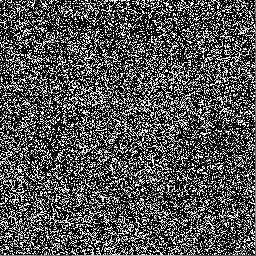
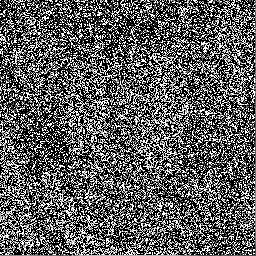
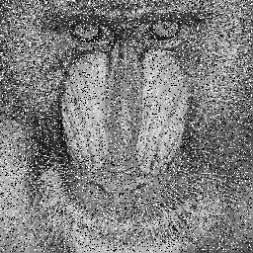
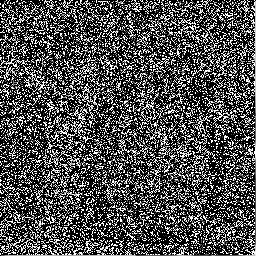
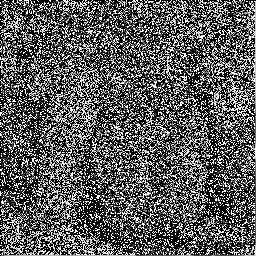
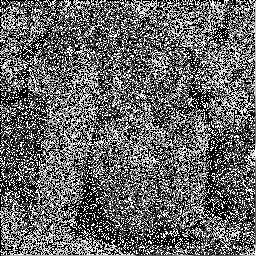
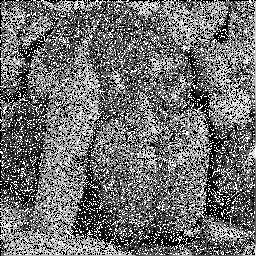
1.

a) Please write a program to add 10%, 30%, 50%, 70%, and 90% salt-and-pepper noise to ‘baboon.bmp’ and ‘peppers.bmp.’

10 30 50 70 90





b) 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 | 15.7 | 10.8 | 8.6 | 7.1 | 6.1 | 14.3 | 10.4 | 8.3 | 7.0 | 6.9 |
| Peppers | 15.4 | 10.6 | 8.3 | 6.9 | 5.8 | 14.7 | 10.3 | 8.1 | 6.7 | 5.8 |

c) 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.

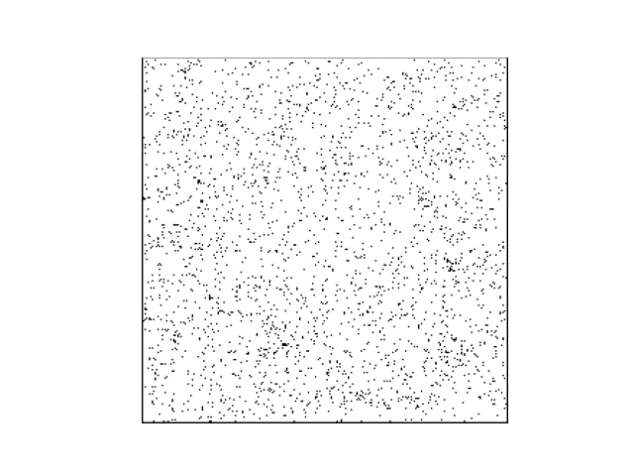
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PSNR | Before Denoising | | | | | After Denoising | | | | |
| 10 | 30 | 50 | 70 | 90 | 10 | 30 | 50 | 70 | 90 |
| Baboon | 15.7 | 10.8 | 8.6 | 7.1 | 6.1 | 19.9 | 18.9 | 17.6 | 16.5 | 15.5 |
| Peppers | 15.4 | 10.6 | 8.3 | 6.9 | 5.8 | 22.6 | 20.0 | 17.7 | 15.7 | 14.2 |

d) Following the previous questions, please implement the “Modified Decision-based Unsymmetrical Trimmed Median Filter” with an adaptive kernel size.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PSNR | Before Denoising | | | | | After Denoising | | | | |
| 10 | 30 | 50 | 70 | 90 | 10 | 30 | 50 | 70 | 90 |
| Baboon | 15.7 | 10.8 | 8.6 | 7.1 | 6.1 | 20.7 | 19.6 | 18.4 | 17.2 | 13.0 |
| Peppers | 15.4 | 10.6 | 8.3 | 6.9 | 5.8 | 28.5 | 25.2 | 22.1 | 19.9 | 13.1 |

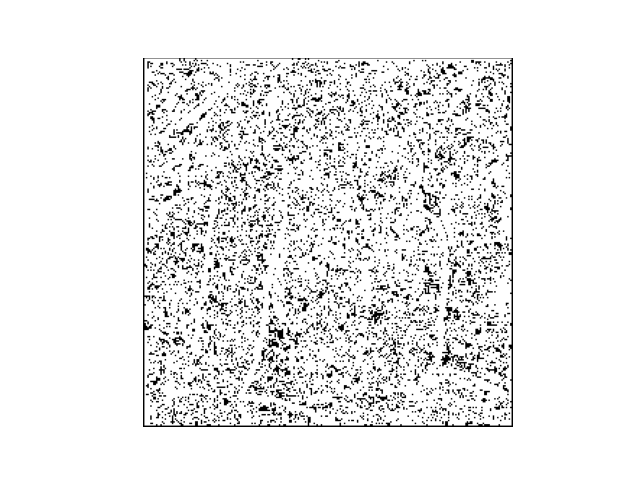
* 1. 2.

a) Please implement Sobel filtering to find the edge map for **‘pepper.bmp’** and **‘pepper\_0.04.bmp’**,

* 1. 
  2. (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 Lapcian operator to the images, and report the results.

Sober:





Lapcian



