CS5680/6680 – Fall Semester 2024
Assignment 3 – Filter Techniques for Image Enhancement, Edge Detection, and Noise Removal
Due: 11:59 p.m. Sunday, October 6, 2024
Total Points: 35 points

Problem I: Exercises on Low-pass and High-pass Filters in the Spatial Domain [Total: 15 points]
Note: Use zero padding to expand pixels beyond the border of an image. Your functions should accommodate any square filter of an odd dimension.

1. [5 points]
Implement an **AverageFiltering** function to perform an average filtering operation (i.e., convolution operation), as explained in class, on the input image. This function has two input parameters, where the first input parameter is the original grayscale image and the second input parameter is a square filter of an odd dimension (e.g., 3×3, 5×5, etc). Make sure that your function shows appropriate error messages when the filter does not possess **three properties** of the low-pass filter (i.e., all elements in the filter are positive, the sum of all elements is 1, and the elements are symmetric around the center). Note: Both input and output images of the **AverageFiltering** function should be an array with the same size and the same data type uint8. **You are NOT allowed to call any built-in filtering or convolution functions.**

Call **AverageFiltering** function to process the noisy image ***Circuit*** using a **standard** 5×5 averaging filter and a **weighted** 3×3 averaging filter, respectively. Display the original image and two processed images in Figure 1 with appropriate titles.

2. [5 points]
Implement a **MedianFiltering** function to perform a median filtering operation, as explained in class, on the input image. This function has two input parameters, where the first input parameter is the original grayscale image and the second input parameter is a square filter of an odd dimension. Make sure that your function shows appropriate error messages when any of the elements in the mask are not **positive integers**. Note: Both input and output images of the **MedianFiltering** function should be an array with the same size and the same data type uint8. **You are NOT allowed to call any built-in median filtering functions.**

Call **MedianFiltering** function to process the noisy image ***Circuit*** using a **standard** 3×3 median filter and a **weighted** 3×3 median filter **M** = [1 2 1 ; 2 4 2; 1 2 1], respectively. Display the original image and two processed images in Figure 2 with appropriate titles.

Note: The standard median filter is a special weighted median filter. Each value in a median filter indicates the number of copies of the corresponding masked value in the original image involved in the standard median filtering.

3. [5 points]
Use the formula **Enhanced Image = Original Image - Filtered Image** to get the final enhanced image using a **strong** 3×3 Laplacian mask to filter the image ***Moon*** by calling an **appropriate built-in function** (Hint: This formula indicates that one of the two strong Laplacian masks should be used). Use **imshow** to display the original image and the enhanced image side-by-side in Figure 3 with appropriate titles (Refer to Figure 3.40 in the textbook or the same figure on slide 57 of class notes).

Problem II: Exercises on Edge Detectors in the Spatial Domain [Total: 8 points]

Edge detection is a fundamental step in computer vision and image processing. Several algorithms were developed for this purpose. Canny and Sobel techniques have the highest efficiency among the available edge detection algorithms. This document compares these two edge detectors and their step-by-step implementation.

The Sobel edge detector requires one threshold to determine edge pixels. The Canny edge detector requires one upper threshold and one lower threshold to determine non-edge pixels, weak pixels, and strong pixels.

Search online for the best strategies to determine the threshold for the Sobel edge detector and the two thresholds for the Canny edge detector. **Summarize your findings and output them to the console.**

Implement a **FindEdgeInfo** function to apply the Sobel filters to locate important edges using an automatically decided threshold and compute the edge histogram of an input image. This function has two input parameters, where the first input parameter is the original grayscale image and the second input parameter is the number of bins. It returns two outputs, where the first output contains important edges in the input image and the second output contains the counts of the orientation (angles) of the edge in each bin.

The orientation of the edge at each pixel location can be computed by: $\theta = \arctan\left(G_x / G_y\right)$, where

- *Gx* is the gradient component produced by the horizontal edge detector via the convolution operation;
- *Gy* is the gradient component produced by the vertical edge detector via the convolution operation;
- / represents the elementwise division.

Refer to slide 63 of the class notes. *Gx* is the filter at left and *Gy* is the filter at right. **You are NOT allowed to call any built-in edge functions and histogram functions.**

Call **FindEdgeInfo** function to compute a 30-bin edge histogram (e.g., dividing the full 360-degree range of edge orientations into 30 equal intervals and counting the number of angles that fall within each interval) of the image ***Rice***. Display the original image, the image with important edges, and the 30-bin edge histogram in Figure 4 with appropriate titles.

Problem III: A Practical Problem [12 points]

Use the techniques explained in class to get rid of the streaks (stripes) in the image "**Text.gif**". Please write a function **RemoveStripes** with the original grayscale image as the input and the cleaned grayscale image as the output. Display the original grayscale image, some important intermediate images or plots, and the cleaned image in a few figures with appropriate titles. **[10 points]**

Call **RemoveStripes** function to get rid of the streaks in the image **Text1.gif**. **[2 points]**