EE 371Q – DIGITAL IMAGE PROCESSING

HOMEWORK #2

Due on October 20th 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 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.
- In all questions that ask you to process a color image directly (without converting to gray-scale), process the luminance part or the intensity (R+G+B) only.

Study as an Alternative to HW

Instead of submitting Homework 2, you have an alternative option to receive 100% on this assignment: participate in an interesting video quality assessment human study!

This study is attempting to understand how humans rate video quality for videos on a TV in a living-room setting. The main requirement of participation is that you have normal or corrected-to-normal vision.

You will be shown several videos on a TV and be asked to evaluate the quality of the video after each video.

This study is divided into two separate test sessions. There will be a short training session before the test session. Each test session lasts approximately 40 minutes. Given each video, you will be asked to provide an opinion score of video quality (not content).

The content will be from various sources and could include clips from movies, documentaries, and sports.

This data will be used for research purposes. No release of the data will contain any identifying information that could associate you with it.

We will send a sign-up sheet for this study (via Canvas or Rishabh, the TA) within a week.

Contact Josh (joshuaebenezer@utexas.edu) or Zaixi (zxshang@utexas.edu) if you have any doubts or questions.

Question 1: Discrete Fourier Transform (30 marks)

The goal of this problem is to use DFT and understand the roles of magnitude and phase components of an image in the frequency domain (Helpful functions: np.fft.fft2, np.fft.ifft2, np.fft.fftshift, np.fft.ifftshift, np.abs, np.angle, real, mesh).

basketball.jpg

soccer.jpg

a) (5 marks) Read both the images, basketball.jpg and soccer.jpg, and convert them to grayscale. Display both the colored and the corresponding gray-scale images in a 2x2 grid.

b) (5 marks) Compute the DFT of the gray-scale version of basketball.jpg. Calculate the DFT phase component and the logarithm of the DFT magnitude component, and perform full-scale contrast stretch (FSCS) on both these components. Show the results side by side. Please move the low frequency part to the center (helpful functions: np.fft.fft2, np.fft.fftshift).

c) (5 marks) Add “π” to the original DFT phase component of the gray-scale version of basketball.jpg. Perform the inverse DFT to reconstruct the image using the modified DFT phase (after adding “π”) and the original DFT magnitude components (use the components before FSCS). Show the original and the reconstructed scotland.jpg side by side. Can you explain the result mathematically? (Hint: Keep only the real part of the inverse DFT, because the imaginary parts may be generated due to rounding errors.)

d) (5 marks) Increase the original DFT magnitude component of the gray-scale version of basketball.jpg to the following powers: 1.5 and 0.3. Perform the inverse DFT to reconstruct the image using the modified DFT magnitude and the original DFT phase components (before the FSCS). Show the two reconstructed images and the original image in a 1x3 grid. Can you explain the effects you observe after the exponentiation operation? (helpful function: mesh).

e) (5 marks) Truncate the shifted DFT component of the gray-scale version of basketball.jpg by a truncation window (you can determine the size and shape) to keep

~25%, ~12.5%, and ~6.25% of the DFT coefficients in center. Perform the inverse DFT to reconstruct the image using the modified DFT (remember to undo the FSCS if required). Show the original and the three reconstructed basketball.jpg images in a 2x2 grid with the appropriate labels. (Hint: Keep only the real part of the inverse DFT, because the imaginary parts may be generated due to rounding errors.)

f) (5 marks) Let the original DFT magnitude component of the gray-scale version of basketball.jpg be the same, and replace its DFT phase component with that of the gray-scale version of soccer.jpg. Perform the inverse DFT to reconstruct this phase-off image. Show the resulting image, with full-scale contrast stretch if necessary, and the original basketball.jpg image side by side. Write a few lines about what you observe, regarding what information the magnitude and phase components carry in the frequency domain.

Question 2: Linear and Non-Linear Image Filtering (45 marks)

The goal of this problem is to understand some types of image noise, and compare linear and non-linear image filtering. (Useful functions: getGaussianKernel, filter2D, median, np.random.normal, np.random.randint)

goat.jpg

a) (10 marks) You need to write a function, "**myAvgFilter()**". The function will have the parameters as follows: myAvgFilter(image, window_size). The function should implement the average filtering on an image with square windows (helpful function: filter2D).

b) (10 marks) You need to write a function, "**myMedFilter()"**. The function will have the parameters as follows: myMedFilter(image, window_size). The function should implement the median filtering on an image with square windows (helpful function: median).

c) (10 marks) You need to write a function, "**myGauFilter()"**. The function will have the parameters as follows: myGauFilter(image, window_size, sigma). The function should

implement the Gaussian filtering on an image with different window sizes and σ (standard deviation) (helpful function: getGaussianKernel).

d) (5 marks) Read the image, **goat.jpg**, and convert it into gray-scale. Add 5% salt-and-pepper noise to the image. This means on an average, 2.5% of the pixels are changed to 255 and 2.5% of the pixels are changed to 0 (helpful function: np.random.randint). Apply the three different filters you implement above to the noisy image. Show the noisy image and the best de-noised results from the three filters in terms of MSE (Mean Squared Error) in a 2x2 grid (with appropriate labels), and report both the resulting MSE and the parameters you picked for each filter. Write a few lines explaining the results and why the filter that works best does so.

e) (5 marks) Read the image, **goat.jpg**, and convert it into gray-scale. Add Gaussian white noise of zero mean and σ= 0.1 to the image (helpful function: np.random.normal). Apply the three different filters you implement above to the noisy image. Show the noisy image and the best de-noised results from the three filters in terms of MSE in a 2x2 grid (with appropriate labels), and report both the resulting MSE and the parameters you picked for each filter. Write a few lines explaining the results and why the filter that works best does so. (Hint: Add the Gaussian white noise to the normalized image, i.e., intensity scaled to [0 1], or alternatively, scale the Gaussian white noise based on the range of image intensity.)

f) (5 marks) Could "**myAvgFilter()**" be used for image sharpening? If yes, suggest how? If not, explain why not? Note: If yes, you do not need to write any code, just a brief explanation.

Question 3: Band-Pass Filter (25 marks)

The goal of this problem is to implement a band-pass filter using two Gaussian filters in the DFT domain. The **DoG filter (The Difference of Gaussian)** is defined as the difference of two Gaussian kernels with different variances $\sigma 1$ and $\sigma 2$. For simplicity, let $\sigma 2 = k\,\sigma 1$ for some k and subtract the Gaussian kernel with variance $\sigma 1$ from the kernel with $\sigma 2$. (Helpful functions: copyMakeBorder, np.fft.fft2, getGaussianKernel, plot_surface)

flowers.jpg

a) (10 marks) You need to write a function, **"mylinearfilter(image, h)"**. This function should filter an image by pointwise multiplication in the DFT domain with a frequency response template 'h' (Hint: don't forget to account for the wrap-around effect).

b) (10 marks) You need to write a 2d-DoG function, **“myDoG(DoGsize, sigma1, k)”**. This function produces a frequency response template of size DoGsize. Use this function to generate four 2d-DoG filters with $\sigma 1$ = 1, 2, 3, 4, k = 1.5, and a window size ten times $\sigma 1$. Show the 3d illustration of the four 2d-DoG filters in a 2x2 grid with appropriate labels.

c) (5 marks) Read in **flowers.jpg** and convert it to gray-scale. Generate the same four 2d-DoG filters except with the window size equal to the size of flowers.jpg. Use your “**mylinearfilter()”** function to filter flower.jpg using the four different filters. Display the filtered images in a 2x2 grid with the appropriate labels. Write a few lines on what you observe, and why. Note: The filtered images have both positive and negative responses. Hint: Make sure image dimensions are odd (Python likes odd window sizes), you can trim an edge(s) off the image to make the dimensions odd.