Yale CS 479/579   Problem Set 3 2014

Due Friday, February 21, 2014, 1:00 pm
Submit by uploading to classesv2.

Below you will find the problems for Problem Set 3. 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)
8. You should not assume that images will be squares or that different images will be the same size, unless otherwise stated.

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

Write your own code to perform histogram equalization on a tiff image. You should create the following functions:

a)A function void grayscale(unsigned char* im_in, unsigned char* im_gray), where im_in is the original image with r, g and b values and im_gray is single channel images computed from .3r+.5g+.2b

b) A function void hist(unsigned char* im_gray, int* hist) that takes the grayscale image and produces a length 256 array with the number of pixels for each grayscale value.

c) A function void cdf(int* hist,double* cdf) that takes the histogram and produces a cumulative distribution function in the range 0 to 1,

d) A function void equalized(unsigned char* im_gray ,double* cdf, unsigned char* equalized_im) that produces a histogram equalized grayscale value image.

e) A function void equalized(unsigned char* im_in, unsigned char* equalized_im, unsigned char* final_out) that produces a color image that is equalized.

Create a program *equalize* that takes an input filename and an output file name, and creates a color equalized image. Create a makefile such that the following will work on the zoo:

```
make
equalize in.tiff out.tiff
```

Problem 2 (Matlab, 479 and 579)

Write your own code to do bilateral filtering on an image:
[base, detail] = mybilateral(input_image, spatial_radius, intensity_radius)

Here, input_image, base, and detail will be an $n \times m \times 3$ array double array. One way to get such an array from a file do

```
input_image = double(imread("filename.tiff"));
```

Problem 3 (C/C++ -- different versions for 479 and 579)

Normals can be encoded in a rgb image as follows: The coordinate system is defend with the center of the image being the origin, the x-axis extending to the right, y-axis extending up and the z-axis extending out of the page. That is, decreasing row value by 1 increases the y coordinate by 1 and increasing the column by 1 increases the x coordinate by 1.

The three color chancels of the image encode the three dimensional normal vector at each location. The r channel should encode the magnitude of the x coordinate of the normal and the g value should encode the magnitude of the y coordinate. Note that the 0-1 range of the magnitude should be scaled to values between 0 and 255. The bits in the b value determine if the two vectors are positive or negative. If the most significant bit of the b value is 1, then the x coordinate is negative, otherwise it is positive. If the second most significant bit of the b vector is 1 then the y coordinate is positive, otherwise it is negative.

(479) Write code of the form unsigned char* sphere_normals(int image_size) that creates a square image of size image_size that contain the normals of a sphere of radius image_size/2 centered on the image,
You should create a program sphere_normals_479 that takes an image size as an input and creates a tiff file "normals-*size*.tiff" as output. You should also make a makefile such that the following should work on the zoo:

```
make
sphere_normals_479 512
```

(579) Write code of the form unsigned char* sphere_normals(int image_size, int n) that creates a square image of size image_size that contain the normals of n spheres of radius image_size/(2n) scattered in the image.
You should create a program sphere_normals_579 that takes an image size and a number n as an input and creates a tiff file "normals- *size-n*.tiff" as output. You should also make a makefile such that the following should work on the zoo:

```
make
sphere_normals_579 5 512
```

Problem 4 (Matlab 479/579)

Write Matlab code for an interactive viewer that takes images of the form described in Problem 3, and displays a lit image with the light direction specified by the position of the cursor.

In particular, create matlab function view_normals_image(image) that takes an image array (as returned by imread) and shows a GUI that displays a lit surface. The location of the light should be determined by allowing the user to click on the image. You can assume that there is a perfect Lambertian surface and can ignore any shadows. It should take no more than 5 seconds between a click and an image update on the zoo.

Problem 5 (Matlab, 479/579)

Write Matlab code that takes three images and three light directions and computes a normal image in the format described in Problem 3.

In particular create a function out=compute_normal_map(image_structure) where image_structure is a matlab structure array describing the images and light direction. Each element in image_structure will have two fields: vector and image. Vector will contain the direction from the light to the object and will be in the format [x, y, z]. image will be in the same format as outputted by imread. We included a function load_lit_images(filename) that will load the lighting date in the structure format. The output of compute_normal_map should be a uint8 array with an image in the form described in problem 3.

The following call should work when you are done with this problem and problem 4:

```
view_normals_image(compute_normal_map(load_lit_images('data_479/data_479.mat')))
```

You can assume that all images will be the same size.

(479) We will only pass you arrays with exactly 3 images.

(579) You can assume that the arrays has at least 3 images, but may contain an arbitrary number.

(Extra Credit) Using a camera on a tripod and a desklamp, take three pictures lit from three different directions that you are able to estimate from cast shadows. Use your code to compute normal from the images you have taken.

If you want to make your example readable by load_lit_images place all your images in one directory, and then save a mat file containing a structure called light_data. Each image should have an entry in light_data with two fields: vector, as described above, and file_name which just contains the name of the file with the image.

If you finish the extra credit before the assignment is due, contact the TF for the possibility of distributing your new dataset to the rest of the class as an additional test case.