Natural Images and Your Brain

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

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

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. Sophie.jpg 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 most of the information about shape is conveyed in the luminance.

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

Instead of 'dragging and dropping" use imread. See Step 1 in natural_images.m for help
Why might using imread be important?
Convert the image to gray scale and to double.

Step 2: Visualizing image redundancy.

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

Visualize this property by creating a histogram of the pixel intensity values using imhist.
Convert these data into a probability density function.
Hint: The sum of all the probabilities must sum to one.
See Step 2 in natural_images.m for help

Step 3: Another way to visualize image redundancy: add noise

Another way to demonstrate that natural images are redundant is to ask how many of an image's pixels can be randomly changed while still allowing for us to make out the meaning of the picture? (Claude Shannon played a similar game with language.) See Step 3 in natural_images.m for help

Write a function that will randomly substitute for a fraction, p, of the pixels with random values.
HINT: First, randomly select pixels. Next, Randomly change their value.

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

Step 4: Neighboring pixels are correlated

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

How might we show this?
How can we summarize this relationship?
Make a plot comparing the value of each pixel with that of its immediate neighbor to the right and summarize this relationship.
See Step 4 in natural_images.m for help

Step 5: What is the relationship between pixel distance and correlation amongst pairs of pixels?

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 100, returning the correlation coefficients as a row vector.
Plot the relationship between correlation and distance.
See Step 5 in natural_images.m for help
Also see im_corr.m

What happens to the correlation as distance increases? What does the shape of that result resemble? Hint: what 'color' noise is this?

EXTRA: Modify your code so that we perform the correlation of each pixel with ALL of its neighbors at a given distance.

Step 6: Redundancy in the frequency domain

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 output tell us? How is the information in this plot related to the previous plots that showed the correlation between intensity values of neighboring pixels?

Step 7: Approximating the retina

Now we're going to do some filtering that approximates the early stages of visual processing. The first step is 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 standard deviation, and the inhibitory surround as a Gaussian with a larger standard deviation. This type of operator is well approximated by a filter that you have already met: the 'Laplacian of Gaussian' or 'LoG' filter (not to be confused with logarithm).

Start by specifying a standard deviation:

```
sigma = 1;     % width of the Gaussian filter in pixels
```

*Filter the image with a "log" filter of size (ceil(sigma*3) * 2 + 1;) and display it. Use the 3D rotate button in the plot window to inspect the function. Play around with different sizes of sigma to see how the filter changes.*
See Step 7 in natural_images.m for help

Apply this filter to our image and view the result. Is the image hard to see? If so, how can you improve it? What has this filtering done?

Now let's look at this image in the frequency domain. The function, sfPlot, (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 it calls is 'fft2', which performs a 2-dimensional Fast Fourier Transform of the image. Open this function and look at the intermediate result known as the "Power Spectrum".

What does this show?
How does the output of sfPlot look different from the previous frequency plot? Why?

Step 8: Edges: Another form of redundancy

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.

Make histograms of the following image measurements:

1) The sum of 9 pixels chosen randomly. See Step 8 in natural_images.m for help
2) The sum of 9 contiguous pixels in either the vertical or horizontal direction.

First, assign few default values for your 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
```

What ways can you come up with for finding the random pixels?
After finding these random values, try finding the contiguous pixels using 'for' loops.
Use tic and toc to determine how long this code takes to run.

Now, try using using colfilt for the selection of contiguous pixels.
How does colfilt work?
How long does it take to run?

Calculate and display a statistical threshold based on the image statistics to define the "suspicious coincidences."
What do you notice about the distributions from 1) and 2)? Which one has longer tails?

Instead of the statistical threshold, we could use the limits of the random distribution to define the tails of the contiguous distribution. Try this. What is the corresponding 'myAlpha' value for this?

Step 9: Display the "suspicious coincidences"

Select the values in the tails from the contiguous distribution and find their locations in the original image. Overlay the regions, making the vertical coincidences RED and the horizontal ones GREEN.
See Step 9 in natural_images.m for help

Extra Step 10: Repeat this exercise with different images

Do you see a similar fall off of the correlation with different images in Step 5?
How do the distributions from Step 8 change between images?
What other changes do you observe at each step? Can you predict these changes based on the appearance of each image?

Version History:
RTB wrote it, Feb. 2008; RTB adapted for QMBC May 2011
DAR modified it, August 2011 as two files. This pdf and corresponding "natural_images.m"