9.35 Sensation And Perception
Spring 2009

9.35 Spring 2009
Problem Set 2 – 100 Points

NB: This problem set assumes you have the Image Processing toolbox installed in Matlab. This is the case if you work on Athena or have downloaded the full version to your personal computer. If you can't get access to the Image Processing toolbox, or you are working in another environment (like Python) then other solutions are possible – please ask for help if needed.

Please turn in all code, figures and results in a single PDF or Word document and post to Stellar. You may also turn in a paper copy or email your PDF to the TA's if you cannot get Stellar submission to work.

Tip: try to write as much code as possible with vector and matrix commands, instead of for loops. This will greatly easy your life! Also beware the difference between matrix operations and element-wise operations, ie `A*B` vs `A.*B`. Ask for help if needed.

For clarity, Matlab syntax is provided in `monospace`, programmatic instructions are *italics*, and conceptual questions are in **bold**.

**1. Linear receptive fields (20 points)**
Let's assume that we can model the receptive field of a neuron as a linear filter. In other words, the response of the neuron is a weighted linear summation of light in its receptive field.

1.a) (10 points) Let's create a model of a receptive field as a 2D Gaussian (like an ON-center cell, but without surround inhibition). Use the syntax `[X Y] = meshgrid(-20:20)` to map a piece of a 'retina' in X and Y coordinates.

A Gaussian filter can be defined as:

$$F_{x,y} \propto \exp\left\{\frac{-(x-x_0)^2-(y-y_0)^2}{2\sigma^2}\right\}$$

Where ($x_0$, $y_0$) is the center of the receptive field, and σ is standard deviation in both *x* and *y* (approximately 1/3 the width of the RF).

*Use this expression, and the matrices X and Y, to create a 2D Gaussian with a standard deviation of 4. Normalize the filter so that it integrates to 1. Call this G1, and use* `imagesc` *and* `colormap('gray')` *to display this receptive field.* **If this is the structure of a linear receptive field, what does the intensity of any particular pixel mean? What is a reasonable assumption about the wiring of the retina that would make the linear weights of pixels in the receptive field decrease with distance from the center?**

1.b) (5 points) *Create another 2D Gaussian receptive field (G2) with a smaller standard deviation (2), normalize, and display it.* **If G1 and G2 were the receptive fields of two ganglion cells, which one would have greater sensitivity to light? Why?**

1.c) (5 points) *Subtract G1 from G2 to make another filter, DoG (Difference of Gaussians), and display this image. Now, plot the (1D) cross section through the middle of this receptive field.*

**2. Predicting ganglion cell responses to natural images (50 points)**
We will be modeling the output of cells as a continuous value, but keep in mind that this represents the firing rate.

2.a) (5 points) Given that we assume our receptive fields are linear, we can predict the output of each cell as the weighted sum of image intensities. That is:

$$O=\sum_{x,y} F_{x,y} I_{x,y}$$

Where *O* is the output, *x, y* are the coordinates of each pixel of the receptive field, *F* is the the receptive field filter, and $I_{x,y}$ is the intensity of the input to pixel *x, y*.

*What would be the output of a ganglion cell with the receptive field created in 1.a if a spot of light of intensity 1 fell on the center (0,0) pixel? What if a second spot of intensity 1 fell on (2,3)? What if the whole field were covered with intensity 1 spots?*

2.b) (5 points)*Use* `im = imread('peppers.png')` *to load a still life picture from the Image Processing toolbox* (if you do not have this toolbox, use an image of your choice). Note that some images load as 8-bit integers, and if this is the case, you should convert it using the syntax `im = double(im)/255;`. For simplicity, we will only use black and white images, so extract the green channel using `im = im(:, :, 2);`. *Display the image.*

*If a ganglion cell with a DoG receptive field (from 1.c) were centered on pixel 125,125 (such that the receptive field spanned pixels 105 to 145 in each dimension), what would that ganglion cell's output be?*

2.c) (15 points) We can model the response of a population of cells using a mathematical concept called convolution. **In plain English, how does convolution model a population of cells? Describe how you would implement this algorithm in Matlab.** Due to complications at the edge of images, it is actually somewhat tricky for beginning programmers to implement a convolution algorithm. You should use `filter2` instead of writing your own function (unless you are particularly ambitious!).

2.d) (10 points) *Model the response of a population of ganglion cells to the peppers image with your DoG filter. Display the output using* `imagesc` *and* `colormap('gray')`.

**What is a plausible (based on what you know about the cells in the retina) interpretation of negative values in this output? What features of the image provide greatest stimulation to cells with a DoG receptive field, and why? How would you change the properties of this receptive field to make the cell more sensitive to fine-grained features, such as the wrinkles in the cloth?**

2.e) (10 points) *Repeat this convolution with the two other (G1 and G2) filters from problem 1*. This would correspond to the output of two populations of retinal ganglion cells without inhibitory surrounds. *Display these two images in one figure using* `subplot`. **What is different between the output of the two filters? Why is it important that we normalized the integral to 1 (with respect to the DoG, and also by itself)? What would happen if we didn't?**

2.f) (5 points) *Find a picture of yourself and repeat the three convolutions*. You will probably need to resize your picture (or redefine the filters) to produce interesting results.

**3. Predicting V1 simple cell responses to natural images (30 points)**
We will now move further along the visual stream and model the response properties of idealized, linear, oriented receptive fields, like those found in V1. Filters of this type are commonly modeled with Gabors, which are the product of an oriented sine wave and a Gaussian envelope.

In problem 1 we created several Gaussian envelopes, now we need only to create the sinusoidal grating. We can define a grating of any given orientation as:

$$G_{x,y} = \sin\{2\pi * f * (x\cos\varphi + y\sin\varphi)\}$$

Where *f* is the spatial frequency in cycles per pixel,*φ* is the angle about the *x*-axis, and *x, y* are the indices of each pixel.

3.a) (5 points) *Use the grid from 1.a and the equation above to create a sinusoidal grating with 4 cycles total, rotated 0 degrees about x. Display this grating in grayscale.* Tip: write a function, or anonymous function, to return a grating of a given orientation.

3.b) (10 points) **If this grating were reproduced in print, at 300 pixels per inch, what would be the spatial frequency of the image (in cycles per degree of visual angle, cpd) if you stood 3 feet away? How far away would you have to stand to make this value around 10 cpd (roughly the peak of your contrast sensitivity function)? If you were standing 4 feet away, what would the wavelength have to be (in pixels), to make the grating 10 cpd?**

3.c) (10 points) *Multiply this grating by a Gaussian envelope (i.e., G1), renormalize, and display.* This Gabor function can serve as a simple edge detector, and is a good approximation of receptive fields of certain cells in V1. *Create another three V1 filters, oriented at 45, 90 and 135 degrees. Convolve both images (you and peppers) with all four receptive fields and display the results.*

3.d) (5 points) **What aspects of real RGC or V1 cells do our toy linear filter models *not* explain?**

## 4. Feedback

We are perpetually refining the Matlab component of this course. Please let us know how many hours you spend on this problem set, if anything was particularly difficult, or if you have any general comments for this assignment or others in the future.