

## 6.581/20.482J Problem Set #6, Image Processing Part 1

**Due: Thursday, May 18, 2006, at 5 PM**

1. **Recovering Data from a Noisy Image.** The imaging system blurs the object data and adds noise, so our image capture setup receives the signal

$$g = Hf^0 + w \quad (1)$$

where  $f^0$  is the original object data,  $H$  is the point spread function,  $w$  is the added noise, and  $g$  is the captured image data. In this problem, we will explore the use of the singular value decomposition (SVD) to recover as much of the original signal  $f^0$  as possible without unduly amplifying the noise  $w$ .

- Run the script `load_data.m` to initialize the object data  $f$ , PSF  $h$ , and random white noise  $w$ . Recall from class that  $h$  is symmetric about  $h[0]$  and defined to be periodic with length  $n$  such that  $h[n-1] = h[1]$ ,  $h[n-2] = h[2]$ , etc. Generate the circulant matrix  $H$  and calculate  $g$ .
- Take the SVD of  $H$  and plot the first several columns of  $U$  and  $V$ . Explain what you see.
- Plot the singular values of  $H$  and observe that they appear in pairs. All circulant matrices exhibit this characteristic—why?
- Blindly deconvolve the resulting image data such that

$$f = H^{-1}g \quad (2)$$

and plot the result. As expected, the deconvolution amplifies the noise and drowns out the original object data. Computing the SVD of  $H$  gives us

$$H = U\Sigma V^T \quad (3)$$

$$H^{-1} = V\Sigma^{-1}U^T \quad (4)$$

where  $\Sigma$  is a diagonal matrix such that  $\Sigma_{i,i}$  describes the quantity of stretching that  $H$  performs when mapping from the input basis vector  $V_i$  to the output basis vector  $U_i$ . If  $\Sigma_{i,i}$  is very small, inverting  $H$  amplifies the component associated with  $U_i$ . Use `partial_inv.m` to only invert those directions of  $H$  that are associated with large singular values  $\Sigma_{i,i}$ . Try setting the threshold to different values and plot the resulting deblurred data. What threshold is optimal for recovering the original function?

- Generate two or three other signals of your choosing. Blur and add noise to these signals as you did in part a.). Use the optimal threshold in part b.) to recover the original signals. Now vary the threshold—is your answer from part b.) still the best?
2. **Modeling Discontinuities.** In class, we chose to use a Fourier representation of our image data. This problem explores one of the consequences of that choice of representation: discontinuities in the image data suffer from what is called the *Gibbs phenomenon*.

- Take the boxcar function  $x$  from `load_data.m` and find its Fourier transform  $X$  via `fft`. Truncate the Fourier representation at  $k = 10, 20, 40$ , where  $k$  is the number of nonzero Fourier coefficients. Plot the inverse transforms. Note that truncating requires a bit of care (this is not as straightforward as forming a low rank matrix via the SVD). The resulting signals should overshoot the original discontinuity. What changes as you increase  $k$ ? What doesn't change?
- Now let's use the object data  $f$  from Problem 1. Transform this smoothed boxcar function, and truncate its representation for the same values of  $k$  as in part a.) When you inverse transform these truncated representations, what do you see?
- If capturing sharp edges is important, a wavelet-like representation of the data might be more appropriate. Load the wavelet basis  $Q$  from `loadWaveletBasis.m`. Project the signals  $x$  and  $xf$  into this basis, and truncate the representation at  $k = k_1, k_2, k_3$  so that

$$x^{(k)} = Q(:, 1:k) * Q(:, 1:k)^T * x. \quad (5)$$

How do the functions change if we represent them using truncated wavelet expansions? Plot your results and comment on possible scenarios in which wavelet representation may be advantageous, and others in which the Fourier representation has the edge.

3. **Two-Dimensional Image Processing.** In this problem, you will explore various ways of doing a two-dimensional convolution between an object and a point spread function (PSF).

(a) Figure 1 shows a simple object and its image under convolution with a PSF. The non-zeros entries of both matrices (as shown in non-black pixels) are:

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 1 & 1 & \frac{1}{4} \\ \frac{1}{4} & 1 & 1 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

respectively. Can you infer what the PSF is? Verify it in MATLAB with

$$\text{abs}(\text{ifft2}(\text{fft2}(M_{obj}) \cdot \text{fft2}(M_{psf})))$$

where each matrix contains corresponding 8x8 pixel values.



Figure 1: Object and image in 8x8 pixels

(b) The two-dimensional convolution between an object and a PSF can be written in matrix form:

$$\begin{bmatrix} H(0,:) & H(n-1,:) & \dots & \dots & H(1,:) \\ H(1,:) & H(0,:) & \ddots & & \\ H(2,:) & H(1,:) & \ddots & & \\ \vdots & H(2,:) & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \\ H(n-1,:) & & & & H(0,:) \end{bmatrix} \begin{bmatrix} f^o(1,1) \\ \vdots \\ f^o(1,n) \\ \vdots \\ f^o(n,n) \end{bmatrix} = \begin{bmatrix} g(1,1) \\ \vdots \\ g(1,n) \\ \vdots \\ g(n,n) \end{bmatrix}$$

where  $f^o(i, j)$  and  $g(i, j)$  are object and image data in pixel  $(i, j)$ . The matrix is "doubly circulant" as each block entry itself is also a circulant matrix

$$H(k,:) = \begin{bmatrix} h(k,0) & h(k,n-1) & \dots & \dots & h(k,1) \\ h(k,1) & h(k,0) & \ddots & & \\ h(k,2) & h(k,1) & \ddots & & \\ \vdots & h(k,2) & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \\ h(k,n-1) & & & & h(k,0) \end{bmatrix}$$

whose entries are from  $k$ th row of a two-dimensional PSF. Construct the matrix with the PSF you find in part (a), and verify that you get the same image data as before by matrix multiplication.

(c) Multiplication by a circulant matrix  $H(k,:)$  can be efficiently computed with a one-dimensional FFT. In MATLAB notation:

$$H(k,:) * f = \text{ifft}(\text{fft}(h(k,:)) \cdot \text{fft}(f))$$

Implement two-dimensional convolution with one-dimensional FFTs, and verify that you get the same image data as before.

- (d) Download `mandrill.bmp` from course web site, and use your one-dimensional FFT implementation to get its convolution with the PSF. You should get a similar picture as shown in figure 2. You may have noticed that it appears brighter than the original image which is physically non-realistic. Normalize the PSF so that it does not amplify the energy of an input signal. Redo the simulation and comment on the difference between the original data and the convolved one.

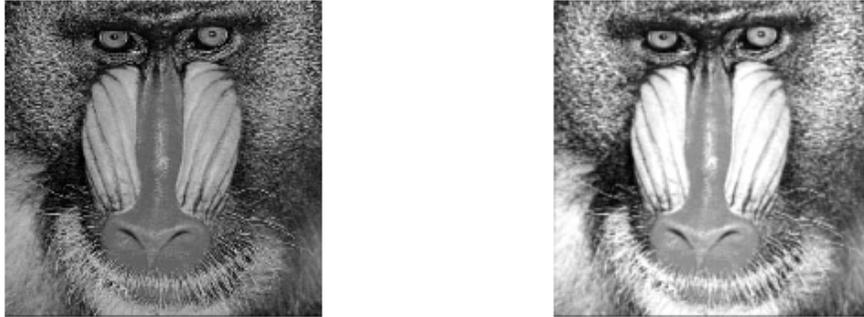


Figure 2: Image of a mandrill and its convolution with a PSF