Programming Assignment 2

Digital Image Processing (EE-402)

Image Acquisition and Enhancement

Dr. Adeel Mumtaz

National University of Computer & Emerging Sciences

Deadline: Till Wednesday 10 October

Attention: You can submit your assignment by the night (22h55) of last date. Late submissions will get zeros. You are welcome to consult each other and discuss the solutions among you, but everybody should write their own code. In case of copying both (or all) the assignments will get zero (no exceptions). You are welcome to ask questions and post queries on Slate.

Description

In this assignment, you will practice/implement the studied image sampling, quantization and point processing algorithms (arithmetic and histogram based) from lectures (7, 8, 9). Other goal of the assignment is to get yourself familiar with an image processing tool for reading, writing and basic matrix operations. Note that for each of the following task you have to write your own code. **You should not use the toolbox/package/library built-in functions**. You can only compare your results for correct implementation with built-in functions. A set of sample images to test your implemented tasks are also attached.

What to Submit

You are required to develop and submit source code of a single GUI based image processing application for all tasks, in any programming language (e.g., Matlab, Python, C#). User should be able to perform following sequence of operations for each task/algorithm in the assignment:

- Select the task number or algorithm from list Box
- Browse/Load input image/images for the selected algorithm
- Set values for selected algorithm parameters
- Press the process button to see the results in output image windows alongside input images. You should also provide an option to save the result image into file.

Figure 1 & 2 shows sample GUI application with expected results.

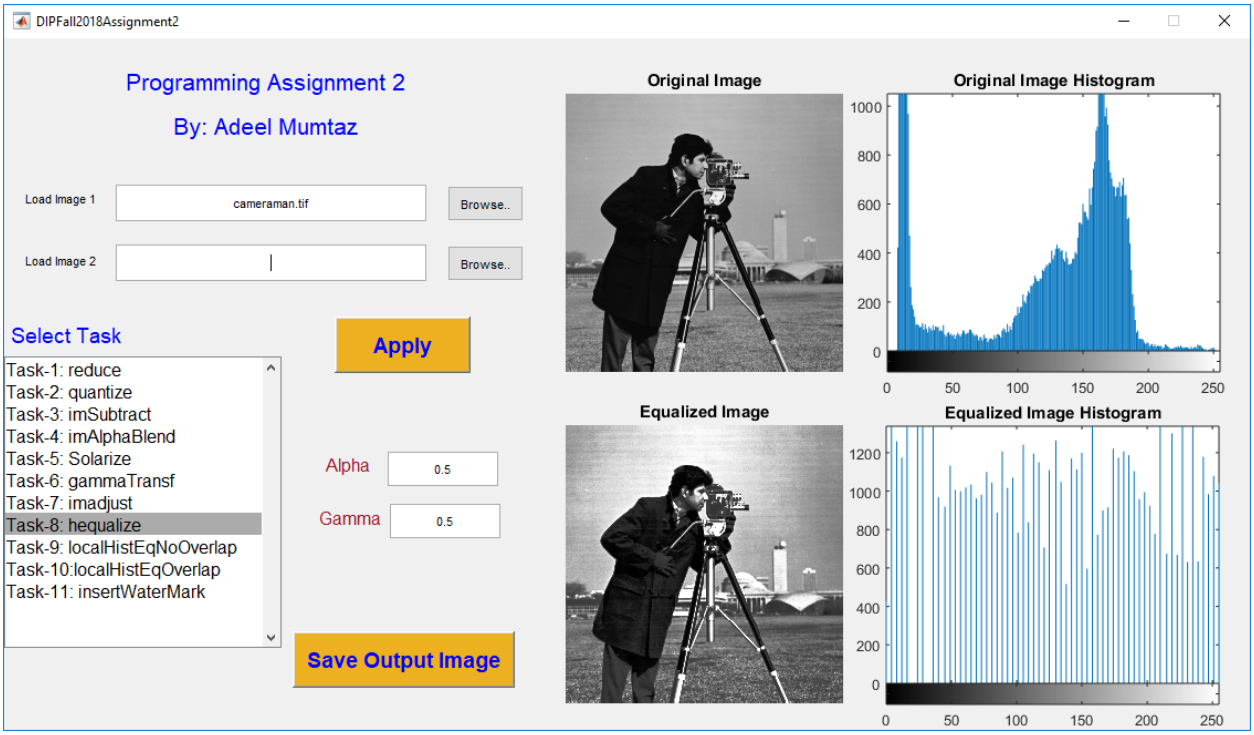

Figure 1.0: Histogram Equalization Results

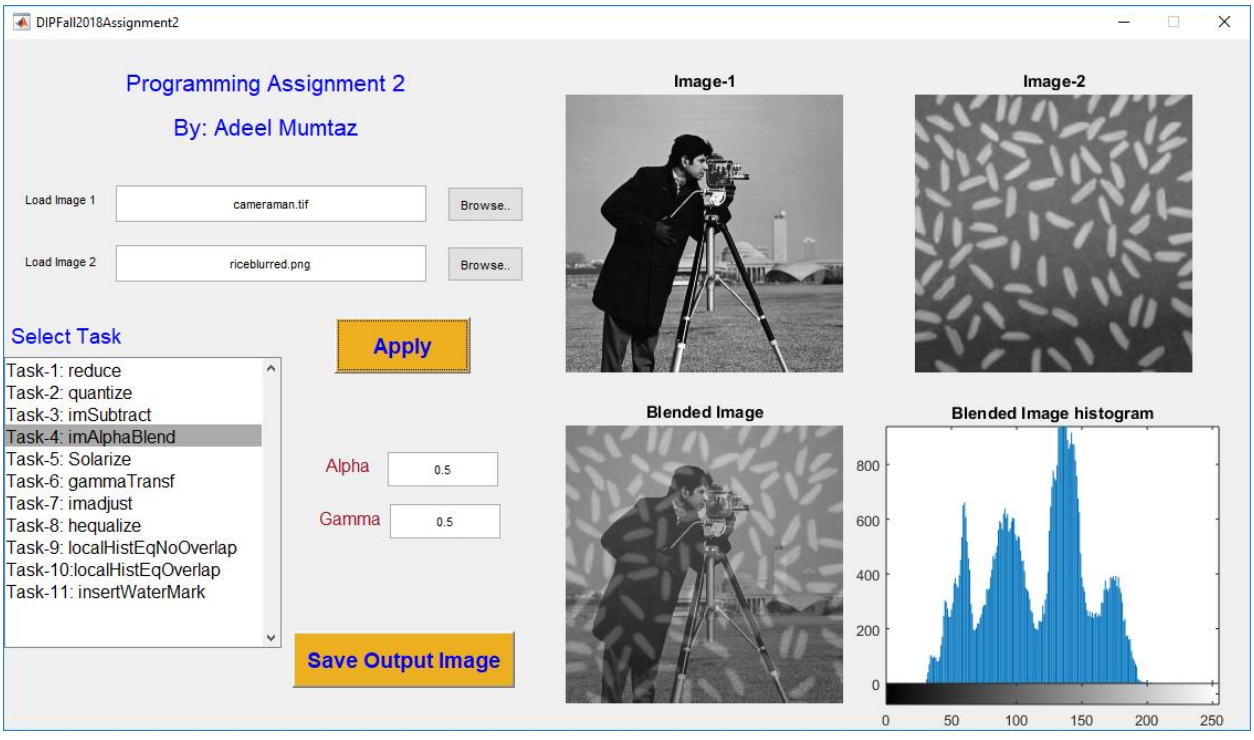

Figure 2.0: Alpha Blending Results

Tasks:

1. Write a function ***reduce*** that takes two arguments as input, an image and a sampling ratio r ($0 \leq r \leq 1$), and scales the input image to the given ratio, e.g. reduce(I,0.5) should reduce the image by half.
2. Write a function ***quantize*** that takes two arguments as input, an image and number of bits b($0 \leq b \leq 8$), and represent the input image using specified quantization depth, e.g. quantize(I,2) should represent the image using only four gray levels, etc. You can easily notice the saturation noise and contour effects by displaying input and output images.
3. Write a function ***imSubtract*** which takes two images of same size as an input and output their difference image.
4. Write a function ***imAlphaBlend*** which takes two images of same size and alpha parameter as an input and provide the blended image as an output using alpha blending algorithm.
5. Write a function ***solarize*** to produce solarization effects on input image (see lecture notes for details.)
6. Write a function ***gammaTransf*** to perform the gamma transformation over the input image using given c and γ parameters. Plot the mapping function and images before and after the gamma transformation.
7. Write ***imadjust*** function to adjust the contrast of a given image. You should write a function that takes three input arguments: (i) image, (ii) input range [a, b] and (iii) output range [c, d] and give as output the contrast adjusted image. Display the transformation function and input/output images.
8. Write a function ***hequalize*** for histogram equalization. The function should automatically equalize the histogram of given input image. You can test your function on Night Time Road Contrast Enhancement Application. To test your function, you can use provided dark_road_1.jpg, dark_road_2.jpg, and dark_road_3.jpg images. For all three images display input image, its histogram, output image and its histogram.
9. Write a function ***localHistEqNoOverlap*** that accepts an image and a window size as input and perform local histogram equalization. Hint: Partition the given image into non-overlapping windows of given window size and perform histogram equalization in each window. Display input image, its histogram, output image and its histogram.
10. [Sliding Window Version] Write another version ***localHistEqOverlap*** of the above function to have overlapping windows based local histogram equalization. Scan your image like a linear filter with the window (moving 1 pixel) and perform local histogram equalization. Display input image, its histogram, output image and its histogram.
11. Write a function ***insertWaterMark*** which should take one grayscale image and one binary image of same sizes as an input. Embed/Hide the binary image into the least significant bit of the gray image. Display gray, binary and embedded images.
12. Save an embedded image from Task-11. Apply histogram equalization (Task-8) to this image. Now ***extract the binary image back and display***.