COMP 546/598 Assignment 1

Prepared by Prof. Michael Langer
and T.A. Ruslana Makovetsky

Posted: Friday, Sept 24, 2015
Due: Tuesday, Oct. 6, 2015 at 23:59.

Note that the due date is the same day as the midterm exam.

General Instructions

- Please use the mycourses discussion boards for clarification questions, rather than emailing the professor or TA. Only email the TA/Prof when your question might give away too much information or if you are sure that it is not of general interest.

- Ruslana will be available to meet by appointment. Please arrange by sending her email ruslana@cim.mcgill.ca.

- Submit a single zip directory **FirstnameLastnameA1.zip** containing all of your files. This should include all Matlab files in the posted code so that your programs can run in that directory. It should also include a PDF file that contains inserted images and discussions. For simplicity, we suggest you write the files as a Word document and save it as PDF. Submit this zip file to your myCourses Assignment 1 folder.

- Ruslana will grade the assignments. If you have any issues that you wish her to be aware of, then include them as a comment with your submission.

- **Late assignment policy:** Late assignments will be accepted up to only 3 days late and will be penalized by 20 percent per day. If you submit one minute late, the instructor reserves the right to treating this as equivalent to submitting 23 hours and 59 minutes late, etc.

- **Bonus points:** It is possible that there are problems (ambiguities or minor errors) in this assignment. If you help out by notifying us of these problems, you will be eligible for bonus points. Getting started early raises the likelihood of benefiting, of course.

Question 1   (50 points)

The Matlab program **color.m** reads in a color image and decomposes it into R,G, and B channels.   For each pixel, the program rotates the RGB vector into a new 3D orthonormal coordinate system which captures changes in three dimensions: (1) overall intensity or "luminance", (2) red versus green, and (3) blue versus yellow.   Note that the new "channels" (2) and (3) can have both positive and negative values.

Recall from the color lecture that "chromaticity" refers to the 2D space in which hue and saturation (but not luminance) vary. In Q1a, you will manipulate the RGB values in the new channels and examine the chromaticities in the image.

In Q1b, you will explore the spatial variations within each the new channels. In particular, you will compute a discrete approximation to the gradient

$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

within each of these new image channels. Here $I(x,y)$ denotes one of the three new channels.   A large gradient magnitude is often referred to as an "edge". Edges are important in vision because they often correspond to visually important changes in the scene e.g. a material or illumination change or both.

It is believed that one way in which the vision system distinguishes an illumination edge from a material edge is by examining if both luminance and chromaticity change across the edge, or if only the luminance changes. If there is a material change, for example due a printed pattern on your shirt, there may be large changes in both luminance and chromaticity, whereas if a uniformly colored surface has a fold on it which produces shading or if there is shadow cast on the surface then there will be a change in luminance but only a small change in chromaticity. (One well known exception to this rule is that, on sunny days, the sun is yellowish and the sky is blue, and so for sunlit scenes you may find that shading and shadows produce yellow-blue chromaticity edges that are aligned with the luminance edges, with the yellowish side of the edge corresponding to the higher luminance side.)

Choose a small set of color images from the web, and explore the above idea. For example, a closeup image of green grass should have roughly constant chromaticity but it may have large luminance variations which are due to shadows cast by the individual grass blades, whereas a scene consisting of red apples among green leaves typically will have large variations in both luminance and chromaticity.

Select and submit an image that contains examples of (1) an illumination edge which produces a luminance edge but no chromaticity edge, and (2) a material edge which produces both a luminance and a chromaticity change. This image should be renamed **Q1-example.jpg** or a suitable extension other than JPEG.   If you require two images – one for each case -- then add number 1 and 2 to the name.   Note: JPEG images are compressed and if the compression is large then they will contain edges that are due to block artifacts.   *Do not choose such an image.*

Specifically, add code to **color.m** which does the following.

a) Compute a "chromaticity-only" RGB image, which we define as follows.   Set the luminance component to a constant value for all pixels, namely 128, and scale the given red-green and blue-yellow components such that the new RGB triplet is a scaled version of the original RGB triplet. That is, the ratios between any two of the R, G, and B values at each pixel are the same as in the original image.

   Your code should use the provided function **remapImageUint8.m** and display the new image as a figure and save it as a file **chromaticity-only.jpg** which you should insert into your PDF document.

Briefly (50-100 words) discuss the chromaticities of the new image, namely the hue and saturation. Are they consistent with what you expected, given the original image?

b) Compute an *"edge map"* of each of the three new channel images (luminance, red-green, and blue-yellow). We define an "edge map" to be a grey scale image that contains the magnitude of the discrete approximation of the gradient vector. (In computer vision, this computation is part of many 'edge detection' methods e.g. search "*Sobel operator"*.) Note: you are <u>not</u> computing edge maps on the R, G, and B channels themselves.

   To compute the image derivatives, you must use the Matlab convolution operator **conv2** rather than using **for** loops, since for loops are too slow in Matlab.

   Display each of your edge maps as a figure with a title, and save the figures as JPEG images.

   Briefly (50-100 words) discuss your three edge maps and how they illustrate the above idea about material versus illumination changes based on your interpretation of the scene in the selected image. Your PDF should include the original image, the three images computed by the given code, and the three edge maps. Organize your answer so that it is easy to grade, e.g. make a 3x2 table of the three new channel images and the three edge maps, and label them appropriately. If you use two examples images, then use two tables. You may use the **subplot** command to make the table(s).

Submit your modified **color.m** so that Ruslana (TA) can test its correctness on a different image.

Question 2 (50 points)

The program **blurEccentric.m** synthesizes a texture and then blurs this texture according to eccentricity, namely pixels further from the image center are blurred more. Run the program and examine the image that is presented. View it from a distance such that the width of the image subtends about 5 degrees of visual angle. Fixate (look at) the center of the image i.e. the black square. Whether you can or cannot detect the blur gradient depends on how you set the **contrast**[1] and **sigmaAtEdge** parameters.

a) Briefly discuss (50-100 words) how your ability to detect the blur gradient varies with the **sigmaAtEdge** and *contrast* parameters. For at least two different contrasts between 0 and 1, what is the minimum **sigmaAtEdge** value for which you can perceive a blur gradient ? For at least two different **sigmaAtEdge** values, what is the minimum contrast for which you can perceive a blur gradient? Include examples of the four images in your PDF.

   We are not looking for exact answers here, but rather just some evidence that you have explored the parameter space. Note that it is important that you fixate the black square: it is much easier to detect the blur gradient if you look instead at the outer part of the image.

[1] There are several ways to define "contrast" in an image. One common definition is Michelson contrast which is defined as (Imax – Imin) / (Imax + Imin). So if the minimum intensity in an image is 0, then the contrast is 1.

b) The given program **layerCosine.m** modulates the image contrast according to a raised 2D cosine function. The image “layer” blending model used in that program is very common in vision science e.g. it is often used to investigate transparency perception. The blending model is also common in computer graphics to combine e.g. in Photoshop. Here we are considering only a simple version of the model

```
I(x,y) = alpha(x,y) * Iforeground +  (1-alpha(x,y)) .* Ibackground(x,y)
```

in which the foreground image has a constant intensity, the background image is a texture, and the opacity alpha is a cosine plus a constant.

Combine the methods of these two programs by writing and submitting a new program **blurCosine.m** that blurs the image, such that the amount of blur at each pixel is a raised 2D cosine taking values between 0 and some maximum blur. To aid in grading, use variable name **maxSigma** for the maximum blur value.

This spatially varying blur that is produced should be visible in the image. Include an example image in your PDF for maximum blur of 2 pixels.

c) Blurring and image layer blending both reduce contrast, but the effects are different. One difference is that blurring reduces the amplitude of high frequency image components more than low frequency components whereas blending treats all frequencies the same. (We will discuss blurring more in lecture 6.)

Consider two different levels of **maxSigma** in **blurCosine.m** such that the blur is visible**.** For both of these maximum blur levels, choose the blending constants **alphaMin** and **alphaMax** in the **layerCosine.m** code so that the contrast variations across the image appear roughly similar in the blurred versus layered images. (Note that ‘roughly similar’ is not well defined, so there is no unique ‘correct’ answer here.) This choice will require that you think carefully about what the layer model does, and you understand the difference between blurring and layering. Briefly justify (50-100 words) how you chose these constants.

Good luck, and have fun!