CSE 166: Image Processing, Fall 2017 – Assignment 2

Instructor: Ben Ochoa

Due: Wednesday, October 18, 2017, 11:59 PM

Instructions

- Review the academic integrity and collaboration policies on the course website.
- 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 containing your solutions and results.
- Your report will be a pdf file named `CSE_166_hw2_lastname_studentid.pdf`, where `lastname` is your last name and `studentid` is your student ID number.
- All of your MATLAB source code must be included as a listing in the appendix of your report.
- Submit your report on Gradescope.
- Additionally, you must create a zip file named `CSE_166_hw2_lastname_studentid.zip`, where `lastname` and `studentid` is your last name and student ID number, respectively. This zip file will contain the pdf file and a directory named `code` that contains all of your MATLAB source code.
- Submit your completed assignment by email to `rkollipa@eng.ucsd.edu` **and** `asrikant@ucsd.edu`. The subject of the email message must be `CSE 166 Assignment 2`. Attach the zip file to the message.
- It is highly recommended that you begin working on this assignment early to ensure that you have sufficient time to correctly implement the algorithms and prepare a report.

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) **Negative transformation (5 points)**

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

   Develop a MATLAB script called `hw2_negative_image.m` that reads the input image `spine.png`, computes the negative image, and writes the output image to `spine_negative.png`. Use the function `imread` to read the input image in MATLAB. The script must call the function `negativeTransform` to calculate the negative image. Use `imwrite` to write the output image in MATLAB.

   Include in your report the input and output images.

   (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. You may use the function `imfilter` to apply the filter.

   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, [])`. Comment on the resulting output image. (Optional: sharpen the image using the Laplacian filter with diagonal directions and comment on 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. You may use the function `imfilter` to apply the filter.

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. Comment on the resulting output image.