

G80.3122-001/G63.2840.003 – Fall, 2009

Representation and Analysis of Visual Images

Homework 3

Due: 18 Dec 2009

Your results should be in the form of a MATLAB file (typically, the filename should have an extension of .m). Please email your solutions to `eero.simoncelli@nyu.edu`. Auxilliary matlab files are in the course homework directory:

`http://www.cns.nyu.edu/~eero/imrep-course/Homework/`,
also linked from the course web page.

Adaptive denoising via thresholding. We'll try several simple methods of denoising, to get a sense of how they compare.

Download and put in your path the `matlabPyrTools` package, available at `http://www.cns.nyu.edu/pub/eero/matlabPyrTools.tar.gz`

Download the file `einstein.pgm` from the course web page, and execute `im=pgmRead('einstein.pgm');`

1. Frequency domain

(a) Fourier transform the image (with `fft2`), and divide by the square root of the product of dimensions of the image (this is necessary to make it an orthogonal transform- matlab scales the forward and inverse transform differently). Look at the log of the squared amplitude, `fftshifted` so that the origin is centered in the middle of the image. Plot a horizontal slice of this, through the middle. What shape is it?

(b) Now construct an image containing the function $1/r^2$, where r is the distance from the origin of the Fourier domain (after it has been shifted to the center - make sure to get this location right!). Find a scale factor A for which this function best approximates (in terms of mean squared error) the squared amplitude computed above [hint: this is a standard linear problem].

(c) **Wiener solution.** Add Gaussian white noise to the image, with standard deviation σ equal to that of the image (this is very severe noise, but will allow you to see what's going on). Compute the Wiener filter: $\frac{A/r^2}{A/r^2 + \sigma^2}$. Plot a slice through the middle - does it look like you expect? Multiply the centered Fourier transform of the noisy image by this filter, and then unshift (call `fftshift` again) and `ifft`. Look at the result, and compute its Mean squared error (MSE).

(c) **Thresholding solution.** Now do the same, but use a binarized Wiener filter (set the filter equal to 0 when it is less than 1/2 and 1 when it is greater). The denoising result should be worse (larger MSE), since the Wiener solution is optimal for MSE. How much worse is it?

2. **Oriented pyramid.** Build a 2-band steerable pyramid on the noisy image you generated above, by calling `[noisyPyr, pind] = buildSFpyr=`, with a second argument equal to 1 (this is the order of the derivative). Display it using `showSpyr`. Note that what's shown here are all the subbands, *except* for the highpass residual (corners of the frequency domain).

(a) Band-wise adaptive Wiener estimate. Make a new variable containing a copy of the noisy pyramid [i.e., `cleanPyr = noisyPyr`]. This will be where you'll store your result. Loop through the pyramid bands letting n go from 1 to `size(pind,1)-1`. For each band, collect the indices of that band within the pyramid (`bandpyrBandIndices(pind,n)=`). Then the vectorized subband is simply `band=cleanPyr(band)`. Now compute the variance of the noise in that band, which is (due to subband scaling):

$$\text{noiseVar} = 0.15 * (\sigma^2) * \text{prod}(\text{size}(\text{im})) / \text{prod}(\text{size}(\text{band})).$$

One other ugliness: for band number 1, you need to multiply this noise variance by an additional factor of 4! Finally, compute a single scalar multiplier:

$$m = \max((\text{var}(\text{band}) - \text{noiseVar}), 0) ./ \text{var}(\text{band})$$

and multiply the band by this scalar. Collapse the pyramid to create a result image, using `reconSFpyr`. Look at the result (use `showIm` to display with the same scaling as the original image, so you can make a correct comparison!). What's the MSE?

(b) Point-wise adaptive thresholding. Now do the same thing again, but this time, multiply each band *pointwise* by a binary value:

$$m = \text{band}.^2 > 8 * \text{noiseVar}$$

Again, look at the result and compute the MSE.

(c) Local context-adaptive thresholding. Finally, do the same thing one more time, but this time, multiply each band pointwise by a binary value that is computed as in (b), but with the function `blur` called on the squared band:

$$m = \text{blur}(\text{band}.^2 > 4 * \text{noiseVar})$$

. Again, how's it look, and what's the MSE?

(d) Comparison. You've now computed five different denoising results. Sort them according to MSE, go down the list, and explain why, for each one, you think it does better than the previous one. Which one looks best (there's no right answer here!).