COMP 546 Assignment 1

Prepared by Michael Langer

Posted: Wed. Jan. 23, 2018
Due: Wed. Feb. 7, 2018 at 23:59

General Instructions

For clarification questions, please use the mycourses discussion boards. You can send emails to the professor or TA, but if the question is of general interest then we may ask you to post it on mycourses so that everyone can benefit from the answer.

TA Office Hours and Location will be posted and updated when necessary on mycourses.Announcements.

Submit a single zip directory **A1.zip** to the myCourses Assignment 1 folder. The directory should contain all of your files, including the Matlab[1] files in the posted code. It should also include a PDF file with your answers.

The TAs will grade the PDF file. For any figure that you include in your answers in the PDF, the code you used to generate that figure should be part of your submission. The TAs will not have time to go through your code, but if an issue arises afterwards where you wish to discuss your grade, then you need to be able to demonstrate how you obtained the figure.

We suggest that you use WYSIWYG word processing software (MS Word or google docs) rather than latex. Latex is great for publishing but one can waste a lot of time with latex figures. You will be graded far more on the content than the presentation style.

Late assignment policy: Late assignments will be accepted up to only 3 days late and will be receive a late penalty. The default will be 20 percent of your grade. For example, if your grade is 70/100 and you submit two days late, then you will receive a grade of 56/100. Note: If your assignment is 1 minute late, then we reserve the right to consider it to be 1 day late, so we strongly advise you to submit well before the deadline.

[1] You are not required to use Matlab. You may use python/numpy/scipy instead. However, you will need to convert the starter code. (If you do, then please post it the translated starter code on mycourses discussion board so that other students can benefit).

Introduction

In this assignment, you will simulate some of the image filtering operations that are performed early in the visual system. This will give you 'hands on' experience with these operations, and also familiarize you with some of the challenges and issues that arise in displaying images for analysis.

The lectures discussed continuous light spectra and LMS cone responses, but we will keep it simple and just deal with RGB. We will also use only simple synthetic images, rather than real images captured with a camera. There is a good reason for this. When you capture an image with a digital camera, the RGB values typically have been processed already. For example, the RGB values in a JPEG image are *not* proportional to the filtered light intensities measured by the camera sensor, but rather they have been non-linearly transformed by the camera's image processing hardware. There is a lot to say about how digital images are captured, coded, and displayed, but these details would take us too far from the content of the course. To avoid confusion, we will just use simple images where we set our own RGB values.

A related point is that we can easily be misled when judging intensities at different image points if our judgements are based on our eyes. It is better to examine the intensity values as numbers.

Enough preamble: Let's get started!

Question 1: Local Contrast (40 points)

The Matlab program **Q1.m** gives you some starter code that performs operations on various images. For each image, it transforms each RGB pixel to a *local contrast* as follows:

- It defines the *intensity* at each pixel to be (R+G+B)/3
- It subtracts the *local mean intensity* from the pixel
- It *normalizes* the intensity by dividing by the local mean intensity.

The local mean intensity is computed using the Matlab `imgaussfilt.m` function which convolves[2] the intensity image with a Gaussian.

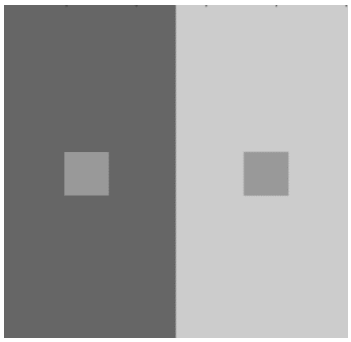
a) The `makecheckerboard.m` function makes a color image and multiplies the RGB values in the right half by a small constant 0.2. Scaling down the intensities like that makes it more difficult to *visually* distinguish neighboring squares in the right half. The effect is similar to a shadow cast on a surface.

   Your task is to vary the sigma that is used for computing the local mean intensity, and examine how changing the sigma affects the local contrast of the shadowed checkerboard.

[2] Cross-correlation and convolution are equivalent operations when the filter is symmetric f(x,y) = f(-x,-y) such as a Gaussian.

Are there values of sigma for which the local contrast counteracts the effect of the shadow, for example, so that the local contrast allows one to better compare the original values of the unshadowed (left) versus shadowed (right) squares? Briefly explain.

b) Consider a *simultaneous contrast* image such as:



The left grey square appears to be slightly brighter than the right one, although in fact the two grey squares have the same RGB values. Can *local contrast* explain the perceived brightnesses. That is, are the computed local contrasts consistent with this illusion? Justify your answer by computing the local contrast and comparing values across the image.

c) What about White's illusion? https://en.wikipedia.org/wiki/White's_illusion
Local contrast *cannot* explain White's illusion. Explain why not, by computing the local contrast on the given image `Whites_illusion.jpg` and examining the values.

d) Define a *single opponency* channel R-G that compares red and green components of an image by computing the difference of the R and G values at each pixel. One can normalize the values of a single opponent channel similar to what was done with local (intensity) contrast in Q1, for example, by dividing the opponency channel by a local Gaussian weighted average intensity in some neighborhood Normalizing the single opponent channel would give *color contrast*.

What might be the goal of normalizing the R-G opponent values? Would the simple method above achieve this goal? Give an example to illustrate your argument, using a random color checkerboard image for which the B value is zero everywhere.

Question 2: Gaussian and DOG filtering of spatial patterns (40 points)

a) Use the `mk2DGaussian.m` function to define a DOG using two Gaussians whose sigmas are very similar.[3] Use the Matlab `filter2` function to convolve your DOG with an image that is a 2D sinusoid. See `mk2Dsinusoid.m`. Try many different spatial frequencies, and plot the root mean square (RMS) of the DOG filter outputs *as a function of spatial frequency*. For simplicity, vary only one of `kx` or `ky` and set the other to 0.

Which spatial frequency gives the largest RMS filter outputs? How does the wavelength of this 'optimal' spatial frequency compare to the sigma(s) used in defining the DOG?

Note that we are *not* asking you to normalize the filter outputs as we did to compute local contrast.

b) Using the same DOG as in the previous question, consider the filter outputs for an image of a single white square on a uniform grey background. Holding the intensities of the square and background constant, vary the size (width) of the square and examine the DOG filter outputs. What happens when the square is much smaller than the size of the center region of the DOG? What if the square size is roughly equal to that of the DOG? What if the square size is much greater than the DOG?

c) A *double-opponent* cell is defined by using a DOG in the R channel and a DOG in the G channel, such that the sizes of the two DOGs are the same but *the signs are opposite*. For example, the DOG in the R channel could be on-center off-surround and the DOG in the G channel could be off-center on-surround (or vice-versa).

Show the filter outputs when convolving with a double opponent cell on images of:

- a red square on a white background.
- a red square on a green background
- a green square on white background
- a green square on a red background
- a white square on a black background

In each case, assume the square size is similar to the center region of the cells.

[3] Note that the values of the DOG will be near zero everywhere. You may wish to rescale the DOG weights, although this has no important effect on your answer.

Question 3: Chromatic aberration and depth (20 points)

The power of a lens depends on the wavelength of light. For the human eye, the difference between the power at the shortest versus longest wavelengths is nearly 2 diopters, with power being greater at shorter wavelengths. This variation in power leads to image blur known as *chromatic aberration*. See Fig 233 in http://www.telescope-optics.net/eye_chromatism.htm. We will consider only *longitudinal* chromatic aberration, which specifically concerns the depth at which a scene point is focused as a function of wavelength.

The L and M cones have peak sensitivity at nearby wavelengths so for simplicity we will neglect any chromatic aberration between L and M channels. The S cone has a peak sensitivity at quite a different wavelength from L and M which gives it a difference in power of about 1 D (diopter) from L or M. For simplicity, we identify the L and M cone images with the R and G channels and we identify the S cones with the B channel. We assume the lens is 1 D stronger in S than it is in L and M. For example, if the eye is focused in the L and M channels for some scene depth, then a nearer depth would be in focus for the S channel. We will investigate the chromatic aberration that is due to this assumed 1D difference in power between L/M and S.

a) Make five RGB images of two isolated white squares on a dark grey background. *To computer blur, assume that the left square is 2 diopters closer than the right square*. (Make the left square bigger than the right square in the image to indicate left is closer.)
Let the L and M channels images be focused:

- 1 D beyond the far square
- On the far square
- At the midpoint (in diopters) between the far and near square
- On the near square
- 1 D nearer than the near square

To calculate the amount of blur in pixels in each RGB channel, assume a pupil size of 5 mm and a resolution of 40 pixels per degree of visual angle and apply the formula in the exercises. (The 40 px/deg resolution is roughly what you'll find a few deg from the fovea.)

To blur the image in Matlab, you can use:

```
f =  fspecial('disk',blurWidth);
Iblurred = filter2(f,Isharp);
```

You will need to apply a different blur width for each of the squares and for each channel.

Show the computed RGB images (as RGB images) and label them. Briefly describe the width and colors of any blur fringes that occur for two of the five cases, namely when

- the L and M channels are focused on the far square
- the L and M channels are focused on the near square.

b) Consider what happens when the retinal images are filtered using a Y-B double opponent color cell (i.e. yellow on-center off surround, and blue off-center on-surround). By "yellow" here, I mean the average of L and M (R and G).

   For simplicity, only consider two of the five images:
   - the L and M channels are focused on the far square
   - the L and M channels are focused on the near square.

   Show the two filtered image outputs, and briefly relate them to your answers above.

In case you are wondering, the main point of this question is that there is useful information about depth in the color fringes, and that basic vision mechanisms can extract these fringes. This is interested for at least two reasons. First, although vision scientists don't talk about the color fringes much, they are always there. Indeed there is recent experimental evidence that the visual system can use these color fringes to help with depth judgments in specific experiments. It remains to be shown whether these chromatic fringes play a role in depth perception in complex natural scenes. Second, it is interesting that we are not aware of these color fringes. Perhaps this shouldn't bother you since there are many curious image properties that we are not aware such as the fact that retinal images are upside down and backwards. I know that, but it does still bother me. That's because it is so easy to fall for the *illusion* that what we experience when looking at the world corresponds exactly to the physical image measured by our eyes.