c_1 & c_0 \end{bmatrix}$$

then

$$K = WDW^*$$

where D is diagonal, and W is unitary, but now they have complex values. (W contains basis for Discrete Fourier Transform.)

Such K assumes the toroidal structure of the image (first row linked to the last row etc.)
D contains the Fourier coefficients for the first column of K. It is real-valued if the blur B is symmetric.

No worries, we can still do the regularization D^+:

$$F = K^+ G = W^* D^+ W G$$

As an added bonus, we use FFT (Fast Fourier Transform), that is multiplying by W is equivalent to making a FFT, and multiplying by W^* is equivalent to a ...

[Live demo here]

GEMAN AND YANG: NONLINEAR IMAGE RECOVERY WITH HALF-QUADRATIC REGULARIZATION

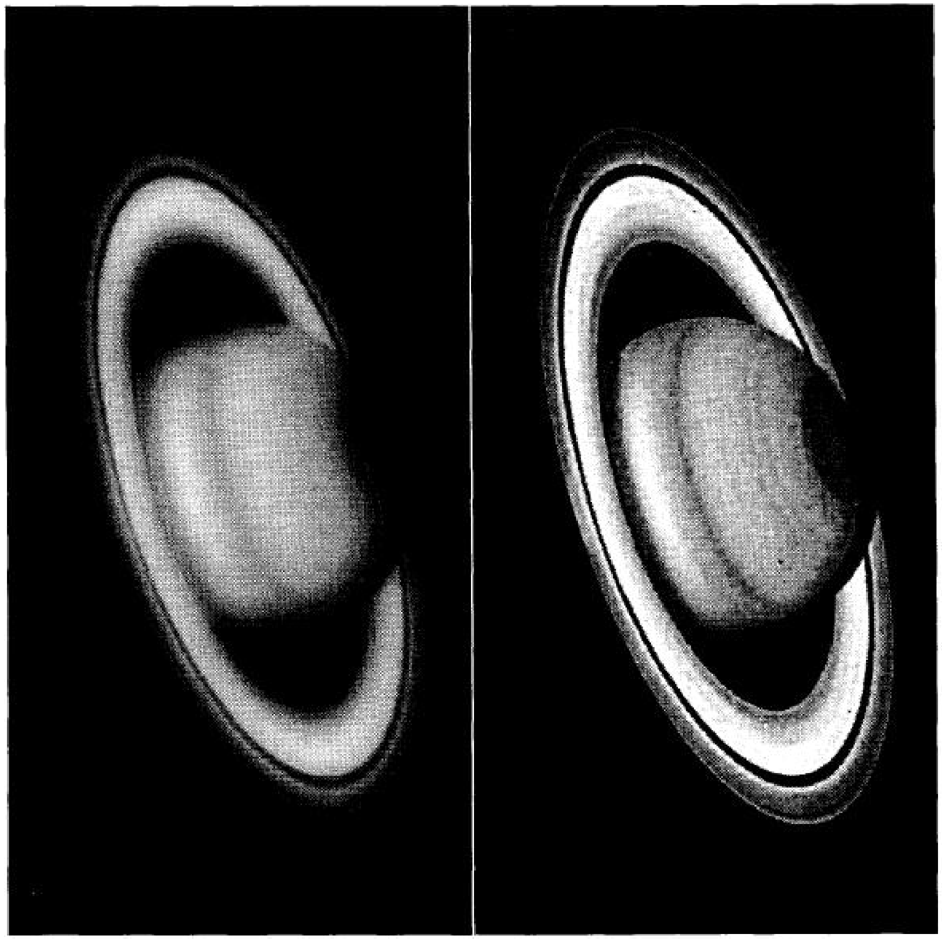
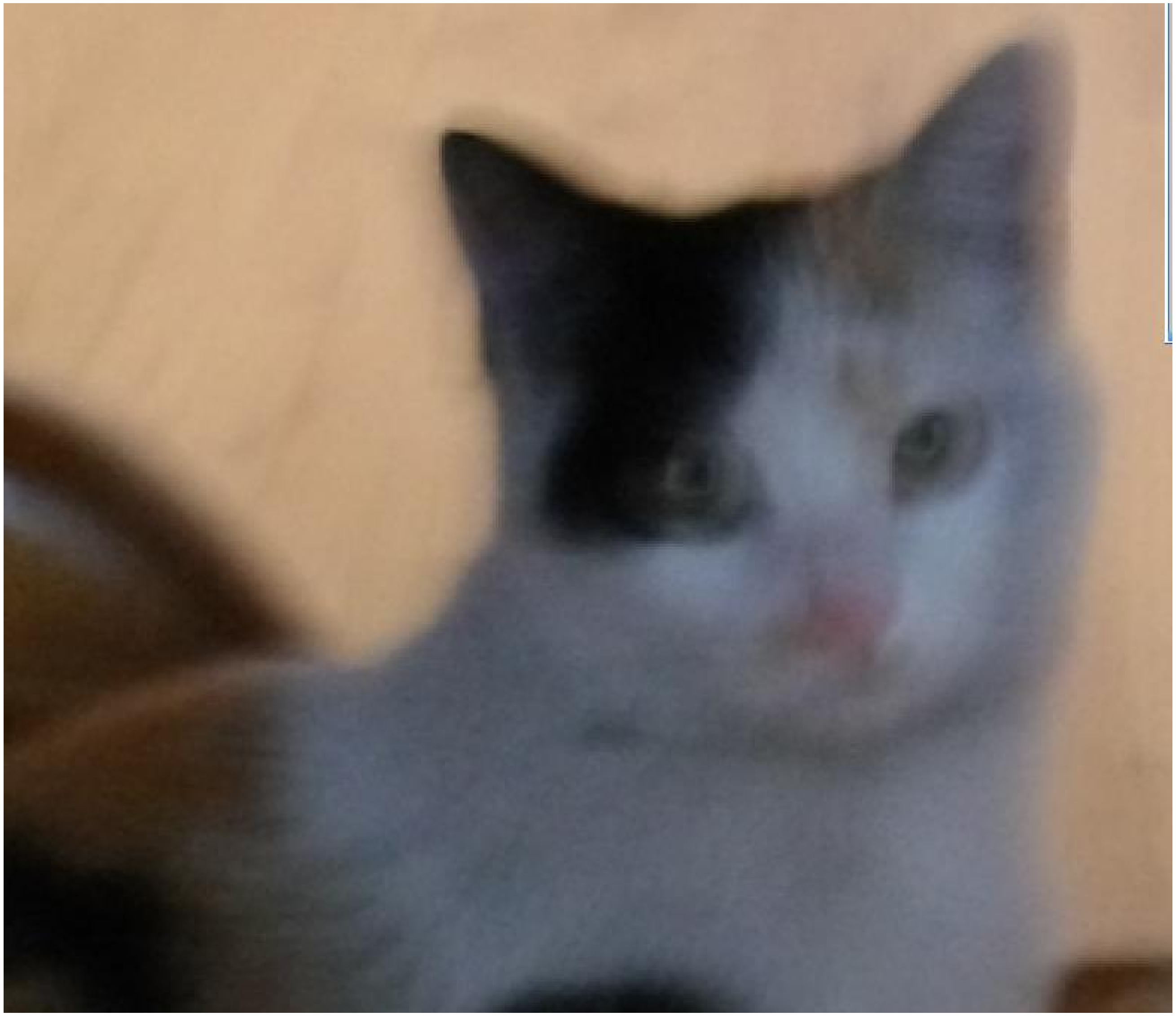


Fig. 1. Experiment 1. Left: Hubble data after despiking. Right: restored by our algorithm.

Geman & Yang

Would you please deblur me?



Challenge: unknown B

Now, in many cases B is not given. How do we find it?

One idea: we can set up a Gibbs sampler to repeatedly draw from

$$p(B|F,G), \quad p(F|B,G)$$

(the latter maybe by FFT approach)

$p(B|F,G)$ can be obtained using multivariate Gaussian, however a lot of computation when B is large

Will this work?

Can we use FFT for $p(B|F,G)$ somehow?

References

S. Geman and D. Geman (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, **6**, 721-741.

see

`http://cis.jhu.edu/people/faculty/geman/publications.html`

D. Geman and C. Yang (1995). Nonlinear image recovery with half-quadratic regularization *IEEE Transactions on Image Processing*, **4**, no. 7, 932- 945

J.G. Nagy and D.P.O'Leary (2003). Image Deblurring: I can see clearly now. *Computing in Science and Engineering*, **5**, no. 3, 82-84

Thanks!