EE371Q Digital Image Processing

Homework #3

- This homework is due two weeks from now, on **November 1st 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. If using Python, Jupyter Notebooks can allow you to generate a PDF for submission quite easily.
- **All the images needed for this homework can be downloaded as .jpg files from Canvas under the HW3 folder in the Files tab.**
- The homework should be formatted as a PDF file. You should also submit your source code file (if there are multiple files, submit it as a zipped file)

Question 1: Band-Pass Filter (20 points)

The goal of this problem is to implement a band-pass filter using two Gaussian filters. The DoG filter (The Difference of Gaussian) is defined as the difference of two Gaussian kernels with different variances σ_1 and σ_2. For simplicity, let $\sigma_2 = k\,\sigma_1$ for some k and subtract the Gaussian kernel with variance σ_1 from the kernel with σ_2. (Helpful functions: getGaussianKernel, plot_surface, copyMakeBorder, filter2D)

cars.jpg

a) (10 points) You need to write a 2d-DoG function, “myDoG(DoGsize, sigma1, k)”. This function produces a kernel of size DoGsize that is the difference of two gaussian kernels with corresponding variances σ_1 and σ_2 = $k\ \sigma_1$.

b) (5 points) Use this function to generate four 2d-DoG filters with σ_1 = 1, 2, 3, 4, k = 1.5, and a window size ten times σ_1. Show the 3d illustration of the four 2d-DoG filters in a 2x2 grid with appropriate labels.

c) (5 points) Read in cars.jpg and convert it to gray-scale. Generate the same four 2d-DoG filters as in part b. Apply the filters to cars.jpg using a method of your choosing (recall the methods used in HW2). Display the filtered images in a 2x2 grid with the appropriate labels. Write a few lines on what you observe, and why.

Question 2 : Non Local (NL) Means (40 points)

Here we will be looking at the application of Non-Local Means in removing noise. Helpful functions: skimage.util.view_as_windows, scipy.spatial.distance.cdist, padarray, im2col, pdist2, sum, reshape.

bird.jpg

a) (5 points) You need to read bird.jpg, convert it to grayscale, and resize the image so that it is a perfect square. Display this resulting image.

b) (15 points) You will now calculate the Luminance Similarity Measure “W(m,i)” between all the 3x3 square windows in the image, using the information given in Module 5 slide 14 onwards. Use K_w = 1 and σ_w = 1. Normalize “W(m, i)” so that the sum across each row is equal to 1.0. Note that even though this is a small image, this step will likely take a while to execute (a few minutes depending on hardware).

c) (5 points) From all the windows, we will now compare the similarity measure images generated between choosing two different windows in the image. For this part, take the 3x3 window in the top-left corner of the image, and reshape it to a 3x3 matrix. Take the first row in W which corresponds to all of the pair-wise distances to that 3x3 window, and reshape it to the size of the image. Display the resulting window and image side by side with appropriate labels.

d) (5 points) You will now need to repeat the same operation you did in part c), but by taking the 3x3 window in the middle of the left side of the image. Display the resulting window and the original image side by side with appropriate labels.

e) (10 points) Using the luminance similarity measure, perform Non Local (NL) Means filtering on bird.jpg, and remove the noise. Display the resulting image and the original image in a 2x1 grid with appropriate results. Comment a few lines on the result achieved.

Question 3: Block Truncation Coding (BTC) (25 points)

BTC is a fast and lossy compression technique. In this problem, we will understand how BTC is used, along with its advantages and disadvantages. 4x4 blocks are used where applicable in the following sub-problems. You can use the following link to learn more about BTC: https://en.wikipedia.org/wiki/Block_Truncation_Coding#Encoder . Helpful functions: col2im, im2col, bi2de, de2bi.

bevo.jpg

a) (5 points) You should read bevo.jpg, and make it grayscale. Resize the image by a factor of ½ to make the compression faster. Resize the image further such that the height and width of the images are both multiples of 4; such that it is still roughly the same

dimensions as ½ of the original image dimensions. Display the grayscale compressed image.

b) (5 points) Write a function to compute the mean and standard deviation of a 4x4 block with ‘B1’ bits to compute and store the mean, and ‘B2’ bits to compute and store the standard deviation; where ‘B1’ and ‘B2’ are input parameters to the function. Hint: you might need to compute the mean and standard deviation values, and then truncate in the end based on ‘B1’ and ‘B2’ parameter values.

c) (5 points) Write a function to compute the 4x4 BTC binary block by thresholding the input 4x4 image block at the mean. Pass bevo.jpg through this function for 3 different ‘B1’, ‘B2’ settings; {2,1}, {3,3}, and {7,5}. Compute the BTC binary blocks image for bevo.jpg, and display the 3 images in a 3x1 grid with appropriate labels.

d) (5 points) Write a function to decode a 4x4 BTC binary block given the mean and standard deviation as the input parameters, and get back the original 4x4 image block. Pass the 3 BTC binary block images generated in part c) through this function, for the aforementioned ‘B1’, ‘B2’ settings; and get back the 3 decompressed bevo.jpg images. Display these three images along with the original image in a 2x2 grid with appropriate labels.

e) (5 points) Compute and report the compression ratios for the 3 BTC encoded images generated.

Question 4: VGG19 Feature Extraction (15 points)

VGG19 is a convolutional neural network architecture used primarily for image classification. The network is primarily composed of 3x3 convolutional filters, making it effective at capturing intricate and abstract features in images. In this problem we will utilize a **pre-trained** VGG19 network to extract features maps from an image. These features can be very useful for other applications; for example, you can compare two images by comparing their corresponding extracted feature maps instead of using their original representations which can help highlight perceptually relevant differences. For more on this see https://arxiv.org/pdf/1603.08155.pdf

lion.jpg

a) (5 points) Download and load a pre-trained version of the VGG19 model. Most machine learning frameworks should have a way to download and use pre-trained networks directly. With PyTorch you can use: https://pytorch.org/vision/main/models/generated/torchvision.models.vgg19.html

b) (5 points) Read lion.jpg and transform it so that it is suitable for the model. The corresponding transform may depend on the pre-trained version of the model that you are using. For the default version in PyTorch, you will need to resize the image to 224x224 and normalize it to the ImageNet dataset mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. Display the transformed image.

c) (5 points) Experiment with passing the transformed image through layers of the network. Note that you may need to add a dummy batch dimension to the input tensor for the model to accept it. Find three feature maps that look interesting to you and display them. In Pytorch you can get a list of vgg19's layers by accessing its *features* field. Then you can use slicing to use only a subset of these layers and get the results of passing your input through them.