EE 371Q – DIGITAL IMAGE PROCESSING

HOMEWORK #4

Due on November 29th at 2:59pm

- You are encouraged to work with other students to understand the course material. However, sharing source code, images, or other answers from outstanding 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.
- Images needed for this homework can be downloaded from Canvas under the “Files” tab.
- This homework will help you to familiarize yourself with Python if you have not worked with it in the past. The internet is a great resource for finding help with Python functions and will be available for you to use.
- You can use either MATLAB or Python to solve these problems.
- Please provide screenshots and necessary code explanation for each question and show them in one pdf document.

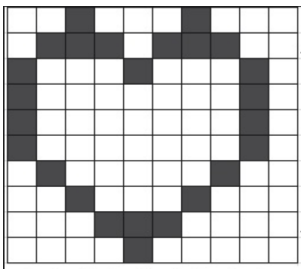
Human Study as an alternative to HW

We will be conducting a Video Quality Assessment study from Nov 14th - Dec 5th in EER 6.866. Participants will watch videos either with a VR headset or on the 2D display (optional) and rate the quality of the videos (not the content) at the end of each video. There will be two viewing sessions. Each session will take 40 minutes and must be spaced apart by at least 24 hours. Participants will be given a sign-up sheet where they can choose their preferred slots for the study. Here is the link to the sheet: https://docs.google.com/spreadsheets/d/1YJ5rTiBSGgfFFbc8FPZgP8sDkY9TA3TA9BPOhnm_7vk/edit?usp=share_link.

Please read the instructions at the top of the sheet before filling it. (* Please write your EID so that we can know that you're taking an image processing class and you choose to participate in this human study to receive full credits on the homework.) After you sign up, we will contact you with further details on the study and how to enter EER, etc. Yu-Chih (Berrie) Chen (berriechen@utexas.edu) and Avinab Saha avinab.saha@utexas.edu) are the Graduate Students conducting the study. Please contact them if you have any questions.

Question 1 : Huffman Binary Tree (40 points)

Huffman coding is a lossless image compression technique. In this problem, we will understand the Huffman coding algorithm better, but will not be required to implement it. PS; you will not be required to generate the compressed image from the original image.



heart.jpg

a) (5 points) Read the image heart.jpg, convert the image to grayscale with each pixel ranging between 0 and 15 (only 4 bits to represent each pixel), and display the histogram of the image.

b) (15 points) Write a python/matlab function to generate the Huffman tree. The function should take the input parameter as an array of size 16 which contains the corresponding number of occurrences for each pixel value. The function should then calculate the probabilities for each pixel value, and then generate the Huffman tree as described in the module slides. Display the Huffman tree for heart.jpg with proper formatting by generating the input array of size 16 for the image and passing through the function.

c) (10 points) Write a python/matlab function which extends/uses the previous function, and additionally calculates the codeword for each pixel value from 0 to 15. The function should take the input parameter as an array of size 16 with the corresponding number of occurrences for each pixel value. Display the codeword generated for each pixel value in heart.jpg by generating the input array of size 16 for the image and passing through the function.

d) (10 points) Write a python/matlab function which extends/uses the previous function, and additionally calculates the new BPP and the compression ratio after applying Huffman coding. The function should take the input parameter as an array of size 16 with the corresponding number of occurrences for each pixel value, and the output of the function should be the new BPP and the

compression ratio. Report the BPP and compression ratio achieved for heart.jpg by generating the input array of size 16 for the image and passing through the function.

Question 2 : Reference Image Quality Assessment (30 points)

In this problem, we will be using MSE and PSNR to quantify the quality of different images with respect to an original image.

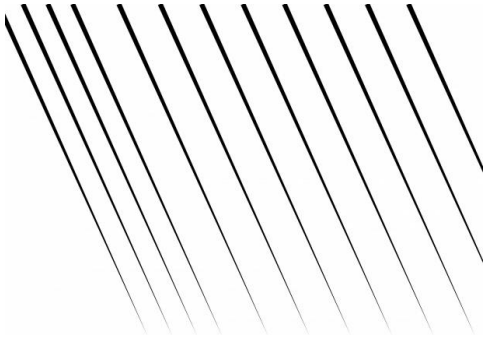
Golden.jpg

a) (5 points) Convert the golden.jpg image to grayscale, and apply FSCS on the resulting grayscale image. Display the original grayscale image and the image after FSCS side-by-side with appropriate labels.
b) (5 points) Convert the golden.jpg image to grayscale. Randomly choose 20% of the pixels and change their intensity to 0. Display the original grayscale image and the resulting grayscale image side-by-side with appropriate labels.
c) (5 points) Convert the golden.jpg image to grayscale. Apply the gamma correction algorithm on this grayscale image for a gamma value of 0.95. Display the original grayscale image and the resulting gamma corrected grayscale image side-by-side with appropriate labels.
d) (10 points) Write a python/matlab function to take in two grayscale images as input, and give the MSE and PSNR of the second image with reference to the first image (original image) as the output.
e) (5 points) Run the function written in the previous parts, and report the MSE and PSNR for the images generated in parts a), b) and c) with reference to the original grayscale golden.jpg image. Report these values in a table with clear

labels. Comment a few lines on the MSE values observed, the similarities and differences between the MSE values of the different images, and whether MSE captures the quality of an image effectively.

Question 3 : Hough Transform (30 points)

In this problem, we will be utilizing the hough transform to detect lines in the lines.jpg image.



Lines.jpg

a) (10 marks) Consider Question 3 of HW2. This question is a continuation of that question. Refer to HW2 in Canvas → Files tab to access the Question 3 of HW2. If you have not completed Q3 in HW2, complete it now and display the four DoG filtered images of lines.jpg (instead of flowers.jpg asked in “HW2 Q3 part c”) in a 2x2 grid with the appropriate labels.
b) (10 marks) One useful application of DoG filtering is edge detection. After DoG filtering, zero-crossings occur near the center of edges, where the DoG goes from positive to negative or from negative to positive between neighboring pixels. On the four filtered images of lines.jpg, find horizontal and vertical zero-crossings to generate the corresponding edge maps. In the edge maps, set the value of pixels corresponding to zero-crossings to 1 and the rest to 0. Display the four edge maps and write a few lines on your observations.
c) (10 points) In this part, we will be using the hough transform to detect lines in the edge map generated in the previous question. Take any one of the four edge maps generated. Write a function in python/matlab which takes in the input parameter as the edge map image, and gives the output as a grayscale image

with only the detected lines. In this function, you will use an internal array of “rho x theta” size, and use this as a hough accumulator and increment an index as you find a line. Then, use a suitable threshold, and every index in the array greater than this threshold should be considered a line. Then with the (rho, theta) pairs found above the threshold, generate a grayscale image with the lines generated using these (rho, theta) pairs. Run one of the edge maps of lines.jpg through this function, and display both the original lines.jpg image and the generated grayscale “line detection” image side-by-side with the appropriate labels.