NEURL-GA 3042.005 – Spring, 2014

Representation and Analysis of Visual Images

Homework 3

Due: 13 May 2014

Your results should be in the form of an executable MATLAB file (typically, the filename should have an extension of `.m`), divided into sections by triple comments (%%%). Any functions you write should be included as separate m-files. Please email your solutions to `eero.simoncelli@nyu.edu`. You'll need a few functions from `www.cns.nyu.edu/~lcv/software.php`. 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.

1. **Pyramid denoising.**

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

   Download the file `nycImage.jpg` from the course web page, and execute `im=mean( double( imread('nycImage.jpg')), 3);`

   Add white Gaussian noise with variance σ^2 equal to 0.25 of the image variance (i.e., standard deviation equal to 0.5 the image standard deviation).

   Now build a 2-band steerable pyramid on the noisy image you generated above `[noisyPyr, pind] = buildSFpyr(noisyIm, 5, 1)`, with the third argument indicating the order of the derivative. Display it using `showSpyr`, which shows 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` (everything except the lowpass band). For each band, collect the indices of that band within the pyramid (`bind=pyrBandIndices(pind,n)`). The vectorized subband is simply `band=cleanPyr(bind)`. The variance of the noise in that subband is proportional to the variance of the noise in the image, where the scale factor depends on the filters used in the pyramid and the subsampling:

   $$\mathrm{noiseVar} = 0.154 * \sigma^2 * \mathrm{prod}(\mathrm{size}(\mathrm{im}))/\mathrm{prod}(\mathrm{size}(\mathrm{band})).$$

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

   $$m = \max\left(\mathrm{var}(\mathrm{band}) - \mathrm{noiseVar}, 0\right)/\mathrm{var}(\mathrm{band}),$$

   and multiply the band by this scalar. Collapse the pyramid to create a result image, using `reconSFpyr(cleanPyr)`. Look at the result (use `showIm` to display with the same scaling as the original image, so you can make a correct comparison!). Also compute the mean squared error (you might find `imStats` useful).

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

$$m = \max\left(\text{band.}^2 - 4 * \text{noiseVar}, 0\right) ./\text{band.}^2$$

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 value that is computed based on a *local* estimate of variance:

$$m = \max\left(\text{av}(\text{band.}^2) - 2 * \text{noiseVar}\right) ./\text{av}(\text{band.}^2)$$

where av is the average over a 5×5 patch, computed with `corrDn`, using `reflect1` edge handling. Note: You'll need to reshape each subband back to an image before computing this average, and then reshape back to a vector to insert it into `cleanPyr`. Again, how's it look, and what's the MSE?

(d) **Comparison.** Which of the three denoising results gives best MSE? Which looks best (in your opinion) and why?

2. **ICA and sparsity**

   (a) **PCA/whitening.** Compute the principal components of 10×10 blocks taken from the image in the previous problem. Gather the full set of blocks (with overlap) into a matrix (X) using im2col with the 'sliding' argument. Compute the eigenvectors (E) and eigenvalues (D) of the covariance matrix of the blocks, and then compute the whitening matrix $W = D^{-1/2}E^T$. (Note: Might be more efficient to do this using the SVD). Compute the "basis functions", which are the columns of W^{-1}, and look at the first 25 of these, as 10×10 images, in a single window (use `subplot(5,5,n)`). What do they look like? Apply the whitening matrix to X, and then compute the kurtosis (use matlab's function) of each component. What is the geometric mean, and the range of the kurtosis values?

   (b) **Symmetric whitening.** As discussed in class, the whitening matrix is non-unique, since pre-multiplying by any orthogonal matrix does not affect the whitening property. One way to make a unique whitening matrix is to demand that it be symmetric: $W_{\text{symm}} = ED^{-1/2}E^T$. Look at the first 25 basis functions (columns of W^{-1}, displayed as images) - what do you see? Apply this whitening matrix to X, and compute the kurtosis of each component. What is the geometric mean, and the range of the kurtosis values?

   (c) **ICA.** Now compute the ICA transform on blocks X, using the `ica4` function in the course directory. (Alternatively, you can try the fastICA algorithm, which can be downloaded from the internet). The function returns the ica-transformed data, and the ICA transform matrix W. Again, compute and look at the basis functions (cols of W^{-1}). Apply W to X and measure the kurtosis of each component, computing the geometric mean and range of values.

   (d) **Comparison.** Plot the sorted kurtosis values from all three transforms on the same plot. For comparison, compute the sorted kurtoses of the components arising from a random transform (`W = randn(100)`). What do you see?

3. **Contrast enhancement**

   (a) Build a Laplacian pyramid on the image from part 1 (`buildLpyr(im);`). A simple method of contrast enhancement is to multiply each subband by a scalar that grows or shrinks geometrically with scale. That is multiply subband n by α^{N-n}, where N is the total number of subbands (`size(pind,1)`). Try doing this for α values of 0.9 or 1.1, reconstruct a modified image (using `reconLpyr`), and look at it. What do you see? Which α improves the image?

   (b) Multiplying the whole band by a constant is not so effective, since it boosts/shrinks all coefficients by the same amount. A better solution is to boost the small-amplitude coefficients more than the large ones, using a saturating nonlinear function. For each subband, measure the initial standard deviation, σ. Then apply the function $\mathrm{sign}(x)|x|^p$, for exponent $p = 1/2$. Finally, restore the standard deviation, by dividing the result by its standard deviation, and multiplying by the standard deviation you measured before the transformation. Reconstruct the modified image and look at it. Try this for different p, ranging from $1/4$ to 2. What do you prefer?