

Image Processing – HW3 (05/18/2023)

Instructions – Follow these carefully:

1. Please upload your work as a zip file attachment to Moodle. The zip file must have the source code and a PDF report where you explain and display the outputs for each problem.
2. You can use either Python or Matlab to do the work.
3. Please feel free to read related materials available in the official Matlab/Python documentation.
4. The due date is 6/3 before 11:59 pm.

Assignment:

1. Remove salt-and-pepper noise
 - a. (10%) Please write a program to add 10%, 30%, 50%, 70%, and 90% salt-and-pepper noise to 'baboon.bmp' and 'peppers.bmp.'
 - b. (20%) Please Write a program that performs two-dimensional 5x5 mean filtering to clean up the 10%~90% noisy images you generated. As the table below shows, you need to exclude the noise pixels before applying **mean** filtering and report PSNR before and after denoising.

PSNR	Before denoising					After denoising				
	10	30	50	70	90	10	30	50	70	90
Baboon										
Peppers										


- c. (20%) Following the previous question, please use two-dimensional 5x5 Gaussian filtering (zero-mean Gaussian distribution with a standard deviation of 2) and report the PSNR results.

PSNR	Before denoising					After denoising				
	10	30	50	70	90	10	30	50	70	90
Baboon										
Peppers										

- d. (10%) Following the previous questions, please implement the “Modified Decision-based Unsymmetrical Trimmed Median Filter” with an adaptive kernel size. It means it does not have ‘Case 1,’ where all the pixels in the sliding window are noisy. For each noise pixel “p,” the adaptive kernel size needs to be the same as the smallest size of the sliding window centered at “p” with at least one non-noise pixel. For example, if a 3x3 window does not have a non-noise pixel, you need to increase the window size to 5x5 and check again until the

window contains at least one non-noisy pixel. Please report the PSNR results like the tables in Q1.b and Q1.c.

2. Edge Detection

- a. (15%) Please implement Sobel filtering to find the edge map for 'pepper.bmp' and 'pepper[0.04].bmp', whose results should look like  <https://www.mathworks.com/discovery/edge-detection.html> (please implement the Sobel filter by yourself)
- b. (15%) Following the previous question, please apply Gaussian filtering (zero-mean and standard deviation of 1) to smooth the images ('pepper.bmp' and 'pepper[0.04].bmp') first, apply the Laplacian operator to the images, and report the results.