EE371Q Digital Image Processing

Homework #1

Homework Instructions

- This homework is due two weeks from now, on **September 30th at 11:59pm**.
- You are encouraged to work with other students to understand the course material. However, sharing source code, images, or other answers from any homework is strictly forbidden. You are to submit your own work.
- Please include your written answer and/or output images for each problem.
- Please show all steps for full credit and attach the code to each problem.
- **All the images needed for this homework can be downloaded as .jpg files from Canvas under the HW1 folder in the Files tab.**
- The homework should be submitted as a PDF file. While submitting on Canvas, zip the PDF file with the images you generated, and submit as a single zipped file.

Problem 1 - Basic Image Processing (25 points)

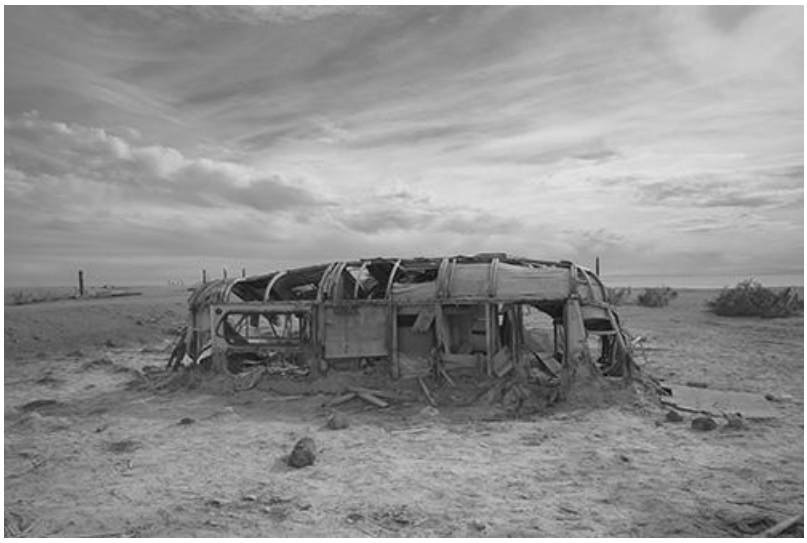
In this problem, you will be performing a few basic operations on images. You will be working on the image given below, which can be downloaded from Files -> HW1 -> Images tab in Canvas.

a. (2 points) Read the image and display it in a new window. Make sure it resembles the image that you downloaded.

b. (3 points) Separate the image into its three color components; Red, Green and Blue. Show these three color component images along with the original image in a 2x2 grid with labels.
c. (5 points) YUV is an older color space described in Module - 1, with the equations to compute Y, U and V from RGB. Convert the original image to YUV format, i.e. generate three images of Y, U and V components separately by using the converting equation. Display all three of these images side-by-side.
d. (5 points) Convert the original image to grayscale. You should write your own function to create the grayscale from the color image. Create 4 different images each with different color intensity weightings, one of which should have equal color intensity weighting for R, G and B. Display these 4 images in a 2x2 grid with labels.
e. (5 points) Rotate the **top left quarter** of the image by 180 degrees. You cannot use the built-in function ***imrotate.*** Display the updated image in a new window.
f. (5 points) Flip the original image horizontally (along the central vertical axis) and vertically. Display the original image and the two flipped images in a 1x3 grid with labels. You cannot use the built-in functions ***fliplr and flipud***.

Problem 2 - Histogram Manipulation (25 points)

We will now be generating the histogram of an image, and performing operations on the histogram to enhance the contrast of the image. Do not use the built-in functions ***imhist, imadjust or histeq*** functions (Helpful functions: hist, bar, min, max, cumsum). You should create your own histogram manipulation functions.



q2.jpg

a. (5 points) Read the image in grayscale. Compute and plot the histogram of the image using ***hist*** and ***bar***, and display it in a new window with a label. You are not allowed to use the ***imhist*** inbuilt function. Is the image underexposed or overexposed?
b. (5 points) You need to now implement the FSCS algorithm taught in Module-2. Write a function **"func_fscs(image)"**, which takes the input as the original image and gives output as the enhanced/processed image. Pass q2.jpg through the function, and display the original image, original histogram, processed image and processed histogram in a 2x2 grid with labels.
c. (5 points) You need to now implement the logarithmic contrast compression. Write a function **"func_logContCompression(image)"**, which takes the input as the original image and gives output as the logarithmic contrast compressed image. Take q2.jpg, pass it through **"func_logContCompression(image)"**, and pass the output of this function through **"func_fscs(image)"** (FSCS algorithm). Display the logarithmic contrast compressed image, its histogram, the final image after passing through both the functions and its histogram in a 2x2 grid with labels.
d. (5 points) You need to now implement the gamma correction algorithm. Write a function **"func_gamma(image)"**, which takes the input as the original image and gives outputs as the gamma corrected image. Play around with different values of 'gamma' and see which looks best. Pass q2.jpg through the function for this gamma value, and display the processed image and its histogram side by side. Also mention the value of 'gamma' used in your report.
e. (5 points) You now have three processed images; FSCS, logarithmic range compression followed by FSCS, and gamma correction. Display the original image and the three processed images in a 2x2 grid with labels. Analyze these three processed images. Which of these three algorithms performs best? Give a brief reasoning on why that algorithm performs best for q2.jpg.

Problem 3 - Binary Image Morphology (30 points)

We are going to apply morphological operators on the two images given below, and analyze the effects of these operations.

flowers.jpg

zebra.jpg

a. (5 points) Read both the images in grayscale. Before running binary morphological operations on the images, we need to binarize the images. Thus, run thresholding on both the images with a suitable threshold. Display the two thresholded images side by side. Which image is thresholding more effective on? **For the subsequent parts of this question, you will be working with these thresholded images, and not the original images.**
b. (10 points) Write two functions “func_dilate()” and “func_erode”. Both of the functions should take two input arguments, the image and width of the square window, and give one output image. You can use padarray, im2col, reshape, max and min. You cannot use any for or while loops in your program. Display the dilated and eroded images of both flower.jpg and zebra.jpg (thresholded images) in a 2x2 grid with labels.
c. (5 points) By slightly modifying “func_dilate” and “func_erode”, write functions for open, close, open-close, close-open and median. Run zebra.jpg (thresholded) through each of these five functions and display the five output images in a 1x5 grid with labels.
d. (5 points) Generate a clean binary image from the thresholded zebra.jpg image by removing blobs and holes. Use a combination of your erode, dilate, open or close operations for the same. With this clean binary image, generate a boundary of the object. Display the boundary image and the thresholded zebra.jpg image side by side.
e. (5 points) Write a function “func_boundaryLength()” to calculate the length of the boundary, and report the boundary length.

Problem 4 - Bit-Planes (15 points)

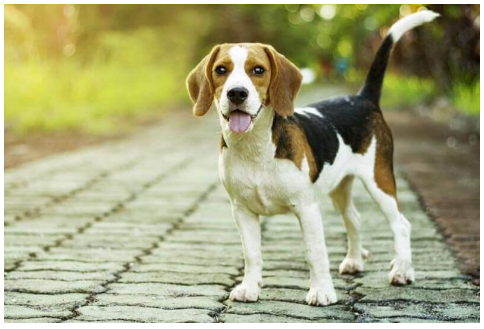
We are given an image, we will be playing with the bit-planes of the image now.

q4.jpg

a. (5 points) Read the image q4.jpg. Modify it using a 3-bit quantization bar. Essentially, the intensity of each pixel should be represented by 3 bits instead of the original 8 bits. Display the 3-bit image and the original image side by side.
b. (5 points) Use ***bitget*** to extract each of the 8 bit-planes, and display the 8 bit-planes in a 2x4 grid with labels.
c. (5 points) We are now going to play around with hidden information in images. Add some hidden information in bit-plane 1 and reconstruct the image. Generate seven more images by adding some hidden information in bit-plane 2, bit-plane 3 and so on. You will have a total of eight images at the end of this. Display all the eight images with “hidden information” in a 2x4 grid with proper labels. Are any of these images perceptually distorted compared to the original image? If so, which ones?

Problem 5 - Open-ended (5 points)

You are given an image below. You are free to use any of the techniques and concepts learnt in the course till date on this image. After your transformations, display the original image and the new image side by side. Describe briefly what were the transformations you implemented.



q5.jpg

This question is worth only 5 points, and is an opportunity for you to try out transformations which have not been tested in this homework. It doesn't have to be a super complex solution. Feel free to implement transformations that have not been taught in class too, but make sure to explain what you have done!