Homework Assignment #1

Due Date: Friday, 13 September, 2013 at 5:00pm (MDT)

General guidelines for Homeworks: Homeworks will consist of two parts: (1) written exercises (many will come from the textbook) and (2) programming exercises. For the programming exercises I will generally ask you to prepare a brief write-up (in PDF format) of what you did (see "**What to Submit"** below) and submit that along with any code that you have written. Your solutions to the written exercises can be submitted online together with the programming exercise write-up and code, this would be my preference (everything is in one place). However, if you prefer, you can turn your solutions to the written exercises in to me in class. If you choose to submit them online, they can be either typed or you can scan your handwritten (and clearly legible =]) solutions.

What to Submit

Please place everything you're submitting online into a single .zip or .tgz file. For this homework you should turn in:

1. Written Exercises (either submit with the rest or give separately to me in class on Friday)
2. Programming Exercise Write-up (see below)
3. All code that you've written (both MATLAB and Python). If you wish, you can leave out the code that you've copied/pasted, but make sure that the code you write on your own (to create the histograms is included.
4. Please include the original image file that you used.

WRITTEN EXERCISES

1) A standard music CD holds 74 minutes of stereo (2-channel) audio at 44,100 samples/second and 16 bits/sample/channel. How many megabytes of raw data can such a CD hold?

Just to think about: the recordable data for CD-ROMs is usually slightly less than this. Can you think of a reason why it would be smaller? Hint: CDs sometimes have read/write errors for individual bits. How would such an error affect music? How would it affect your financial records?

The following exercises are from the **Gonzales and Woods (G&W)** textbook:
2) G&W 2.5 - Assume the maximum resolution occurs when each of the black/white lines falls one-to-one on a row of the sensor's pixel array.

3) G&W 2.10

4) G&W 2.19 - Consider this: if you add each of a sequence of numbers to the corresponding elements of a sequence of identical length, is the median of the sum the same as the sum of the respective medians?

PROGRAMMING EXERCISES

For this assignment you will be completing the same brief series of exercises in two different programming environments (MATLAB and Python/numPy). I know that this will feel slightly repetitive, but the purpose of doing this is to give you practice with each of these widely-used platforms and allow you to choose which platform you'll use on future programming assignments.

These three platforms are:

- MATLAB (Mathworks, Inc) - available for Linux, Mac, Windows
- Python / NumPy - available for Linux, Mac, Windows

It is recommended that you complete these exercises on the linux workstations in the department's computer labs. They can be completed on other machines/platform, but MATLAB is not freely available. The department has it installed in these computer labs (at least on the linux machines).

For these exercises you will need to select an image (something approximately 500 pixels in width and/or height would be ideal). In the example code, I've used an image called "`original.png`".

Programming Exercise Write-up

Please prepare a written write-up (preferably submitted as a PDF) which includes the following:

- The `original` image that you used
- The `MIX` or `MIX2` image that you generated
- The `QUAD` image that you generated
- The histograms for the original image (generated in either MATLAB or Python)
- 1-2 paragraphs about whether you preferred MATLAB or Python/NumPy and why.
- 1 paragraph describing how long each part (Written Exercises, Matlab, Python) took along with any problems you encountered and/or suggestions for improving this assignment.

Programming Environment Setup

1. **MATLAB**

   Open a terminal window. Type "`matlab`". This will launch the matlab IDE. There is a command window where you enter commands (look for the ">>" prompt) and several other windows or panes in the IDE (for inspecting variables, seeing your previous commands, etc.). There's also a built-in editor where you can create code file (*.m) if you like.

2. **Python/NumPy**

   On the CS lab machines, everything that you already need is installed. In the event that you want to use your own machine, you'll want to install **ipython**, **numpy**, **matplotlib** and perhaps **scipy** (though this can be a little tricky to build if it's not available directly as a package, email me or drop by if you need help with this).

   To launch the interactive pylab, type "`ipython -pylab`". You may find that the command "cpaste" is very useful, it allows you to copy blocks of code disregarding any unnecessary leading indentation. Python is very particular about indentation. To use cpaste, just type "`cpaste`", paste in the code, then enter two hypens "`--`".

Activity	MATLAB	Python / numPy
Setup	**See Above.** Also, note that the command "help" (e.g. "help zeros") may be useful.	**See Above.** The command "help(pylab)" may prove useful. ipython also allows you to get info on functions with the ? operator as in "pylab.zeros?"

Startup	`$ matlab`
	`>> # MATLAB prompt`
		`$ ipython -pylab`
		`In [1]: # ipython prompt`
		
		`import numpy as np`

Read and display an image.
 	
Also see the raw values for a small portion of the image.
	`IMG = imread'original.png');`
	
	`imshow(IMG);`
	`size(IMG)`
	
	`IMG(1:5,1:5,:) % Note that MATLAB reads images using`
	`% 8-bit unsigned ints (0 to 255).`
	
	`% In MATLAB, terminating a statement with a semicolon`
	`% is optional; it suppresses MATLAB displaying the`
	`% results.`
		`IMG = imread( 'original.png' )`
		
		`imshow(IMG)`
		`IMG.shape`
		
		`IMG[0:5,0:5,:] # Note that by default, these are`
		`# floating-point values (0.0 to 1.0)`

Compute the inverse image	`INV = 255-IMG;`
	`imshow(INV);`
		`INV = 1.0-IMG`
		`imshow(INV)`

Create a mix of the two images	`MIX = zeros(size(IMG),'uint8');`
	`for r = 1:size(MIX,1)`
	`  for c = 1:size(MIX,2)`
	`    FRAC = 1.0*c/size(IMG,2);  MIX(r,c,:) = ...`
	`          (1-FRAC)*IMG(r,c,:)+(FRAC)*INV(r,c,:);`
	`  end`
	`end`
	`% In MATLAB "..." is used to continue a statement`
	`% onto the line below`
	
	`subplot(1,3,1)`
	`imshow(IMG)`
	`subplot(1,3,2)`
	`imshow(MIX)`
	`subplot(1,3,3)`
	`imshow(INV)`
		`MIX = np.zeros(IMG.shape)`
		`for r in range(IMG.shape[0]): # for each row`
		`    for c in range(IMG.shape[1]): # for each column`
		`        FRAC = 1.0 * c / IMG.shape[1]`
		`        MIX[r,c,:] = (1-FRAC)*IMG[r,c,:] + \`
		`                     FRAC*INV[r,c,:]`
		
		`% In Python, "\" is used to continue a statement onto`
		`% the line below`
		
		`subplot(1,3,1)`
		`imshow(IMG)`
		`subplot(1,3,2)`
		`imshow(MIX)`
		`subplot(1,3,3)`
		`imshow(INV)`

	`% Now for a glimpse of why MATLAB is powerful, we`
	`% can do this same thing without for loops.  The ".*"`
	`% operator performs an element-wise multiplication.`
	`GRAD = repmat( linspace(0,1,size(MIX,2)), ...`
	`               [size(MIX,1) 1 3]);`
	`MIX2 = uint8( (1-GRAD).*double(IMG) + ...`
	`              GRAD.*double(INV) );`
		`# numpy can also do operations without the for loops.`
		`# As a challenge, lookup and try to figure out how to`
		`# do this.`

Convert the original to gray	`GRAY = 0.299*IMG(:,:,1) + 0.587*IMG(:,:,2) + ...`
	`       0.114*IMG(:,:,3);`
	`imshow(GRAY)`
		`GRAY = 0.299*IMG[:,:,0] + 0.587*IMG[:,:,1] + \`
		`       0.114*IMG[:,:,2]`
		`imshow(GRAY)`
		`set_cmap('gray')`

Resize the image (create a thumbnail)	`THUMB = imresize(IMG,[32 32]);`
	`imshow(THUMB)`
		`# Note that the scipy package provides an easier`
		`# and higher-quality way to resize images, but`
		`# installing scipy generally requires compiling it`
		`from PIL import Image as PILI`
		`ORIG_PIL = PILI.open('original.png')`
		`THUMB = np.asarray( ORIG_PIL.resize((32,32)) )`
		
		`# what's the difference between these two?`
		`imshow(THUMB)`
		`imshow(THUMB,interpolation='nearest')`

Separate out the channels	`THUMB = imresize(IMG,[64 64]);`
	`QUAD = zeros(128,128,3,'uint8');`
	`QUAD(1:64,1:64,:) = THUMB;`
	`QUAD(1:64,65:128,1) = THUMB(:,:,1);`
	`QUAD(65:128,1:64,2) = THUMB(:,:,2);`
	`QUAD(65:128,65:128,3) = THUMB(:,:,3);`
	
	`imshow(QUAD);`
		`THUMB = np.asarray( ORIG_PIL.resize((64,64)) )`
		`QUAD  = np.zeros([128,128,3])`
		`QUAD[0:64,0:64,:]      = THUMB`
		`QUAD[0:64,64:128,0]    = THUMB[:,:,0]`
		`QUAD[64:128,0:64,1]    = THUMB[:,:,1]`
		`QUAD[64:128,64:128,2] = THUMB[:,:,2]`
		
		`imshow(QUAD,interpolation='nearest')`

Save a file	`imwrite(QUAD,’quad.png’)`	`imsave('quad.png', QUAD)`

Create histograms of your original photo	`% I'd now like for you to create 3 histograms for`
	`% your original image, one for each of the red, green`
	`% and blue channels.  For displaying histograms, the`
	`% bar() function may come in handy.  Also, note that`
	`% you can save your figure (look in the File menu) to`
	`% include in your report.`
		`# Same thing in python... there’s also a bar()`
		`# function for drawing histograms.  Note that while`
		`# I’d like you to generate histograms in both MATLAB`
		`# and Python, you only need to save and one set to`
		`# turn in with your write-up`