NENS 230
Assignment #8

Signal and Image Processing

Due: Tuesday, November 17th, 2015

Goals

- Sharpen an image using a `imfilter` and `fspecial`
- Understand how to use the Fast Fourier Transform (FFT) to view signals in the frequency domain
- Analyze real world signals

Problem 1: Image Sharpening

In this problem we will process the retina image that we used in lecture. Specifically, we will try to automatically extract cell bodies as regions of interest. To begin, load the `retina.jpg` image using `imread`, flatten the different color channels using `rgb2gray` and convert the matrix to a double using `double`. (Note for this problem that you will need the Image Processing Toolbox. This toolbox, if you do not have it, is available on the Corn machines. Let us know if you have trouble getting access to the toolbox.)

(a) Thresholding

Let's try and see if we can automatically select cell bodies as regions of interest. We'll first try doing this with thresholding. Save a new image which is a thresholded binary version of the retina image (set all values in the matrix that are greater than some threshold to 1, and the rest to 0). Use a threshold of 100. If you image this new matrix, you should only have two colors (corresponding to 0 and 1).

(b) Smoothing + Thresholding

Notice how we have some noise (small spots that aren't cell bodies) in the thresholded image. To deal with this, we are going to preprocess the image before thresholding. The preprocessing step we will use is to smooth the image by filtering it with a Gaussian filter. Use the `fspecial` command to make a Gaussian filter. Then, filter the original image (*before* you threshold it) with the `imfilter` command. Now threshold the image, again using a threshold of 100. Plot the figures from part (a) and (b) on the same figure (using `subplot`) for comparison. Does smoothing help get rid of some of the noise? Try varying the size of the filter and the standard deviation of the Gaussian filter.

(c) Bonus: Better method?

Try to come up with a different processing step that helps with our automatic detection of regions of interest. If you do this, discuss why your method is better at detecting cells. (For inspiration, you can try looking at some of the other types of filters from `fspecial`). This part is not required to get full credit for the problem.

Problem 2: Fourier Transform

In this problem, we will look at the fourier spectrum of some simulated EEG traces. Load the data in `sim_eeg.mat`. In a 4x2 subplot (4 rows, 2 columns), plot the 4 signals side by side with the fourier transform of the signals (signals on the left, fourier transform on the right). The `plotfft.m` function that we wrote in lecture (and posted under the Lecture 7 materials) will be useful for this problem, although you may want to modify it. Plot the axis of the fourier transform from 0 to 50Hz on the x-axis. The titles for the four signals are, in order: ”Background Activity”, ”Sporadic Spikes”, ”Rhythmic Activity”, and ”Slow quasi-sinusoidal activity”; add these as titles to the corresponding plot. Comment on differences in the fourier spectra of the different traces. You should make this plot using a `for` loop to loop over the four different traces, rather than copying and pasting code.

Problem 3: ECG

In this problem, we will use electrocardiogram (ECG) data recorded from frogs. In the experiment, electrodes were placed around a frog’s heart, and ECG waveforms were recorded. In addition, a couple of drugs (epinephrine and atropine) were administered, to see the effect of these compounds on the ECG waveform.

(a) Getting the data

Load the data, found in the file `ecg.mat`. The data is stored as three different Matlab structures, one corresponding to each condition of the experiment: normal (`norm`), epinephrine (`epi`), and atropine (`atro`). Each structure contains the following fields: the time vector in minutes (`time`), the voltage recording in volts (`V`), and the sampling frequency in Hertz (`fs`). Make a 3x1 subplot of the voltage traces from each of the different conditions. Label the axes, and title each subplot based off the condition.

(b) Peak detection

Write a function called `peakanalysis.m` that takes as input three arguments: a time vector, a voltage vector, and a threshold. The function should compute the local minima and maxima of the voltage vector using the `peakdet` function described in class (taken from FileExchange, and included in the assignment materials). You may use another method of extracting peaks if you wish. The function should return four vectors: the times of the minima, the values of the minima, the times of the maxima, and the values of the maxima. The function should also make a plot of the input trace with the maxima and minima plotted on top of the input trace. In addition, the function should put the value of the threshold used in the title of the plot. An example of a plot that the function should return is shown in Figure 1.

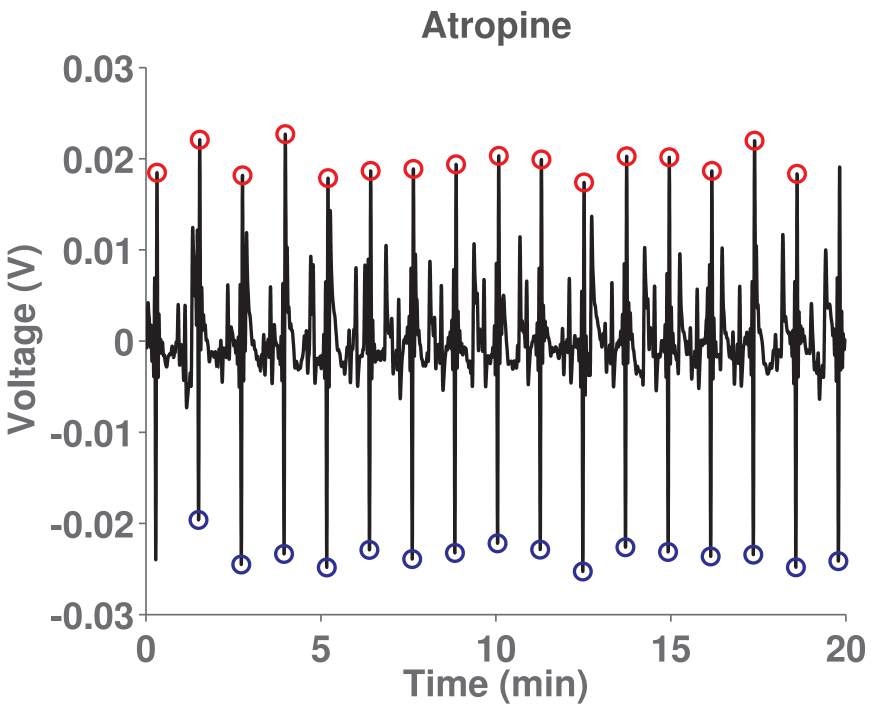

Figure 1: Peak Detection example

(c) Classification

Use your newly created function to extract the peaks and troughs of the ECG waveforms for the three conditions. You may have to choose different thresholds for the three different conditions. Use the plot generated by your function to gauge whether or not your threshold is good (you should have roughly one maximum and one minimum for each heartbeat. Once you have the extrema for each of the three conditions make a scatter plot of the maximum voltage vs. minimum voltage. Plot the values for the three conditions with different markers and colors. Are you able to distinguish the three conditions based off of the waveform extrema (are the clusters separated)?

Submission

When you are finished, publish your work with the `publish` command. If that doesn't work for you, feel free to create a zipped folder with your code and resulting plots. Email the generated PDF file along with your code, or your zipped folder, to `nens230@gmail.com` with `[Assignment 8]` in the subject line to submit your assignment.