Yale CS 479/579 Problem Set 1 2014

Due Monday January 27, 2014, 1:00 pm
Submit by uploading to classesv2.

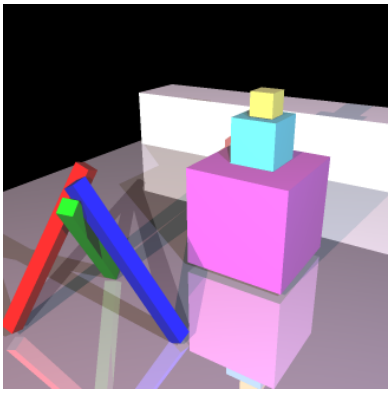
Below you will find the problems for Problem Set 1. Please note the following:

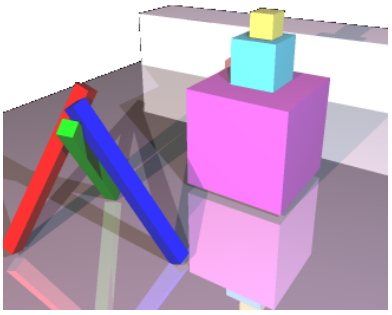
1. All problems should be uploaded to ClassesV2 together in a single zip folder.
2. Each problem should be in its own folder of the format "problem*n*". For example the solution to the first problem should be placed in a folder called "problem1a"
3. All assignments will be graded in the Zoo, which is a Linux environment. This means that all file and folder names are case-sensitive! It also means that you should test your solution on the Zoo computers.
4. You can start Matlab on the Zoo by opening a terminal and typing in *matlab*.
5. An automatic grader may be used, which may not be forgiving of formatting errors.
6. Check ClassesV2 from time to time, as we will post updates and correct any errors in the assignment there.
7. All Matlab functions should have its own m file with the same name as the function (as per Matlab's convention)

Problem 1a (C/C++, 479 and 579)

In this problem you will make a program that reads in an input file and returns the average Red and Blue values.

As a starting point, we have included a file *ImageFilter.cpp* that reads in an input tiff image and outputs a new image where dark colors have been replaced by white. For example, the program would transform the image on the left to the image on the right.





You are to create a program,"AverageColor", that takes an image as input and prints its average blue and green values. The output should be in the format (for a hypothetical example):

Average Blue: 23.0
Average Green: 43.2

Note the spacing and that there is accuracy to a single decimal point.
Place all your code in a file called AverageColor.cpp. Include a makefile with you code, which generates a program called AverageColor such that the following commands will produce the desired output.

Yale CS 479/579 Problem Set 1 2014

make
AverageColor *input.tiff*

Problem 1b (C/C++, 479 and 579)

In this problem you will make a program that reads in an input image and outputs a filtered image with a specific set of properties.

Before Goldilocks learned her lesson about moderation, she wanted everything to look like her name, golden. She has hired you to create a program that emphasis the gold in an image, and de-emphasize her least favorite color: blue.

Your program "Goldification" should:

1. Double the luminance of any pixel that is gold or yellow.
2. Half the luminance of any pixel that is blue

Note that you will have to identify what color pixels are (there is no explicit "yellow" channel), and you will have to scale the result to fit in the range 0-255.

Your should place all your code in a file called "*Goldification.cpp*" and include a make file such that the following commands will produce the desired result:

make
Goldification input.tiff output.tiff

Problem 1c (C/C++,579 Only)

In this problem you will produce a High-Dynamic-Range (HDR) image that fits a given description and then map that image onto two Low-Dynamic-Range (LDR) images for display.

The program "HDRtest" will take one input: *size* and produce two LDR images with dimensions *size* by *size.* Both of these outputs will be tone mappings of the same HDR image, but will use different methods to reduce the range.

The HDR image you produce should have values between -*1000* and *1000.* The center of the image should contain a square whose sides are $\frac{1}{8}$the length the size of the image. The rest of the image should be a sign-wave with a spatial wavelength equal to the width of the image and whose values are 0 on the entire first and last row.

The first LDR file, linear-*size*.tiff should be created by lineally scaling the values of the image. The second LDR file logarithmic-*size*.tiff should be created by scaling the image logarithmically.

All your code should appear in a file called *HDRtest.cpp* and take one parameter, the size of the image. It should output files linear-*size*.tiff and log-*size*.tiff. You should also include a make. You should ensure that the following commands produce "linear-512.tiff" and "log-512.tiff":

make
HDRtest 512

Problem 2a (Matlab, 479 and 579)

In this problem you will combine color channels from historically significant images to create color images – see

http://www.loc.gov/pictures/collection/prok/

Take a look at the series of images to the right. Since the image predates color film, the photographer needed to take three separate images with a color filter in front of his camera. Your job is to recombine the images into a color photograph. You can download the file from the Library of Congress at http://www.loc.gov/pictures/resource/prok.01468/

To start, you should find the upper left and bottom right coordinates of each of the three pictures. You can use whatever method you feel is best to find those coordinates, computerized or otherwise. Document these coordinates and how you found them in a *readme.txt* file.

Next, create a Matlab function:

make_color(image,red_rec,green_rec,blue_rec)

This function takes in the image to the right and three arrays that specify the location of each color channel. (The arrays should be of the format *[row col width height].)* The function should combine the three channels to produce a single color image. Save your best result as *out.tiff*

Hint: You may need to use the matlab function *imresize*

Problem 2b (Matlab, 579 Only)

In this problem you actually find the coordinates of the rectangle.

Propose a method for automatically finding the rectangles defining the individual pictures. Implement this method in a Matlab function *[red_rec,green_rec,blue_rec] = find_sub_images(image)* which takes an image like the one on the right and returns the location and size of each of the three sub-images, in the format described in 2a.

Compare the result your method has to the "ground truth" locations you used before. Describe your method and explain why your method did, or did not work in a *readme.txt.* Make sure you include the resulting rectangle coordinates in the readme.

Problem 3 (Matlab,479 and 579)

In this problem you will examine the 2D Fourier transom and visualize it.

Write a function show_fourier(image) that inputs an image and visualizes the amplitude of the Fourier transform of the image. In particular the function should use matlab's graphics capability to show one or more figures. Since this is an open-ended problem, please place a comment at the top of your code explaining what your function shows, and how it works.

Transform images that you can find of buildings, clouds (only), text and a person's face, and explain the different structures that appear in the visualization of the transform in a *readme.txt* file. Include screen-shots or images of your visualization for each of these examples in folders called: *buildings, clouds, test,* and *face*.

Hint: You should look up the following matlab functions:

fft2
fftshift
abs
imshow

Problem 4a (Matlab, 479 and 579)

In this problem you will implement the steps needed to create a hybrid image and also attempt to create one.

A hybrid image is an image that looks different depending on the distance that it is viewed. That is, when viewed up close it appears to be one image. However, when viewed from afar it appears to be another image. In 2006 Oliva et. al. produced a SIGGRAPH paper that demonstrated how to autocratically create a Hybrid image.

First, read the paper "Hybrid Images" by Aude Oliva, Antonio Torralba, and Philippe. G. Schyns. You can find the paper online at: http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf.

You should create two matlab functions.

The first function *[low,high] = hybrid_split(image)* should take an image and blur the images to separate spatial frequency. The function should return two values: *low* should be an image containing only low spatial frequencies, while *high* should be an image containing only high spatial frequencies.

The second function *create_hybrid(low_image,high_image)* should attempt to take two input images and return a hybrid image of both of them. That is, you should extract the low spatial frequencies of the first image and extract the high spatial frequencies of the second image and then merge them. You may call your *hybrid_split* and can expect it to work as you defended it. Note that we do not expect a perfect result but we do expect a *readme.txt* explaining what you tried and if it worked.

Problem 4b (Matlab,479 and 579)

In this problem you will provide examples of the program you created in problem 4a.

Select three pairs of images from the Internet or your own photos that you think would be good hybrid pairs. For each pair, create a folder called "*par_n"* where *n* is a number between 1 and 3. Place the files described below in the folder for each pair. Note that the pairs must be disjoint. Please convert all non-tiff files to tiff.

Use your *hybrid_split* function to produce the low and high spatial frequencies of your example images, and then use *create_hybrid* to create two hybrid images. Name the initial images *image1.tiff* and *image2.tiff*. The output of *hybrid_split* on *imagen.tiff* should be stored in files *imagen_low.tiff* and *imagen_high.tiff*. The output of *create_hybrid(image1,image2)* should be stored in *hybrid_12.tiff*. The output of *create_hybrid(image2,image1)* should be stored in *hybrid_21.tiff*.

In all, each folder should have the following 8 files:

image1.tiff	*image2.tiff*	*image1_low.tiff*	*image1_high.tiff*
image2_low.tiff	*image2_high.tiff*	*hybrid_12.tiff*	*hybrid_21.tiff*