6.815/6.865 Digital & Computational Photography
http://stellar.mit.edu/S/course/6/sp11/6.815/

Problem Set 0: MATLAB Warmup
Due **Tuesday, February 8 5:00pm**

The goal of this assignment is to get you familiar with MATLAB. You'll be looking at some simple image manipulations which will highlight the basic functionality of the programming language.

1 Getting started

First, make sure that you can run MATLAB. It's available on Athena. If you want to run it at home, you can download it from MIT: `http://web.mit.edu/matlab/www/`.

If you're already familiar with MATLAB, that's great. If not, don't worry, because the assignments won't rely too much on extensive knowledge of the language. It might be beneficial to browse through some basic tutorials, such as the one provided by The MathWorks: `http://www.mathworks.com/academia/student_center/tutorials/launchpad.html`. The `help` command might also be handy. For example, try `help('imfilter')` to learn about the `imfilter` command.

Now go ahead and grab the starter code from the homework page. The zip file contains an image (`image.jpg`), several functions (`ps0grayscale.m`, `ps0threshold.m`), and a script that runs everything (`ps0main.m`). Open MATLAB, `cd` to the directory with the files, and type `ps0main` at the prompt to run everything.

You should see a bunch of images. Take a look at the `m`-files to see what's going on. The comments walk you through each step in detail.

Okay, now that we're done with the introductory stuff, here's what you should do:

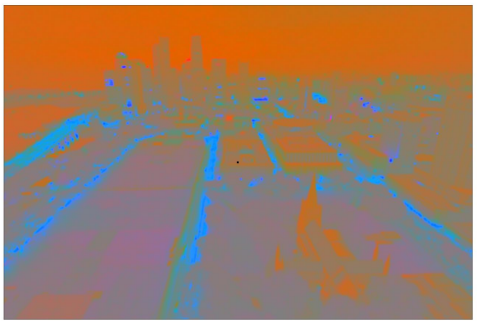
Color Threshold Images. Right now, `ps0threshold` will produce an threshold image with only black and white, as shown in the third image on this page. Write a new threshold function that allows users to specify RGB colors in place of black and white. You should be able to call it as follows: `mythreshold(Igray, 80, [10 0 178], [255 252 0])`, which should produce the fourth image on this page. The arguments are the grayscale image `Igray`, the threshold value, and the two RGB colors for pixels below vs. above the threshold.

2 Spanish Castle Illusion

Your goal is to automatically create a pair of images that induce the Spanish castle illusion as described at

`http://www.johnsadowski.com/big_spanish_castle.php`

In this illusion, the viewer is first shown an image with uniform luminance and where color are inverted (see below). The viewer fixates on a black dot in the center for a reasonable amount of time (10-30 seconds). The display is then switched to a grayscale version of the scene, but because of the prior exposure, it appears colorful. This is because the visual system has adapted to the inverted-color image and is biased towards the opposite (correct) colors. The illusion disappears progressively, or as soon as a saccade takes the gaze away from the fixation point. It works best for colorful landscapes with vegetation, blue skies and earth tones.



Given an input RGB image, you will create both the inverted color and the greyscale versions. Write a function
`function [G, C] = ps0spanish(I)`
that takes as input an RGB image `I` and returns a greyscale version `G` and an inverted-color version `C`, following the steps below

Convert to double Images are read by default in 8 bit format. Turn your input image `I` into floating point values with
`I=double(I)/255.0;`
where we divide by 255 because 8 bit images go between 0 and 255 while real-valued images should be between 0 and 1.

Create a grayscale image Use the function from the previous section.

Fixation point To create a fixation point in your grayscale image, turn pixels in the center of the image to black. Extract the size of your image with `S=size(C);` This gives you a vector with 3 values: width, height, and 3 (because there are 3 color channels).

With a single line of matlab code and no `for` loop, turn the 5×5 pixels in the center to black. Hint: = the colon operator ":" is your friend.

Invert the color image Invert the colors in your input image, that is, replace every pixel value x by $1 - x + \epsilon$. You do not need any loop, you don't even need the colon operator.

ϵ should be set to a small value such as 0.001 and is here to avoid divisions by zero later.

Boost saturation and contrast The illusion is more pronounced if the colors are enhanced. Contrast and saturation can be enhanced by applying an exponent (a.k.a. gamma curve). That is, each channel value x of each pixel is replaced by x^γ where $gamma > 1$ to increase contrast and $\gamma < 1$ to reduce it. We advise an exponent of 1.5 here.

Use the per-component exponentiation ".^ " for a fully Matlab-fashion solution. Note that the dot is important: it makes the distinction between taking an exponent using full matrix multiplication (without the dot) vs. taking the exponent of each individual coefficient (with the dot).

Create an isoluminant version Now we have inverted colors but we still have luminance variations. Remove them by normalizing each pixel by its luminance. Use your favorite grayscale conversion function to compute the luminance of the inverted image.

Advice: for normalization, treat each channel separately and use the colon operator ":" within each channel to avoid loops. Unfortunately, there is no perfect way (that we know of) to treat all three channels in a single expression. This is because the luminance array only has one value per pixel, while we need to normalize 3 channels per pixels. You can write one line of Matlab per channel or triplicate the greyscale image.

Blur Another bell or whistle is to blur the inverted-color image to keep the viewer on the fixation point and avoid distraction due to sharp edges. We'll keep it simple and use a box filter where each pixel gets assigned the average of its 5×5 neighbors. To create this filter, we can use the function `ones(n)` that creates a matrix of size $n \times n$ full of coefficients equal to 1. Don't forget to normalize this matrix so that the coefficients sum to 1.

Then, given this filter kernel, use `imfilter` to apply a convolution.

Yet another fixation point Use the same strategy as for the grayscale image to add a 5×5 black fixation point. Don't forget to adapt it to handle the three channels (whereas the previous grayscale version only had one).

Submission.

Create a zip file named `login.zip` (using your Athena login) that includes `mythreshold.m` and `ps0spanish.m`, and submit on the Stellar website by

Tuesday, February 8 at **5:00pm**