NEURL-GA 3042.005 – Spring, 2014

Representation and Analysis of Visual Images

Homework 1

Due: 4 Mar 2014 (part 3 (Wiener) due 11 Mar)

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. 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.

Trichromacy

Load the file `colmatch.mat` in your MATLAB environment. This file contains a number of matrices and vectors related to the color matching experiment presented in class. In particular, the variable `P` is an $N \times 3$ matrix containing wavelength spectra for three basis lights (sometimes called "primaries"), and the variable `M` is a $3 \times N$ color-matching matrix corresponding to these primaries. For these problems $N = 31$, corresponding to samples of the wavelength spectrum from 400nm to 700nm in increments of 10nm.

Applying M to any light $\vec{l}$ produces a 3-vector that contains the intensities of the three primaries that, when combined, would appear the same as $\vec{l}$ to a human observer. That is, $P * M * \vec{l}$ appears the same as $\vec{l}$. More generally, two lights $\vec{l}_1$ and $\vec{l}_2$ will match in color appearance if and only if $M\vec{l}_1 = M\vec{l}_2$.

1. Create a light with a random wavelength spectrum, by generating a random column vector with all positive components (use `rand`). What combination of the three primaries in `P` will match the appearance of this randomly chosen test light? Compute the (N-dimensional) wavelength spectrum of this combination of primaries, and verify that it satisfies the general "matching" criterion described above. Plot it together with the original light spectrum, and explain why the two spectra are so different, even though they would appear the same to a human.

2. The variable `Phosphors` contains the emission spectra of three standard color display phosphors (it's an old-fashioned cathode ray tube!). Suppose you wanted to make the background color of this screen match the appearance of the light you generated in the previous problem. Write a matlab expression to compute the three intensities that should be for the phosphors. Verify that this mixture of phosphor spectra will look correct (i.e., that it satisfies the "matching" criterion described above)

3. The variable `Cones` contains (in the rows) approximate spectral sensitivities of the three color photoreceptors (cones) in the human eye: `Cones(1,:)` is for the L (long-wavelength, or *red*) cones, `Cones(2,:)` the M (green) cones, and `Cones(3,:)` the S (blue) cones. Applying the matrix `Cones` to any light $\vec{l}$ yields a 3-vector containing the average number of

photons absorbed by that cone. Plot the three Cone sensitivities using `plot`. Verify that the cones provide a physiological explanation for the matching experiment, in that the cone absorptions are equal for any pair of lights that are perceptually matched. Specifically, give an informal proof (write the linear algebra, and then test it in MATLAB). Hint: Think about null spaces.

LSI Systems, 2D Fourier

1. Create a random image of size 32x32, and a random filter of size 5x5 (use the function `rand`). Convolve them using the function `cconv2` (provided). Verify the convolution theorem: Show that the same result may be obtained by `fft2`ing, multiplying, and `ifft2`ing back. Be careful with indexing (i.e., make sure you know where the origin is, in both the spatial and frequency domains)! Try this with the MATLAB function `conv2`, using the optional second argument `'same'`. Why doesn't it work?

   Now consider handling signal boundaries by reflection about the border sample, implemented in the function `rconv2` (provided). Create a random *symmetric* filter of size 7x1 (a column v vector). Show that the expression `rconv2(eye(16),filt)` produces a matrix that, when applied to a 16x1 input vector `v`, can generate the same output as `rconv2(v,filt)`. Compute the eigenvectors of this matrix (using the MATLAB function `eig`). Pick one and *verify* that it is an eigenvector. Are the eigenvectors sinusoidal? Why, or why not?

2. Write a function `mksine2` that takes arguments `xsize, ysize, amplitude, freq, phase` and returns an array of the specified size containing samples of a 2D sinusoid. `freq` should be vector of length 2, specifying frequency in the `x` and `y` directions (in radians). Remember that MATLAB 's indices start from 1. Hint: For efficiency, you should begin by creating x- and y-ramp images using the MATLAB function `meshgrid`. Look at the sinusoids that result from your function for frequencies $(\omega, 0)$, as ω runs from 0 to 2π. What happens to the sinusoid beyond $\omega = \pi$? Do the same thing for frequencies (ω, ω). NOTE: when displaying images, you should *always* use the matlab function `truesize` to guarantee that the number of screen pixels used to display the image is either an integer multiple of the image size, or the result of dividing the image size by an integer and discarding the remainder (explain why).

3. Examine the discrete Fourier transform of some example sinusoids produced by your function. Why is the result not always confined to a single frequency?

Wiener Filtering

Use `imread` to load the image `testIm.jpg` into your matlab environment, and convert it from unsigned integers to floating point using `double`.

1. 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, since matlab scales the forward and inverse transform differently). Look at the log of the squared amplitude, `fftshift`ed so that the origin is in the middle of the image (note: the indices of this

location are `ceil((size(im)+0.5)/2))`. Plot a horizontal slice through the middle of the log amplitude. What shape is it?

2. Now construct an image containing the function $1/r^p$, with $p = 2.0$, where r is the distance in pixels from the origin of the Fourier domain (after it has been shifted to the center - make sure to get this location right!). You may use the provided function `mkR` for this. Now find a scale factor A for which A/r^p best approximates the squared Fourier amplitude computed above, in the sense that it has the same mean value over all frequencies. That is, set A equal to the ratio of the mean of the squared Fourier amplitude to the mean of your $1/r^p$ image. When taking the mean of the two images, don't include the value at the origin, which is infinite (hint: use `isfinite` to gather the indices of all finite values). Plot a horizontal slice of the log of A/r^p on top of the plot from the previous problem, to make sure the fit seems reasonable.

3. **Wiener estimate**. Add Gaussian white noise to the image, with standard deviation σ equal to half that of the image (this is very severe noise, but will allow you to see what's going on). Now compute the squared Fourier amplitude, and fit it with the function $\frac{A}{r^p} + \sigma^2$, by subtracting σ^2 and proceeding as in the previous problem. Since A must be positive, you should set any negative results to 0. Now compute the Wiener filter: $\frac{A/r^p}{A/r^p+\sigma^2}$. Plot a slice through the middle of this - does it look like you expect? Multiply the centered Fourier transform of the noisy image by this filter, unshift (with `ifftshift`) and invert the FFT (with `ifft2`). Look at the result, and compute its mean squared error (MSE). How well does it work?

4. **Block Wiener estimate**. Apply the same methodology independently to non-overlapping square blocks of the image. You can use the function `im2col` with the `'distinct'` argument to gather the blocks as columns of a large matrix, loop over these columns, reshaping them into little images and denoising them, then vectorizing them and storing them as columns of a new matrix which can be passed to `col2im` to re-assemble them into a full-sizez image. Try this for square blocks of with dimensions $\{2^M : M = 2, 3, ...7\}$, measuring the MSE for each block size. What block size gives the best MSE? Why is the result worse for larger and/or smaller block sizes? Look at each of the denoised images. What kind of artifacts do you see in the results, and which block size produces the best-looking result?