Perception and Attention
G89.2223
David J. Heeger

V1 and Direction Selectivity Assignment

Use Matlab and the motion tutorial to do the following calculations and to answer the questions. Write up a report that explains your solutions, including graphs, and the relevant snips of Matlab code. For each question, please write a brief explanation of what you did, including any equations that you used to do the calculations, and write a brief interpretations of your results. Please submit a single pdf file (not MS Word) that contains everything.

1) Implement a recursive temporal filter. In class, we've focused only on using convolution for shift-invariant linear filters. But there's another way to do it with feedback in which the weights of the convolution kernel are implicit. These are called infinite impulse response (IIR) filters. Take a look back at the Signals, Linear Systems, and Convolution handout, near the end (p. 13-15), for an example. We'll be using a cascade of exponential low-pass filters. An exponential low-pass filter is computed using this equation:

$$\tau \, dy_1/dt = -y_1 + x$$

where $x(t)$ is the input and $y_1(t)$ is the output. The value of τ is the time constant of the exponential. To implement this, you need to first specify a time step, `deltaT`, for discretely sampling time. Use real units, in this case ms. Use `deltaT=1` ms and `tau=25` ms and compute the impulse response. Your code should look something like this:

```
deltaT = 1; % ms
duration = 1000; % ms
t = [0:deltaT:duration-deltaT]
x = zeros(size(t));
x(100)=1;
y1=0;
y1Save = zeros(size(t));
tau = 25; % ms
for tt = 1:length(t)
   deltaY1 = (deltaT/tau) * (-y1 + x(tt));
   y1 = y1 + deltaY1;
   y1Save(tt) = y1;
end
```

Note that y_1 has no memory. It updates itself based on the current values of y_1 and x. But then you can't plot the resulting time course of y_1 without saving it, and that's the point of the `y1Save` variable.

a) Compute and plot the impulse response for two different values of τ. The result should be an exponential function $e^{-(t/\tau)}$ so plot this exponential as well on the same graphs.

b) Compute and plot the step response, also for two different values of τ. The result should ramp up exponentially: $1 - e^{-(t/\tau)}$. So plot this as well on the same graphs.

c) Compute and plot the responses to sinusoidal inputs of various temporal frequencies. The frequency response for this IIR filter is derived in the handout (p. 14-15). Use those formulas to compute what the amplitude and phase of the responses ought to be for each of the temporal frequencies.

2) Implement a cascade of exponential low-pass filters such that:

$$\tau \, dy_n/dt = -y_n + y_{n-1}$$

where $y_0 = x$. Then compute the differences: $f_1=y_3-y_5$ and $f_2=y_5-y_7$. Plot the impulse responses of f_1 and f_2 . They should look familiar, similar to the temporal filters in the Adelson and Bergen (1985) paper. You should have a code segment that looks like this:

```
for tt = 1:length(t)
   deltaY1 = (deltaT / tau) * (-y1 + x(tt));
   y1 = y1 + deltaY1;
   deltaY2 = (deltaT / tau) * (-y2 + y1); y2 = y2 + deltaY2;
   y2 = y2 + deltaY2;
   % Continue with y3, y4, y5, y6, and y7
   f1 = y3-y5;
   f2 = y5-y7;
end
```

3) Implement a set of space-time filters and motion energy filters. The input will be a 3D stimulus array (x,y,t). As was done above with time, you need to specify the spatial sampling in real units. Use Use `deltaX=1/120` deg of visual angle because that's about equal to the cone spacing in the fovea (30 sec of arc). And make the input stimulus array cover -2 to 2 deg spatially and 0 to 1000 ms. Extend the code from #2 so that f_1 and f_2 are images, i.e., simulated neural images of responses. During each time step, update both f_1 and f_2 and then do series of spatial convolutions. Use 4 Gabor filters, two vertical and two horizontal, with odd and even phases. A Gabor filter is a sinusoid multiplied by a Gaussian window. Use 4 cyc/deg for the preferred spatial frequency and use $\sigma=0.1$ deg for the spread of the Gaussian window. You'll end up with 8 output arrays, each of which will be equal in size to the input array (2 temporal filters x 2 orientations x 2 phases).

Note that you should scale the Gabor filters so that the sum of the squares of the filter weights equals 1. Something like this:

```
evenFilt = exp(-(x.^2)./(2*sigma^2)) .* cos(2*pi*sf*x);
oddFilt = exp(-(x.^2)./(2*sigma^2)) .* sin(2*pi*sf*x);
integral = sum(evenFilt.^2 + oddFilt.^2);
evenFilt = evenFilt / integral;
oddFilt = oddFilt / integral;
```

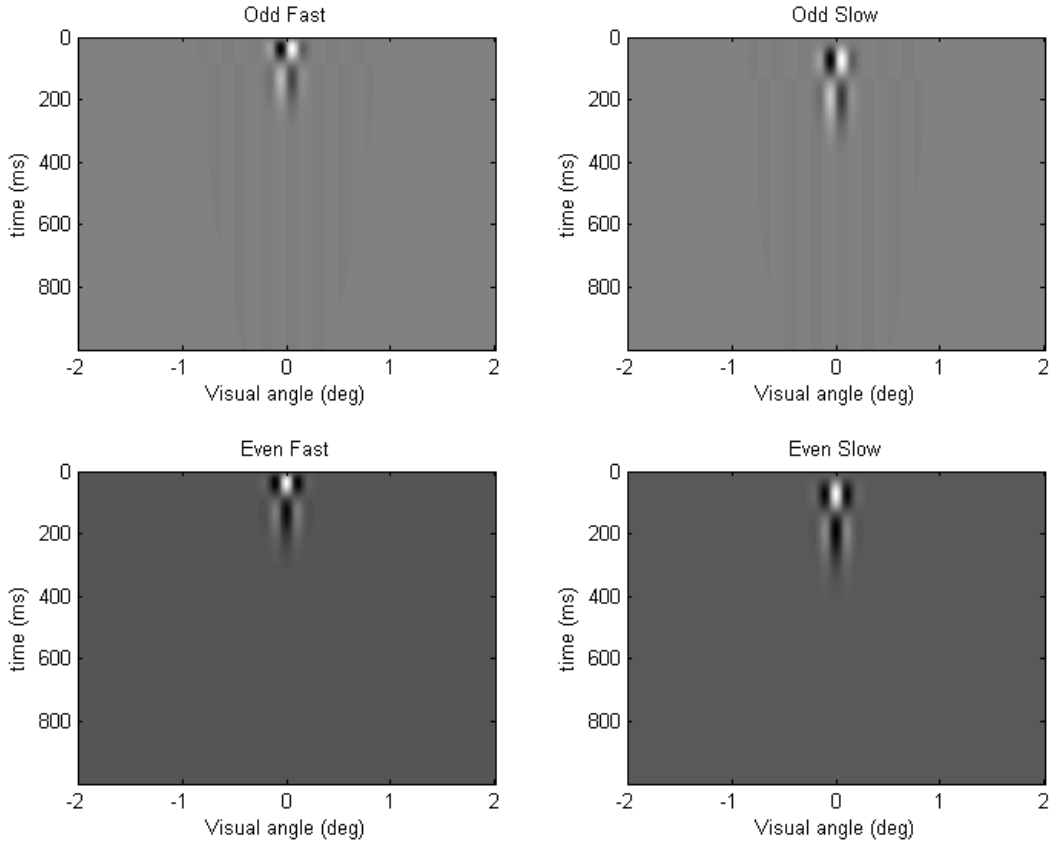
Doing so will make it easier when you get to parts 3d and 4.

Note also that you might choose to use a larger sample spacings such as `deltaT=10` ms and `deltaX=0.05` deg for debugging.

Hint: Remember that in matlab, you can treat arrays the same way that you treat numbers. See above code segment for #2. For #3, this code will be almost exactly the same except that each of the variables (`y1`, `y2`, ..., `f1`, `f2`) will each be an image. The only difference will be that you'll have to extract a single frame from x:

```
deltaY1 = (deltaT / tau) * (-y1 + x(:,:,tt));
```

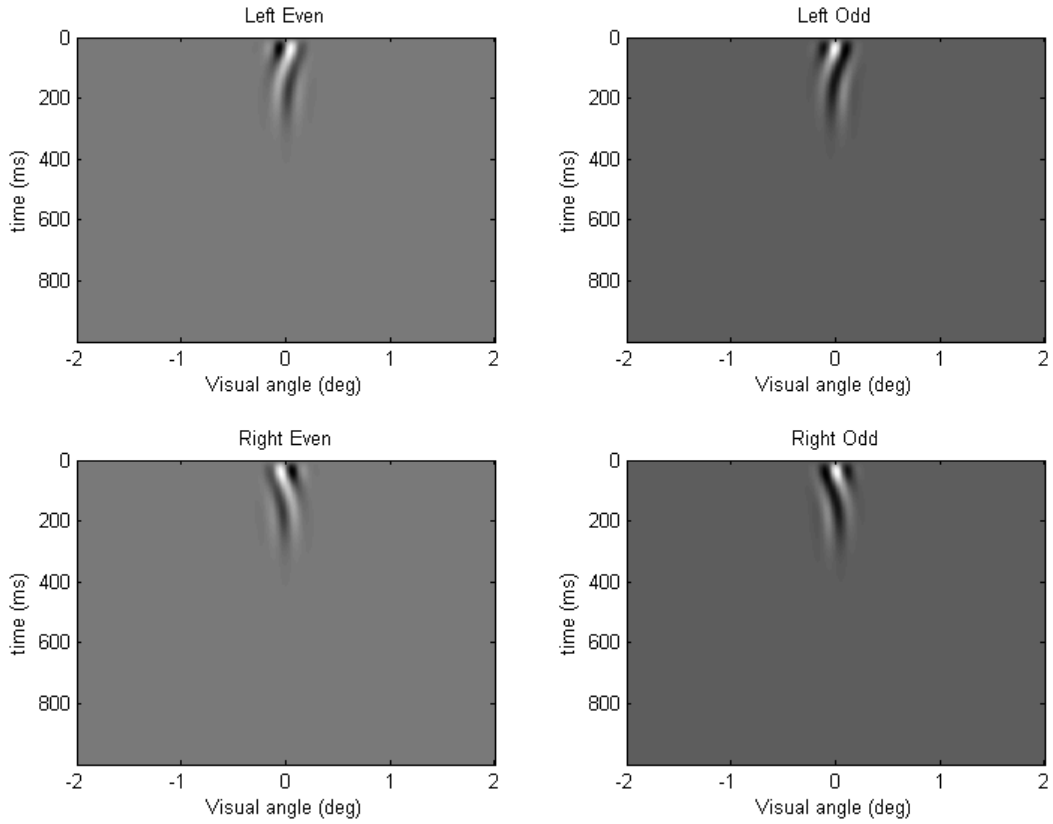
a) Display x-t slices of the impulse responses of the 4 filters that prefer vertical. Your results should look something like this:



b) Compute the appropriate sums and differences (according to Adelson and Bergen) to get space-time oriented filters:

```
leftEven = oddFast + evenSlow;
leftOdd = -oddSlow + evenFast;
etc.
```

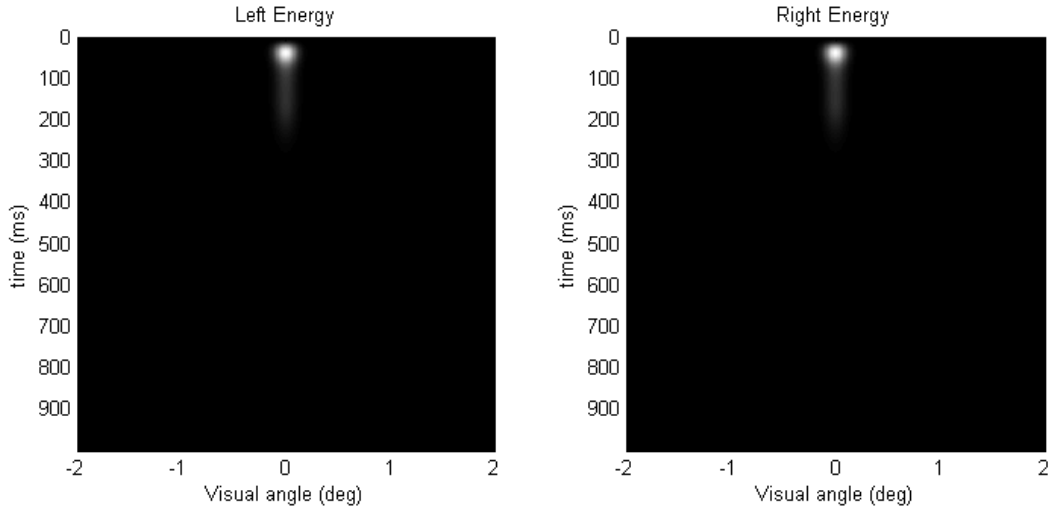
and display x-t slices of the impulse responses of the 2 leftward- and the 2 rightward-selective filters. Your results should look something like this:



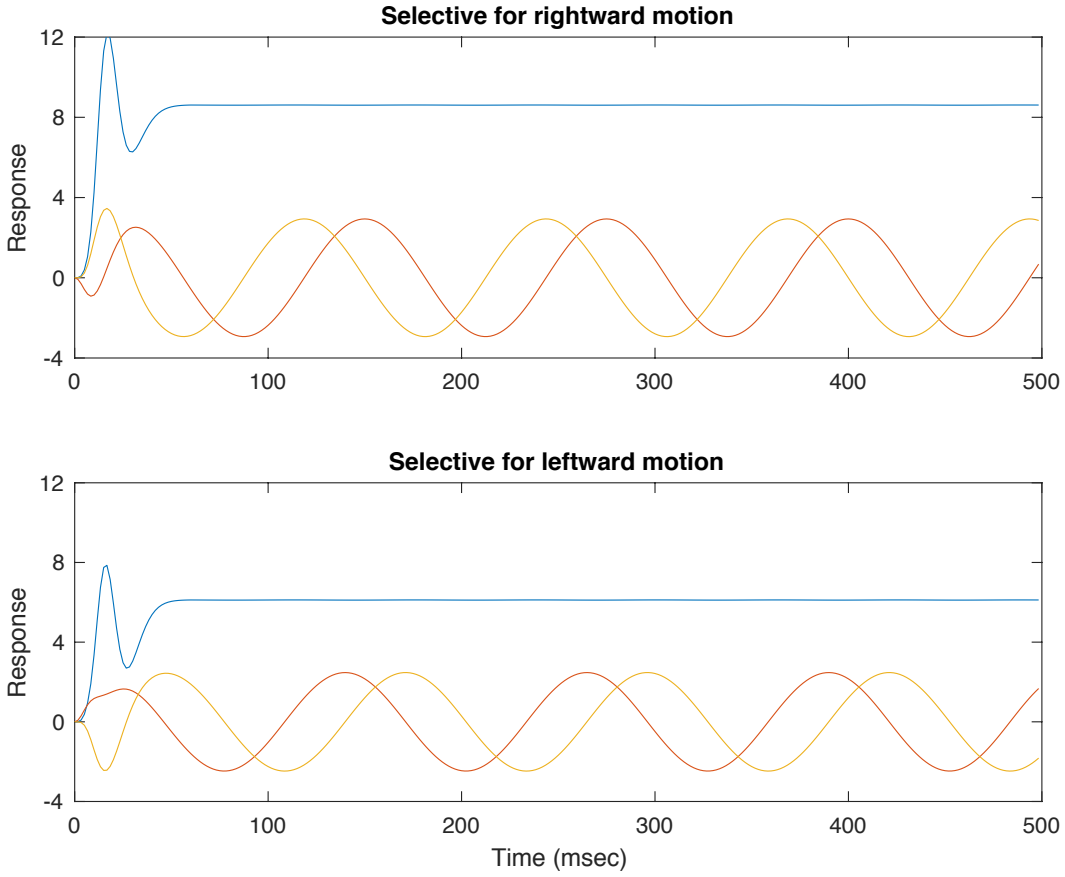
c) Compute motion energy:

```
leftEnergy = leftEven.^2 + leftOdd.^2;
rightEnergy = rightEven.^2 + rightOdd.^2;
etc.
```

and display x-t slices of the impulse responses of the energy responses. Your results should look something like this:



d) Create 4 drifting sinusoid stimuli (each a 3D array): 4 cyc/deg and 8 Hz, drifting in 4 directions (up, down, right, left). Run these stimuli through your code to compute the motion energy output arrays. Plot the responses (over time) corresponding to neurons at the center spatial location. Make 4 graphs corresponding to leftward, rightward, upward, and downward-selective. In each graph, include 3 curves corresponding to the even and odd linear filter responses and the energy response (e.g., leftEven, leftOdd, leftEnergy). Repeat for each of the 4 stimuli so that end up with a 4x4 set of graphs corresponding to the 4 stimulus motion directions and the 4 direction preferences. Your results should look something like this:



4) Implement normalization. Extend the code from #3 to normalize the energy responses:

```
sumEnergy = leftEnergy + rightEnergy + upEnergy + downEnergy;
leftEnergyNorm = leftEnergy ./ (sumEnergy + sigma^2);
```

where of course this sigma is different from the one that controls the width of the Gabor filters.

a) Simulate a contrast-response measurement. Compute the normalized energies for rightward drifting gratings of various contrasts (from 1% contrast to 100% contrast in log steps). Ignore the first few hundred msec of the responses for which you should observe some transient funkiness (as in the figure above). Compute the mean over time

for the remaining duration of the response time courses. Plot this mean response amplitude versus log contrast for each of the 4 simulated neurons (upward, downward, rightward, and leftward-selective) corresponding to the center spatial location. Select the value of sigma so that a stimulus of 10% contrast evokes about half the maximal response. See Fig. 3 of Carandini and Heeger (Nat Rev Neurosci, 2012) for examples of what your results should look like.

b) Simulate a cross-orientation experiment. Repeat part a with rightward drifting gratings of various contrasts from 1% to 50%, either presented on their own, or superimposed with an 50% contrast upward drifting grating. Make 4 graphs (upward, downward, rightward, leftward-selective) each with 2 curves (rightward grating alone, rightward grating superimposed with upward grating). See Fig. 3 of Carandini and Heeger (Nat Rev Neurosci, 2012) for examples of what your results should look like.