E9 241 Digital Image Processing
Assignment 03

Due Date: October 15, 2023 - 11:59 pm **Total Marks:** 75

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.
- Along with your code, also submit a PDF with all the **results** (images or numbers) and **inferences** (very important: you may not be explicitly asked to give inferences in each question. You should always include your inferences from what you have observed). Include answers to subjective questions, if any.
- Put all your files (code files and a report PDF) into a single zip file and submit the zip file. Name the zip file with your name.

1. **Radial Sinusoid and its Frequency Response:**
   (a) Generate a radial sinusoid image of size $M \times M$ using $x(u,v) = \cos(\frac{2\pi f_0}{M} \times D(u,v))$, where the distance from centre $D(u,v) = \sqrt{(u-M/2)^2 + (v-M/2)^2}$ with the side length $M = 500$ and frequency $f_0 = 50$.
   (b) Compute the DFT of this image. The DFT is computed with origin at $(0,0)$. To visualize the DFT response, cyclically shift this to the centre (using the library function `fftshift`). Visualize the magnitude of DFT.
   (c) Compute IDFT of the DFT response and compare the reconstructed image with the input.

   Try changing the frequency f_0 and comment on the results.

   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.

   (5+5+5=**15 Marks**)

2. **Frequency Domain Filtering:**
   (a) Filter the image `characters.tif` in the frequency domain using an ideal low pass filter (ILPF). The expression for the ILPF is

$$H(u,v;D_0) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

   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. What artefacts do you notice in the image obtained by computing the inverse DFT of the filtered image?

Note: The filter given above is centred in frequency domain. To use such centred filter, you will either need to shift the filter to $(0, 0)$ (by using `fftshift` in matlab or 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 inverse DFT of the filtered image by -1^{x+y}.

(b) Filter the image `characters.tif` in the frequency domain using the Gaussian low pass filter given by

$$H(u, v; D_0) = \exp(-D^2(u, v)/2D_0^2)$$

where all the terms are as explained in part (b). For $D_0 = 100$, compare the result with that of the ILPF.

(10+10=**20 Marks**)

3. **Image Deblurring:** Deblur the images `Blurred_LowNoise.png` (White Noise Standard Deviation (σ)=1) and `Blurred_HighNoise.png` ($\sigma = 10$) which have been blurred by the kernel `BlurKernel.mat` using

   (a) **Inverse filtering:** Simple inverse filtering may lead to amplification of noise (why?). To mitigate amplification of high frequency noise, set the inverse filter fft values to 0 wherever blur filter fft values are below a threshold t. Set $t = 0.1$.

   Note: The blur kernel given is centered at $(10, 25)$ in matlab convention and $(9, 24)$ in python convention. In other words, the maximum value 0.02 corresponds to $(0, 0)$ of the filter. So, before computing DFT, you need to shift the filter such that $(10, 25)$ moves to $(1, 1)$, $(10, 24)$ moves to $(1, N)$, $(9, 25)$ moves to $(M, 1)$ and $(9, 24)$ moves to (M, N) and so on in matlab convention. In python convention $(9, 24)$ moves to $(0, 0)$, $(9, 23)$ moves to $(0, N - 1)$, $(8, 24)$ moves to $(M - 1, 0)$ and $(8, 23)$ moves to $(M - 1, N - 1)$ and so on. M, N are the height and width of the blurred image respectively.

   (b) **Wiener filter:** You can assume the PSD of the white noise is equal to σ specified. For signal PSD, use power law

   $$S_f(u, v) = \frac{k}{\sqrt{u^2 + v^2}}.$$

   Use $k = 10^5$.

   Note: Recall that $S_f(u, v)$ is periodic and $S_f(-u, -v) = S_f(M - u, N - v)$ where M, N are height and width of the blurred image respectively .

Note: `BlurKernel.mat` is a matlab workspace variable file. You can load this file using the function `load('BlurKernel.mat')` in matlab provided that the file is stored in your working directory. For python, use `scipy.io.loadmat('BlurKernel.mat')`. (15+15=**30 Marks**)