Assignment #1

CSE 6367-002 **Student Name:**

Fall 2024 **Student ID:**

Due date: 09/24/24 11:59 pm Central Time

Submission Guidelines:

Submit through Canvas your source code in a single .ipynb file. The name of the .ipynb file should be YourStudentID.ipynb. (For example: 1001234567.ipynb) The images are available in the ./Images directory. Your TA will use the same directory name to grade your submission. You don't need to attach the images folder with your submission.

1. **(45%)** The Canny edge detector is an edge detection algorithm with multiple steps. The steps in the Canny edge detector are listed below:

   1. Smooth the image to remove the noise (Gaussian filter)
   2. Find the gradients of the image.
   3. Apply non-maximum suppression.
   4. Apply double threshold. Select edge pixels by hysteresis (suppress all edges that are weak and not connected to strong edges).

   Implement "Canny Edge Detector" from scratch using the steps above (for each step above, you can use library functions) and compare your result on `cameraman.jpeg` image with OpenCV implementation of `Canny Edge Detector`. (`https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html`)

2. **(30%)** Read the `cameraman.jpeg` image and apply the following operations:

   1. Perform average blur with kernel size 9X9 and 25X25. Display the original image with the filtered/processed images in a 1X3 grid.
   2. Perform Gaussian blur with kernel size 9X9, sigma (standard deviation of the Gaussian) of 2.0 and another Gaussian blur with kernel size 25X25, sigma 15. Display the original image with the filtered/processed images in a 1X3 grid.
   3. Perform median blur with kernel size 5X5 and 15X15. Display the original image with the filtered/processed images in a 1X3 grid.
   4. Resize the Gaussian blurred image with kernel size 25X25 and sigma 15 to 40X40 pixels, and resize the original image to 40X40 pixels. Display the original image with the filtered/processed images in a 1X3 grid.

   Note: Apply zero-padding to make the filtered image size same as original image.

3. **(45%)** Read the `bandnoise.png` image and display it along with its 2D Discrete Fourier Transform (via the FFT) in a 1x2 `matplotlib` figure. Note that the image has a specific type of noise. Design and apply a filter using the Image's Fourier Transform to remove the image's noise. Finally, display all three images (Noisy image, Fourier Transform image, and clean image) in a single `matplotlib` figure.

   Note: You can use any library to compute the FFT. It is helpful to use the properties of the Fourier transform.