Image Processing – HW3 (05/18/2023)

111753127 資碩工一 蘇冠華

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.'

    Sol:

    Step1.先定義一個專門處理"salt-and-pepper noise"的function，

    function內容:i. 根據噪聲百分比計算noise像素的數量

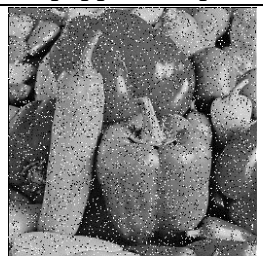
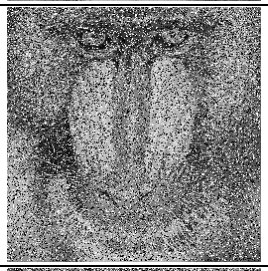
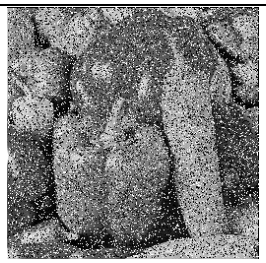
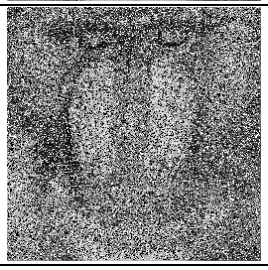
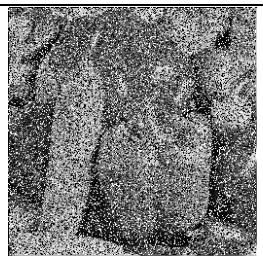
    ii. 為salt-and-pepper noise生成隨機坐標

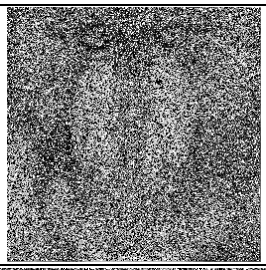
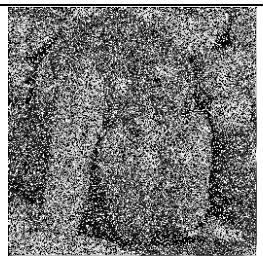
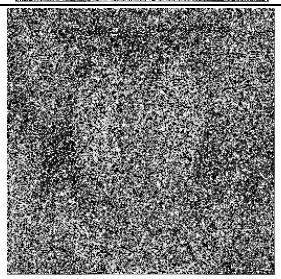
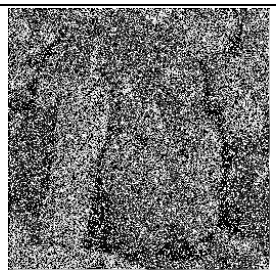
    iii.將上面所生成的salt-and-pepper noise加到圖片中

    Step2. 定義一個內容為[10,30,50,70,90]的Array，目的是讓程式能根據10%, 30%, 50%, 70%, and 90%的salt-and-pepper noise輸出圖片

    Step3.設立一個for迴圈依照Step2所定義的Array 執行Step1所定義的function

    Step4.將輸出的圖片儲存至指定路徑的資料夾，結果如下：

	baboon.bmp	peppers.bmp
10%		
30%		
50%		

70%		
90%		

b. (20%) Please Write a program that performs two-dimensional 5x5 mean filteringto 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.

Sol:

PSNR	Before denoising					After denoising				
	10	30	50	70	90	10	30	50	70	90
Baboon	38.276	33.872	32.095	31.040	30.335	28.614	28.267	28.117	27.997	27.950
Peppers	37.964	33.619	31.837	30.750	30.056	29.718	28.699	28.346	28.132	28.054

Step1. 定義一個計算圖片mean filtering的function

Step2.定義一個計算圖片PSNR的function，它使用均方誤差（MSE）來計算加噪圖像和去噪圖像相對於原始圖像的差異，然後將其轉換為PSNR值並回傳

Step3. 設立一個for迴圈執行Step1及Step2所定義的function

Step4.將輸出的圖片儲存至指定路徑的資料夾，結果如下：

```
Noisy Baboon (10% noise):
PSNR (Noisy): 38.27560093659953
PSNR (Denoised): 28.61406028439998

Noisy Peppers (10% noise):
PSNR (Noisy): 37.963730044145635
PSNR (Denoised): 29.717687703069792

Noisy Baboon (30% noise):
PSNR (Noisy): 33.87209716428415
PSNR (Denoised): 28.266954765010077

Noisy Peppers (30% noise):
PSNR (Noisy): 33.618611569746335
PSNR (Denoised): 28.699548386298908

Noisy Baboon (50% noise):
PSNR (Noisy): 32.09541797761618
PSNR (Denoised): 28.1171473704976

Noisy Peppers (50% noise):
PSNR (Noisy): 31.837228148826497
PSNR (Denoised): 28.34567696835105
```

```
Noisy Baboon (70% noise):
PSNR (Noisy): 31.040232605938264
PSNR (Denoised): 27.997125437297253

Noisy Peppers (70% noise):
PSNR (Noisy): 30.750065675399554
PSNR (Denoised): 28.13234130030015

Noisy Baboon (90% noise):
PSNR (Noisy): 30.335397057069564
PSNR (Denoised): 27.950095097066043

Noisy Peppers (90% noise):
PSNR (Noisy): 30.05636817155995
PSNR (Denoised): 28.053586091656143
```

c. (20%) Following the previous question, please use two-dimensional 5x5 Gaussian filtering (zero-mean Gaussian distribution with a standard deviation of2) and report the PSNR results.

Sol:

PSNR	Before denoising					After denoising				
	10	30	50	70	90	10	30	50	70	90
Baboon	38.276	33.872	32.095	31.040	30.335	28.655	28.304	28.158	28.018	27.927
Peppers	37.964	33.619	31.837	30.750	30.056	29.773	28.711	28.328	28.121	28.040

Step1. 定義一個計算圖片Gaussian filtering的function

Step2.定義一個計算圖片PSNR的function，它使用均方誤差（MSE）來計算加噪圖像和去噪圖像相對於原始圖像的差異，然後將其轉換為PSNR值並回傳

Step3. 設立一個for迴圈執行Step1及Step2所定義的function

Step4.將輸出的圖片儲存至指定路徑的資料夾，結果如下：

```
Noisy Baboon (10% noise):
PSNR (Noisy): 38.27560093659953
PSNR (Denoised): 28.65506915095548

Noisy Peppers (10% noise):
PSNR (Noisy): 37.963730044145635
PSNR (Denoised): 29.77297894463486

Noisy Baboon (30% noise):
PSNR (Noisy): 33.87209716428415
PSNR (Denoised): 28.30374530112918

Noisy Peppers (30% noise):
PSNR (Noisy): 33.618611569746335
PSNR (Denoised): 28.711351724015053

Noisy Baboon (50% noise):
PSNR (Noisy): 32.09541797761618
PSNR (Denoised): 28.1584503262296

Noisy Peppers (50% noise):
PSNR (Noisy): 31.837228148826497
PSNR (Denoised): 28.32804508274059
```

```
Noisy Baboon (70% noise):
PSNR (Noisy): 31.040232605938264
PSNR (Denoised): 28.01778118005099

Noisy Peppers (70% noise):
PSNR (Noisy): 30.750065675399554
PSNR (Denoised): 28.1211035251642

Noisy Baboon (90% noise):
PSNR (Noisy): 30.335397057069564
PSNR (Denoised): 27.92766935515441

Noisy Peppers (90% noise):
PSNR (Noisy): 30.05636817155995
PSNR (Denoised): 28.04018961398082
```

d. (10%) Following the previous questions, please implement the “Modified Decision-based Unsymmetrical Trimmed Median Filter” with an adaptive kernelsize. 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 bethe same as the smallest size of the sliding window centered at “p” with at leastone 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 likethe tables in Q1.b and Q1.c.

Sol:

PSNR	Before denoising					After denoising				
	10	30	50	70	90	10	30	50	70	90
Baboon	38.276	33.872	32.095	31.040	30.335	39.424	34.927	33.007	31.742	30.925
Peppers	37.964	33.619	31.837	30.750	30.056	42.954	38.626	36.114	34.178	32.718

Step1.定義一個計算圖片PSNR的function，它使用均方誤差（MSE）來計算加噪圖像和去噪圖像相對於原始圖像的差異，然後將其轉換為PSNR值並回傳

Step2. 定義一個計算圖像kernel size的function，它會根據指定的圖像和像素位置，找到最小的有效kernel size。

說明: i. kernel size的定義是指其範圍內不全為0或全為255。

ii. function包括三個參數，分別是加噪圖像、像素位置i和j。

iii.它以指定位置為中心，從3開始遞增kernel size，並在每個大小上檢查核內的像素值是否都不為0或255。一旦找到符合條件的kernel size，就回傳該kernel size。

Step3.定義一個“Modified Decision-based Unsymmetrical Trimmed Median Filter” 的算法function對圖像進行去噪。

說明: i. 它是一種非線性濾波器，用於平滑圖像並去除噪點。

ii. function接受一個圖像作為參數。

iii. 首先將圖像轉換為NumPy數組，然後創建一個與原始圖像大小相同的數組來保存濾波後的結果。

iv. 接下來使用Step2.的kernel size function，根據該像素周圍的有效kernel size計算修剪中值並將值給濾波後的結果。如果該像素的值不為0或255，就直接複製到濾波後的結果中。

v. 將濾波後的數組轉換回圖像並返回。

Step4. 設立一個for迴圈執行Step1、及Step3所定義的function

Step5.將輸出的圖片儲存至指定路徑的資料夾，結果如下：

```
Noisy Baboon (10% noise):
PSNR (Noisy): 38.27560093659953
PSNR (Denoised): 39.42424781274026

Noisy Peppers (10% noise):
PSNR (Noisy): 37.963730044145635
PSNR (Denoised): 42.95426347055386

Noisy Baboon (30% noise):
PSNR (Noisy): 33.87209716428415
PSNR (Denoised): 34.92718012135641

Noisy Peppers (30% noise):
PSNR (Noisy): 33.618611569746335
PSNR (Denoised): 38.625766925959525

Noisy Baboon (50% noise):
PSNR (Noisy): 32.09541797761618
PSNR (Denoised): 33.00705670726324

Noisy Peppers (50% noise):
PSNR (Noisy): 31.837228148826497
PSNR (Denoised): 36.11359268995318
```

```
Noisy Baboon (70% noise):
PSNR (Noisy): 31.040232605938264
PSNR (Denoised): 31.74168260505124

Noisy Peppers (70% noise):
PSNR (Noisy): 30.750065675399554
PSNR (Denoised): 34.17757061208833

Noisy Baboon (90% noise):
PSNR (Noisy): 30.335397057069564
PSNR (Denoised): 30.925468356479662

Noisy Peppers (90% noise):
PSNR (Noisy): 30.05636817155995
PSNR (Denoised): 32.71834451375418
```

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)

    Sol:

    Step1.定義一個“Sobel filtering” 的算法function來進行邊緣偵測

    說明: i. function接受一個圖像作為參數。

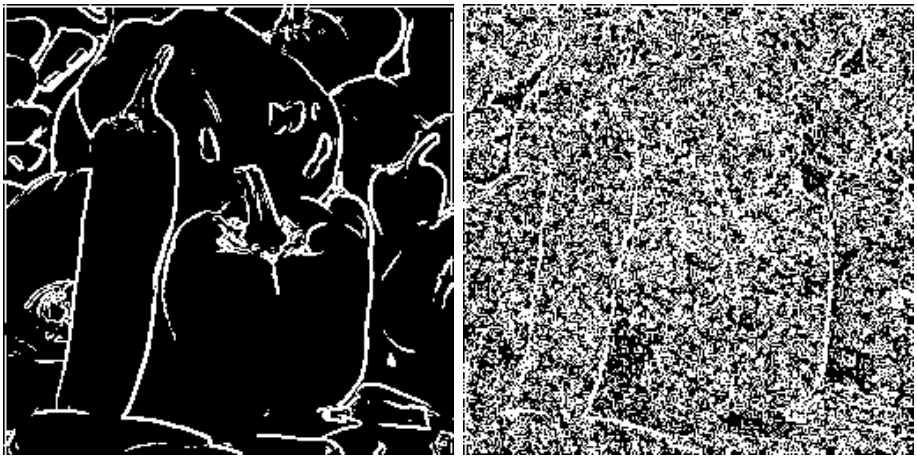
    ii. 將輸入圖像轉換為灰階圖。

    iii. 使用cv2的Sobel濾波器計算圖像x軸、y軸的梯度幅值

    iv. 進行歸一化 : cv2.normalize()

    v.對梯度幅值進行二值化及閾值調整

    Step2. 將輸出的圖片儲存至指定路徑的資料夾，結果如下：



‘pepper.bmp’ **‘pepper_0.04.bmp’**

b. (15%) Following the previous question, please apply Gaussian filtering (zero- mean and standard deviation of 1) to smooth the images first, apply the Lapcianoperator to the images, and report the results.

Sol:

Step1.定義一個“Gaussian filtering” 的算法function來進行邊緣偵測

說明: i. function接受一個圖像作為參數。

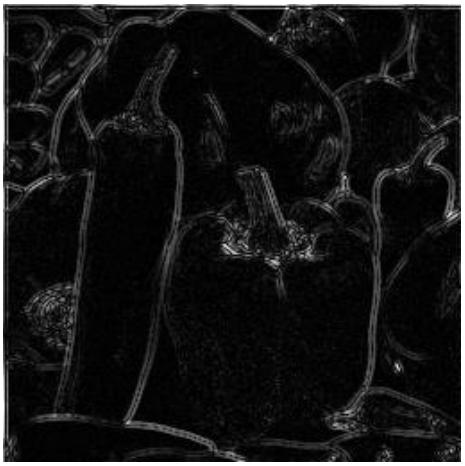
ii. 將輸入圖像轉換為灰階圖。

iii. 使用cv2.GaussianBlur()，高斯平滑處理來減少noise

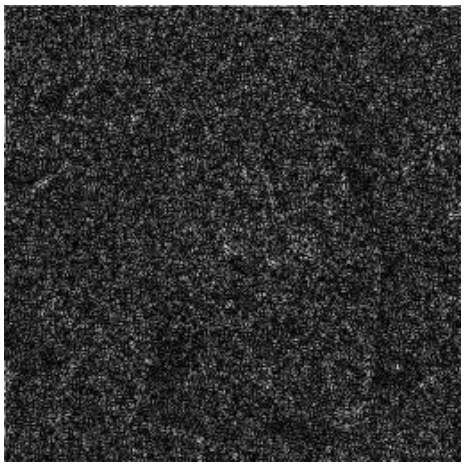
iv. 使用cv2.Laplacian()輸入平滑後的灰階圖，並使用”cv2.CV_64F”的格式作為輸出圖像的深度圖。

v. 進行歸一化 : cv2.normalize()

Step2.將輸出的圖片儲存至指定路徑的資料夾，結果如下：



‘pepper.bmp’



‘pepper_0.04.bmp’