CSE 166: Image Processing, Spring 2019 – Assignment 2

Instructor: Ben Ochoa

Due: Wednesday, April 15, 2020, 11:59 PM

Instructions

- Review the academic integrity and collaboration policies on the course website.
- This assignment must be completed individually.
- This assignment contains both math and programming problems.
- Programming aspects of this assignment must be completed using MATLAB.
- Unless specified below, you may not use MATLAB functions contained in toolboxes, including the image processing toolbox. Use the MATLAB `which` command to determine which toolbox a function is contained in. If you are unsure about using a specific function, then ask the instructor for clarification.
- You must prepare a report as a pdf file. The report must contain your solutions and results, and all of your MATLAB source code as a listing in the appendix of your report.
- Additionally, you must create a zip file containing all of your MATLAB source code.
- Your source code must contain a file `main.m` which runs all code necessary to produce results for your report. `main.m` should run start to finish without error. Use relative paths to read input data. For questions which require numerical output, `main.m` should print a message indicating what question is being answered followed by the numerical output for that question. Example: `display('Problem 2b')` followed by your answer to problem 2b. The instructors should be able to reproduce your report by running `main.m`
- You must submit both files (.pdf and .zip) on Gradescope. Further, you must mark each problem on Gradescope in the pdf file.
- It is highly recommended that you begin working on this assignment early.

Problems

1. **Textbook problems (10 points)**
   (a) Problem 3.5 (2 points)
   (b) Problem 3.7 (2 points)
   (c) Problem 3.11a (1 point)
   (d) Problem 3.12a (1 point)
   (e) Problem 3.24 (1 point)

(f) Problem 3.29 (2 points)

(g) Problem 3.44 (1 point)

2. **Programming: Intensity transformations and spatial filtering (30 points)**

   (a) **Bit Plane Slicing (5 points)**

   Develop a MATLAB function called `bitPlaneSlice` that computes the bit plane slices of a grayscale image. You may assume that the image is stored as an 8 bit unsigned integer (`uint8`). The function should take a $N \times M$ grayscale image as input and return an $8 \times N \times M$ array containing the bit plane slices of the image.

   Develop a MATLAB script called `hw2_bitslice.m` that reads the input image `cameraman.tif` (included with MATLAB) and calls `bitPlaneSlice` to compute the bit plane slices. Display the original image and each slice in the same figure. The first subplot in the picture should be the original image. Then display each of the bitplane slices in least to most significant bit order. Add a title to each subplot indicating which bit is displayed. Include the resulting figure in your report. Hint: use the `subplot` function to plot multiple images side by side in the same figure.

   (b) **Histogram equalization (5 points)**

   Develop a MATLAB function called `histogramEqualization` that calculates a histogram equalized image corresponding to a given image. The function input is a grayscale image and the function output is the histogram equalized image corresponding to the input image.

   Develop a MATLAB script called `hw2_histogram_equalize.m` that reads the input image `tire.tif` (included with MATLAB), computes the corresponding histogram equalized image, and writes the output image to `tire_histeq.png`. Use the function `imread` to read the input image in MATLAB. The script must call the function `histogramEqualization` to calculate the histogram equalized image. Use `imwrite` to write the output image in MATLAB.

   Include in your report the input and output images.

   (c) **Sharpening using the second derivative (10 points)**

   Develop a MATLAB function called `sharpen` that calculates a sharpened image corresponding to a given image. The function input is a grayscale image and the function output is the sharpened (double precision) image corresponding to the input image. The function must use the Laplacian filter (without diagonal directions). Note that $g(x, y) = f(x, y) - \nabla^2 f(x, y)$, where $f(x, y)$ is the input image and $g(x, y)$ is the output sharpened image. Filtering should be implemented without using the `imfilter` function.

   Develop a MATLAB script called `hw2_sharpen.m` that reads the input image `moon.tif` (included with MATLAB), computes the corresponding sharpened image and displays the output image. Use the function `imread` to read the input image in MATLAB. The script must call the function `sharpen` to calculate the sharpened image.

Include in your report the input and output images. Scale the output image such that zero is black and the maximum value in the sharpened image white. This can be accomplishing using `imshow(I, [])`. Briefly discuss your results. (Optional: sharpen the image using the Laplacian filter with diagonal directions and briefly discuss the qualitative differences between these results and those obtained using the filter without diagonal directions.)

(d) **Magnitude of gradient (10 points)**

Develop a MATLAB function called `gradientMagnitude` that calculates a magnitude of gradient vector image corresponding to a given image. The function input is a grayscale image and the function output is the gradient magnitude (double precision) image corresponding to the input image. The function must use the Sobel approximation to the derivative.

Develop a MATLAB script called `hw2_gradient_magnitude.m` that reads the input image `circuit.tif` (included with MATLAB), computes the corresponding magnitude of gradient vector image and displays the output image. Use the function `imread` to read the input image in MATLAB. The script must call the function `gradientMagnitude` to calculate the gradient magnitude image.

Include in your report the input and output images. Scale the output image such that zero is black and the maximum gradient magnitude value in the image is white. Briefly discuss your results.