E9 241 Digital Image Processing

Assignment 03

Due Date: October 21, 2022 - 11:59 pm **Total Marks:** 70 + 20

Instructions:

For all the questions, write your own functions. Use library functions for comparison only.

- Your function should take the specified parameters as inputs and output the specified results.
- Also provide the wrapper/demo code to run your functions. Your code should be self-contained, i.e., one should be able to run your code as is without any modifications.
- For python, if you use any libraries other than `numpy`, `scipy`, `scikit-image`, `OpenCV`, `pillow`, `matplotlib`, `pandas` and default modules, please specify the library that needs to be installed to run your code.
- Along with your code, also submit a PDF with all the results and inferences. Include answers to subjective questions, if any.
- Put all your files into a single zip file and submit the zip file. Name the zip file with your name.
- **Vectorize your code. Non-optimized code may be penalized.**

---

1. **Frequency Domain Filtering:**

   (a) Generate $M \times N$ sinusoidal images using $\sin(2\pi u_0 m/M + 2\pi v_0 n/N)$ for $M = N = 501$ and compute their DFT. Take $(u_0, v_0) = (40, 60)$ for the first image (Image A) and $(u_0, v_0) = (20, 100)$ for the second image (Image B). To visualize the DFT of an image, take the logarithm of the magnitude spectrum and apply contrast stretching. Comment on the actual images and their DFTs.

   Add both the DFTs pointwise and find the IDFT of their sum. Compare the result with the pointwise addition of Image A and Image B in the pixel domain.

   Note: Fast Fourier Transform (FFT) is an algorithm that is used for efficient computation of DFT of discrete signals. You can use MATLAB (or python) built-in function for computing the 2D FFT and IFFT.

   (b) Pass the image `dynamicSine.png` through an ideal low pass filter (ILPF), ideal bandpass filters (IBPFs), and an ideal high pass filter (IHPF).

   - The expression for the ILPF is

   $$H_{\text{ILPF}}(u, v; D_0) = \begin{cases} 1 & D(u, v) \leq D_0 \\ 0 & D(u, v) > D_0 \end{cases}$$

   where D_0 is a positive constant referred to as the cut-off frequency and $D(u, v)$ is the distance between a point (u, v) in the frequency domain and the center of the frequency rectangle, i.e., $D(u, v) = \sqrt{(u - P/2)^2 + (v - Q/2)^2}$, where P and Q are the number of rows and columns in the image.

   - An IHPF with a cut-off frequency at D_0 can be computed as

   $$H_{\text{IHPF}}(u, v; D_0) = 1 - H_{\text{ILPF}}(u, v; D_0)$$

   - An IBPF with a lower cut-off at D_l and a higher cut-off at D_h can be computed as a product of ILPF and IHPF.

   $$H_{\text{IBPF}}(u, v; D_l, D_h) = H_{\text{ILPF}}(u, v; D_h) \times H_{\text{IHPF}}(u, v; D_l)$$

Perform different frequency domain filters on the DFT of the input image and find their IDFT to get the filtered outputs. Use the following parameters for each filter:

Filter	Function
Low pass	ILPF(20)
Band pass - 1	IBPF(20, 40)
Band pass - 2	IBPF(40, 60)
High pass	IHPF(60)

Plot the filter and the filtered images. What do you observe in the resultant images? How do they relate to the filters?

Note: The filters given above are centered in the frequency domain. To use such centered filters, you will either need to shift the filter to $(0, 0)$ (by using fftshift in MATLAB or the corresponding function in python) or center the DFT of the image. To center the DFT of the image, you can either shift the DFT of the image or scale each image pixel $I(x, y)$ by -1^{x+y} before computing its DFT. If you center the DFT of the image, then you will need to compensate for it by multiplying the image obtained from the inverse DFT of the filtered image by -1^{x+y}.

(c) Filter the image `characters.tif` in the frequency domain using an ILPF and the Gaussian low pass filter (GLPF) given by

$$H_{\text{GLPF}}(u, v; D_0) = exp(-D^2(u, v)/2D_0^2)$$

where all the terms are as explained in the last part. For $D_0 = 100$, compare the results. Do you observe any artifacts?

(10+20+10 Marks)

2. **Image Denoising:**

(a) Denoise the image `circuitboard.tif` corrupted by the impulsive noise by a mean and Median filter with the same spatial neighborhood size and compare the results.

(b) Use the bilateral filter to denoise the image `noisybook.png` corrupted by the Gaussian noise and compare the results with the Gaussian smoothing. The bilateral filter is computed for location (i, j) as

$$J(i, j) = \frac{1}{K} \sum_m \sum_n I(m, n)G(i - m, j - n)H(I(i, j) - I(m, n))$$

where K is computed appropriately to normalize the effect of weights. The summation is computed over a window of size $(2M + 1, 2M + 1)$ such that $m \in \{i - M, i - (M - 1), \ldots, i + (M - 1), i + M\}$ and $n \in \{j - M, j - (M - 1), \ldots, j + (M - 1), j + M\}$. You can use $M = 3$ (window size: 7×7) while computing. You will have to tune the other parameters to get a good result.

In order to understand the use of the spatial weights $G(.)$ and the luminance distance weights $H(.)$, we can try to visualize the weights on an image patch of size $(2M + 1, 2M + 1)$ around a specific pixel (a, b). For a better visualisation, use $M = 10$ (window size: 21×21) here.

- Take a patch centered at (a, b) in the image.
- Find the spatial weight map G_{map} as

$$G_{\text{map}}(m, n) = G(m, n)$$

where $m \in \{-M, -(M - 1), \ldots, M - 1, M\}$ and $n \in \{-M, -(M - 1), \ldots, M - 1, M\}$. You can shift the matrix such that the resultant map has non-negative indices.

- Similarly, find the luminance distance weight map H_{map} as

$$H_{\text{map}}(m,n) = H(I(a,b) - I(a+m, b+n))$$

For the above operations, use a patch centered at $(a,b) = (178, 260)$ in the image `noisybook.png` ($(179, 261)$ in MATLAB indexing) and use the parameters that worked well with the image while denoising (You may also make use of these maps to tune the parameters). Plot P, G_{map}, H_{map} and $G_{\text{map}} \odot H_{\text{map}}$. What do you observe from the maps? Can you relate it to how the bilateral filter works?

Note: The symbol $\odot$ refers to element-wise multiplication.

(10+20 Marks)

3. **DFT as matrix products (Bonus):** Find a way to compute DFT of an $M \times M$ image $f(m,n)$ as a product of matrices of the form $F = A^T f A$ (determine A) such that the resulting matrix follows the 2D-DFT expression

$$F(u,v) = \sum_{n=0}^{M-1} \sum_{m=0}^{M-1} f(m,n) e^{-j2\pi\left(\frac{um}{M} + \frac{vn}{M}\right)}$$

Use the derived matrix multiplication to find the DFT of the image `characters.tif` and compare it with the inbuilt function. Compare them by computing the mean squared absolute error between the results using matrix form and the inbuilt library. Also, plot the DFT outputs.

Plot $A^H A$. Think about why you are getting the result.

Note: A^H is the hermitian of A, defined as the element-wise complex conjugate of A^T. For $[A]_{ij} = a_{ij}$, the hermitian of A is defined as $\left[A^H\right]_{ij} = \overline{a_{ji}}$.

(20 Marks)