Figure 1: A blurred mystery photo taken at the Ithaca SPCA.

Proj 2: Where Are My Glasses?

1 Introduction

The image in Figure 1 is a blurred version of a picture that I took at the local SPCA. Your mission is to try three approaches to de-blurring this picture, and to analyze why these approaches behave as they do.

Images in MATLAB are represented as three-dimensional arrays of size height $\times$ width $\times$ 3, where the three layers represent the red, green, and blue color intensities at each pixel. The blurred image file in Figure 1 has eight-bit color depth, which means that each of the RGB values is represented as an integer between 0 and 255 (inclusive). For convenience, we will reshape these three-dimensional arrays into matrices with height $\times$ width rows and three columns, with the columns representing red, green, and blue color intensities as floating point numbers (still between 0 and 255).

The image was constructed from the original picture by the MATLAB commands

```
h = fspecial('motion', 30, 25);
im1 = imread('origimg.jpg');
im2 = imfilter(im1, h);
imwrite(im2, 'blurry.png');
```

These commands simulate motion blur at an angle of 25 degrees over a line about 30 pixels long. This is a *linear* transformation; that is, we can write

$$V^{\text{blur}} = \text{round}(HV^{\text{orig}})$$

where

$$V^{\text{orig}} = \begin{bmatrix} v_{\text{red}}^{\text{orig}} & v_{\text{green}}^{\text{orig}} & v_{\text{blue}}^{\text{orig}} \end{bmatrix}$$

and V^{blur} has a similar layout. Here the function round$(\cdot)$ means rounding each color intensity to eight bits (i.e. to an integer between 0 and 255). If everything were done exactly (no noise and no rounding to eight bits), we could recover the original image by computing $H^{-1}V^{\text{blur}}$. Of course, then this would not be much of a project!

You are given the following files:

`blurry.png` : Blurred image file.

`p2setup.m` : Loads the image and forms the blurring matrix H.

`p2image.m` : Displays an image (loaded or reconstructed).

`p2runner.m` : Runs the three methods, saves images, and reports run times.

You are responsible for the following deliverables:

- Four MATLAB functions to implement the deblurring methods.
- A report that addresses the questions posed in the rest of this prompt.

This project is inspired by the project on image deblurring by James G. Nagy and Dianne P. O'Leary ("Image Deblurring: I Can See Clearly Now" in *Computing in Science and Engineering*; Project: Vol. 5, No. 3, May/June 2003, pp. 82-85; Solution: Vol. 5, No. 4, July/August 2003, pp. 72-74). As usual, you are welcome to use any ideas or code you find in the literature, including in the article by Nagy and O'Leary, provided you give an appropriate citation.

2 The approaches

2.1 The naive approach

The simplest reconstruction approach is to simply solve

$$V^{\text{naive}} \approx H^{-1} V^{\text{blur}}.$$

You should complete the following function to implement this approach:

function imresult = p2naive(imblurd, H)

Questions

1. What does the reconstructed image look like? You can use the function `p2image` to see a picture.
2. For this image, give an estimated bound on $\|HV^{\text{orig}} - V^{\text{blur}}\|_F / \|V^{\text{blur}}\|_F$, and use the MATLAB function `condest` to estimate the condition number of H. Use these estimates to explain the picture.

2.2 Tikhonov regularization

A better approach to deblurring the image is *Tikhonov regularization*:

$$V^{\text{tik}} = \text{argmin}_V \, \|HV - V^{\text{blur}}\|_F^2 + \beta^2 \|V\|_F^2.$$

You should complete the following function to implement this approach:

function imresult = p2tikhonov(imblurd, H, **beta**)

The regularization parameter β should be small (in order that the data mismatch term $\|HV - V^{\text{blur}}\|_F^2$ should be small); but it should not be so small that the regularized problem still has the conditioning problems of the naive approach. How to choose the regularization parameter is in general an interesting problem, but one we will leave for another day. For this particular example, $\beta = 10^{-1}$ does a pretty good job.

Questions

1. Show that the Tikhonov regularized problem is equivalent to
$$V^{\text{tik}} = \text{argmin}_V \left\| \begin{bmatrix} H \\ \beta I \end{bmatrix} V - \begin{bmatrix} V^{\text{blur}} \\ 0 \end{bmatrix} \right\|_F^2$$
If you were in class on Feb 27, you may use your class notes!

2. Write an analytic formula for the singular values of the regularized matrix
$$\begin{bmatrix} H \\ \beta I \end{bmatrix}$$
in terms of β and of the singular values $\sigma_1, \ldots, \sigma_N$ for H. The largest singular value of H is just below one; using this fact, together with the condition number estimated earlier, give an estimate of the size of the smallest singular value for this matrix in the case of $\beta = 10^{-2}$.

3. What would be the complexity of solving this ordinary least squares system using dense QR factorization? Fortunately, we do not need to use dense factorization to solve this problem. I recommend using the sparse QR factorization in MATLAB; do `help qr` to learn more. Or you can just use backslash!

2.3 Landweber iteration

The *Landweber iteration* is a fixed point iteration of the form:
$$V^{(k+1)} = V^{(k)} + \alpha H^T (V^{\text{blur}} - HV^{(k)}).$$
For α sufficiently small, $V^{(k)} \to V^{(\infty)}$ as $k \to \infty$, where $H^T H V^{(\infty)} = H^T V^{\text{blur}}$. Thus, we should *asymptotically* recover the same solution generated by the naive approach. However, we can effectively regularize the problem by looking at *partly converged* results. You should implement this approach in a MATLAB function with the following signature:

```
% imresult = p2landweber(imblurd, H, n, alpha)
%
% Deblur the image using the Landweber iteration
% for n steps, starting from the blurred image.

function imresult = p2landweber(imblurd, H, n, alpha)
```

Questions

1. Show that the fixed point iteration converges to the solution to the normal equations whenever

$$0 < \alpha < \frac{2}{\sigma_1^2},$$

where σ_1 is the largest singular value of H.

Computing σ_1^2 can be expensive, but there is a cheap upper bound:

$$\sigma_1^2 \leq \|H\|_1 \|H\|_\infty.$$

In this particular case, $\|H\|_1 = \|H\|_\infty = 1$ and σ_1^2 turns out to be just a smidge less than one.

2. My script uses a default value of $\alpha = 2$. Play with this parameter. Do you get better results for other values?

3. My test script (`p2runner`) simply runs for 100 iterations. Build an alternate version of `p2landweber` that shows the images at each intermediate step of the iteration. At what step are you first able to read the text? You don't need to include your code, but ideally your writeup should include a picture of the reconstruction at the relevant step.

4. Compare the run time and image quality for the Landweber iteration for this problem (with $\alpha = 2$ and 100 iterations) to the Tikhonov regularized computation (using MATLAB's sparse direct QR solver).

2.4 LSQR iteration

The *LSQR iteration* is a Krylov subspace iteration that minimizes the residual for least squares problems or ill-posed linear systems. Like the Landweber iteration, LSQR tends to have a regularizing effect when it is not run all the way to convergence. MATLAB has an LSQR solver built in (type `help lsqr`), which you should use to implement the following function.

```
% imresult = p2landweber(imblurd, H, n)
%
% Deblur the image using the LSQR iteration for n steps.

function imf = p2lsqr(imblurd, H, n)
```

You will need to make three calls to `lsqr`, one per column of `imblurd`. You will probably want to specify a tolerance and a maximum iteration count. I recommend choosing a tolerance of 10^{-6} – you won't reach this tolerance within a hundred iterations, but this doesn't matter so much.

Questions

1. Compare the run time, image quality, and residual error (measured in Frobenius norm) for LSQR as compared to Landweber with $\alpha = 2$. Comment on what you see for 30 iterations and for 100 iterations.

3 Notes

1. The type of very ill-conditioned problems seen here occur frequently not only in image reconstruction and inverse problems, but also in solving "first kind" integral equations that occur in mathematical physics. Integral equations of the second kind tend to be much better behaved.

2. The Landweber iteration also works for ordinary least squares problems in which the matrix is tall and thin (rather than square problems of the sort described here).

3. One of the advantages of the Landweber and LSQR iterations which we have not highlighted in this assignment is that we do not actually need the matrix H in any explicit form. We only need the ability to multiply by H and H^T. This is particularly useful in medical imaging problems, where H can often be efficiently applied via fast Fourier transforms.