COSC 190 Midterm

Dr. Fred Park

Whittier College

Wednesday March 22nd, 2017

Name:_____

Student ID#:_____

Note: For full credit you must show all work! Please box your final answers. All work must be done on your own. No group work. This is strictly an individual work exam!! Only your class notes from COSC 190 and Matlab can be used. No other resources allowed.

1. Computer Science Coding Basics. Do as indicated.
   (a) Write a while loop that sums the squares of the numbers from 1 to N where N is input by a user. Make sure to include a prompt.
   (b) Using a for loop, prompt a user to input an integer N and output that many terms from the Fibonacci sequence defined by:

   $$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ \ldots,\ f_n,\ f_{n+1},\ \ldots$$

   so that the i-th Fibonacci term is defined recursively as:

   $$f_i = f_{i-2} + f_{i-1}$$

   for $i \geq 3$ with $f_1 = 1 = f_2$.

2. Image Downsampling and Beyond:
   (a) Load in the 'cameraman.tif' image and call it f. It is clearly a grayscale image of size 256×256. Downsampling is the methodology of reducing the resolution and hence, the size of an image. Create a set of 3 images obtained from the original cameraman image called f_1, f_2, and f_3 by downsampling where they are of sizes 128×128, 64×64, and 32×32 respectively. Note, the downsampling operator is called decimation since you are essentially tossing out image information when creating a smaller image.
   (b) By what factor are you downsampling when you go from a 256×256 image to a 128×128 one? i.e. what factor of less pixels do you have after the downsampling?
   (c) Create a 2×2 subplot and view all four images clearly labeling each plot and the factor of downsampling.
   (d) (Extra Credit) Image upsampling is the process of adding resolution to an image and hence information and size. This is also known as digital zooming. From your downsampled 128×128 image, you can embed it into a 256×256 image in the following manner, for every 1 pixel of info you have, there will be three pixels that there is no information. Set those pixels to zero. Now, fill in those missing pixel values with that one pixel value. This is called nearest neighbor interpolation. Plot both images on a single figure using subplot and clearly label each image as to what is going on. What factor are you zooming when going from a 128×128 image to a 256×256 one?

3. Histogram Equalization.

   (a) From class we saw that an ideal image contains pixel values in equal amounts across the entire image value range. This yields a uniform distribution of pixel intensities in an image. If you apply the histogram equalization method from class to an image that has pixel intensity values uniformly distributed, what is the distribution of the resulting image? Show this via specific example that you generate. Show the resulting distributions and image. Make sure to also plot the original image and corresponding distribution.

   (b) What is the relationship between the histograms counts of an image and its probability density function (PDF) of image intensities? What is the relationship between the discrete PDF and its cumulative distribution function (CDF)? Create a 5×5 synthetic image example and show the relationship.

4. RGB color space.

   (a) Load the 'fish.png' image from my webpage and call it f. It is an RGB color image with 3 channels. It is a stunning image with vibrant colors. In particular, the abstract fish is a nice blue color. By reshuffling the channels, can you make the fish into a beautiful pastel purple color and the surrounding non-fish circles into varying shades of green? Show this explicitly.

   (b) Can you create an image where the beautiful blue fish is kept in its full glory, but the other circles in the background are set to black? Show your result in a plot and include the commands that allowed this. For this problem, you are required to use the 'repmat' command at least once.

   (c) Is a shark a fish?

5. Edge detection and extraction.

   (a) From class, we discussed edge detection. Write code that computes an edge detector using the jump information from class. Consider the following notation for an image f that is grayscale:

$$f_x(i,j) \approx f(i+1,j) - f(i,j) \ := \ \text{the jump in the x-direction} \tag{1}$$
$$f_y(i,j) \approx f(i,j+1) - f(i,j) \ := \ \text{the jump in the y-direction.} \tag{2}$$

   Let us define the discrete gradient of an image f in the following manner:

$$\nabla f(i,j) := \langle f_x(i,j), f_y(i,j) \rangle. \tag{3}$$

   The magnitude of the discrete gradient vector associated to f is defined as the following:

$$|\nabla f(i,j)| := |\langle f_x(i,j), f_y(i,j) \rangle| = \sqrt{(f_x(i,j))^2 + (f_y(i,j))^2}. \tag{4}$$

   By staring at the above equation (4) for an extended period of time, one sees that the jump information in both the x-direction and y-direction is embedded into the magnitude of the gradient. Does this take into account a positive or negative jump? Does it discriminate? Why or why not?

   (b) Due to noise in an image, it is better to smooth the image first before applying the gradient. This can be done with the following commands:

```
K = fspecial('gaussian',9,9);
Kf = imfilter(f,K,'symmetric');
```

   where the new image Kf is the smoothed out one. Use this image when creating your edge detector. From the discrete gradient defined above, construct an edge detector that has value close to 1 near an edge and 0 away from it. Show this indeed works by testing it on the statue image obtained from my website.

   (c) Now, using the edge detector that you created, extract the pertinent edge information directly from the image. You can do this by utilizing an algebraic operation between your edge detector and the given image.

   (d) Extract the color edges from the 'toucan_color.bmp' image from my website. You can do this by creating an edge detector for each channel and then applying the extraction channel by channel.