Natural Images and Your Brain

Corresponding MATLAB script is natural_images.m
Formerly a function called barlow_filt3.m: [Ivert,Ihor] = barlow_filt3(I,sigma,npix,ndir,nbins,myalpha);

RTB wrote it, Feb. 2008; RTB adapted for QMBC May 2011

MATLAB concepts covered:
1. loading and displaying images
2. histograms and probability density functions
3. correlation
4. imfilter
5. colfilt

Reference: Barlow H (1994) "What is the computational goal of the neocortex?" in Large-scale neuronal theories of the brain, C Koch and JL Davis, eds. (Cambridge, Mass.: MIT Press, 1994, pp. 1-22).

In this exercise, we are going to play with some natural images and filter them in ways that mimic the early stages of visual processing in the primate brain. These filters are ones with which you are already familiar, but we are going to approach them from a different perspective: the ideas of "redundancy reduction" and "efficient coding"; ideas first developed by Horace Barlow. In the process, we will look at some statistical properties of natural images and what they imply for the kind of filtering the visual system performs.

First, we need an image to work with. As Mike showed us, we could just "drag and drop" the image into the MATLAB workspace, but it's also useful to know how to do it the old-fashioned way. Why? This is a color image, but we're going to convert it to a gray-level image to make our calculations somewhat simpler. The same principles apply to color images as well, though as a quick demo may convince you that most of the information about shape is conveyed by its luminance.

Ex. 1: Load the image "sophie.jpg", convert it to a gray scale image and display it.

default values for analyses:

```
myalpha = 0.01;      % significance level for determining "suspicious coincidences"
ndir = 2;            % number of orientations we're sampling (V + H = 2)
npix = 9;            % # of pixels to sum for the Barlow Filter
sigma = 1;           % width of the Gaussian filter in pixels
```

One of the first major principles about natural images is that they are highly redundant. One way to see this is that some pixels occur much more frequently than other pixels; that is, any given gray level is not equally likely.

Ex. 2: Use 'imhist' to display the image histogram, then convert this to a probability density function. Hint: The sum of all the probabilities must sum to one.

Another way to see this is to ask how many of the pixels might we randomly screw up and still be able to make out the meaning of the picture? (Claude Shannon played a similar game with language.)

Ex. 3: Write a function that will randomly substitute for a fraction, p, of the pixels with random values.

What proportion of the image do you have to degrade before it is no longer recognizable?

Not only are the pixel probabilities not random, but we can see that there tend to be large regions of the same basic color. This means that if a pixel is a given color, chances are that its neighbor has the same color. How might we show this?

Ex. 4: Make a plot comparing the value of each pixel with that of its immediate neighbor to the right.

How can we summarize this relationship?

Ex. 5: Write a function that will accept an image, I, as input and plot the relationship between distance and correlation over the range from 1 to 40, returning the correlation coefficients as a row vector.

Yet another way to examine this relationship is by looking at the relative energies at different spatial frequencies by using the Fourier Transform: sfPlot(I,1); What does this tell us?

Now we're going to do some filtering that approximates the early stages of visual processing. The first step the center-surround operator found in the retina. This is typically modeled in the physiology literature as a "Difference of Gaussians" (or "DoG"). That is, the excitatory center is modeled as a Gaussian with a relatively small sigma, and the inhibitory surround as Gaussian with a larger sigma. This type of operator is well approximated by a filter that you have already met: the 'Laplacian of Gaussian' or 'log' filter. At this point, you should be able to simply code up a "DoG" using 'normpdf' and compare it to a 1D 'log'.

Ex. 6: Filter the image with a "log" filter of size (ceil(sigma*3) * 2 + 1;) and display it. Play around with sigmas of different sizes.

Now let's look at it in the frequency domain. This function (which is beyond the scope of the boot camp) makes a plot of the "energy" in the image at different spatial frequencies. The key function is 'fft2', which performs a 2-dimensional Fourier Transform of the image. Open this function and look at the intermediate result known as the "Power Spectrum". What does this show?

We might notice one other feature about our natural images: the pixels making up edges tend to fall along contours that run in relatively straight lines for many pixels. These sorts of features would almost never occur in random images, but they occur quite frequently in real images—Barlow referred to these as "suspicious coincidences" and argued that this sort of feature should be explicitly represented in the cortex.

Ex. 7: Make histograms of the following image measurements: 1) the sum of 9 contiguous pixels in either the vertical or horizontal direction vs. 2) the sum of 9 pixels chosen randomly.

What do you notice about the two distributions? Which one has longer tails? Where are the "suspicious coincidences"?

Ex. 8: Select the values in the tails and find their locations in the original image. Overlay the regions, making the vertical coincidences RED and the horizontal ones GREEN.

Repeat this exercise with different images: 'trees' or 'durer'.