Assignment II

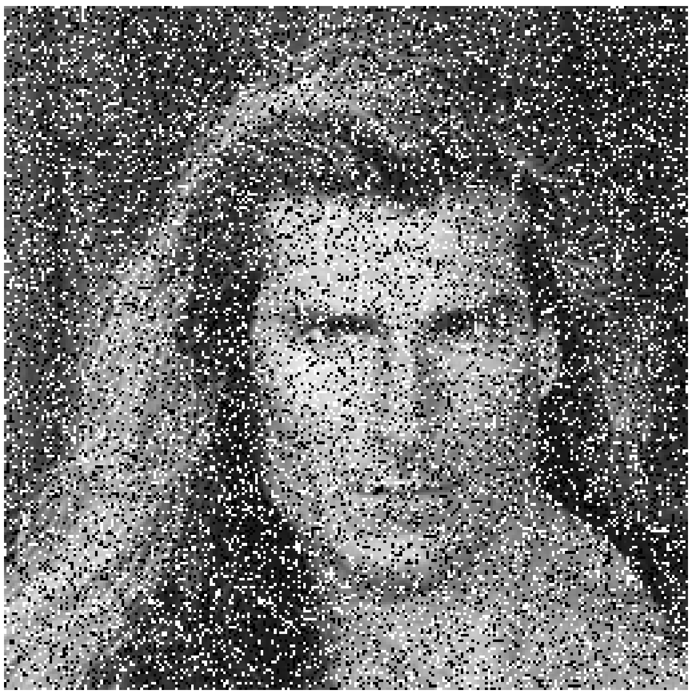
s1116007 資工碩一 蔡煒俊

1 Statement of Problem

In the previous assignment, we discussed an operation from a dark image to a bright one. Today, in this assignment, we will discuss another problem that often happens in *data transmission*, *failure in the memory cell* or *analogue-to-digital converter errors*. This problem usually comes up with **salt-and-pepper noise**, that is, it takes the form of randomly occurring white and black pixels, which can significantly deteriorate the quality of an image.

1.1 Salt-and-Pepper Noise

Why is this kind of noise called salt-and-pepper? Because the image with this kind of noise will have a lot of white (like salt) and black (like pepper) spots.



2 Related Technique

2.1 Median

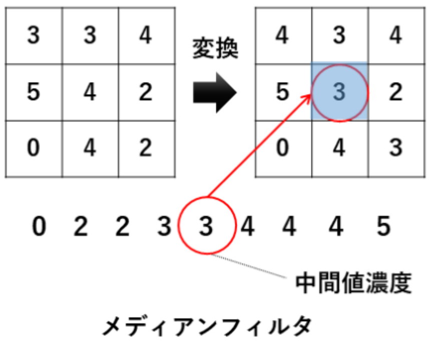
What is the median? A Median is a number in the middle of a sorted number list. The total elements in the sorted number list should be odd and must greater or equal to 3. For instance, a number list $S = 10, 9, 6, 8, 7$. After the sorting operation, we got the number list $S = \{6, 7, 8, 9, 10\}$ and found the median, the number 8 (at position 3).

2.2 Filter

Filtering is a technique for **modifying** or **enhancing an image**. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement. In this assignment, we are focus on image smoothing.

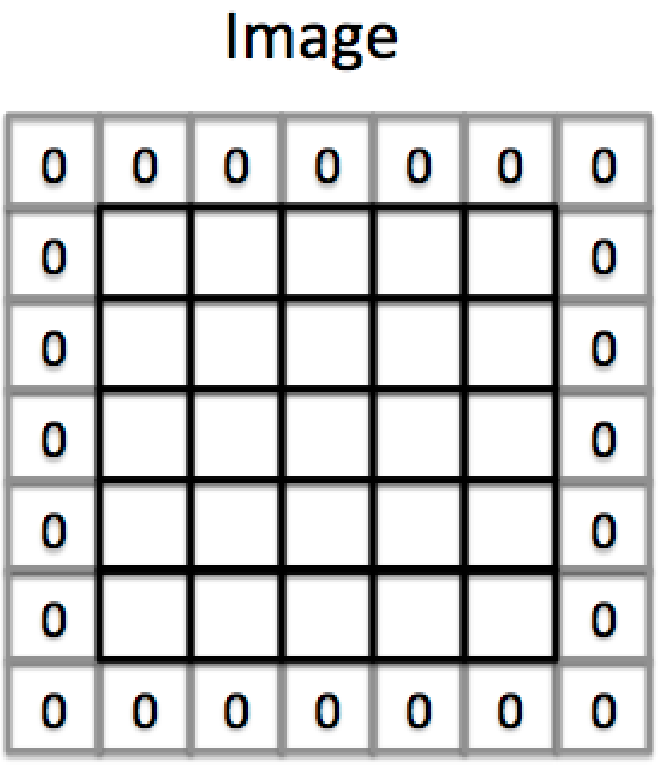
2.3 Median Filter

Deal with salt-and-pepper noise, a technique called **median filter** to handle it. The median filter is a non-linear digital filtering, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it **preserves edges while removing noise**, also having applications in signal processing.



2.4 Zero-Padding of An Image

It means **to enlarge the image by adding rectangular strips of zeros outside the rectangular edge of the image**, so that you have a new larger rectangular image with a black frame around it.



3 Experiments

3.1 Input Image

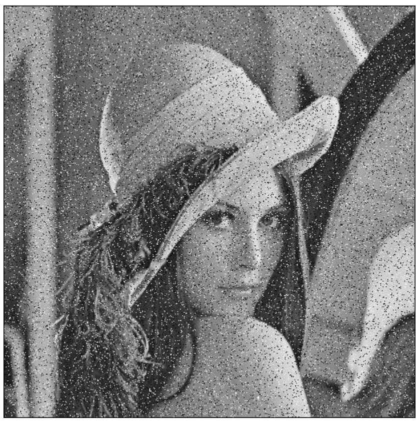


Figure 1: Salt-and-Pepper Noise with Zero Padding

3.2 Output Image

3.2.1 2-pass 3 x 3 Median Filter

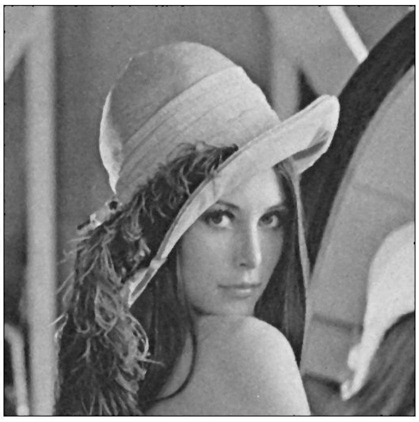


Figure 2: 2-pass 3 by 3 median filter

3.2.2 1-pass 5 x 5 Median Filter

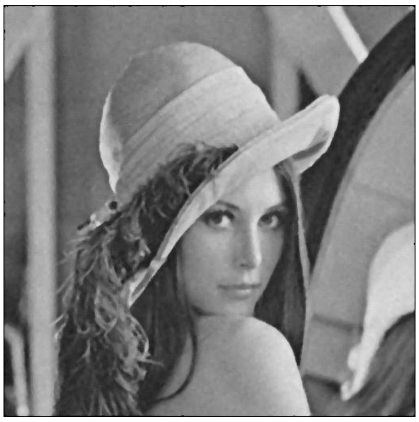
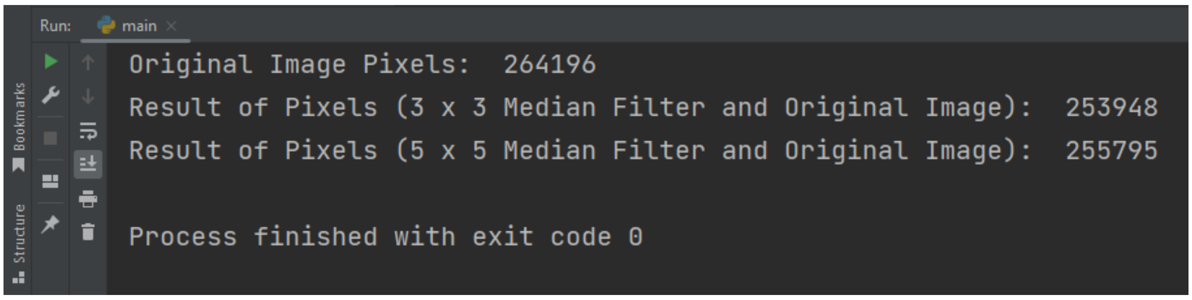


Figure 3: 1-pass 5 by 5 median filter

3.3 Results: The Number of Noise Removed

1. Original Image: 514 x 514 pixels = **264196** pixels
2. The Number Pixels of Noise Removed (2-pass 3 by 3): **253948** pixels
3. The Number Pixels of Noise Removed (1-pass 5 by 5): **255795** pixels



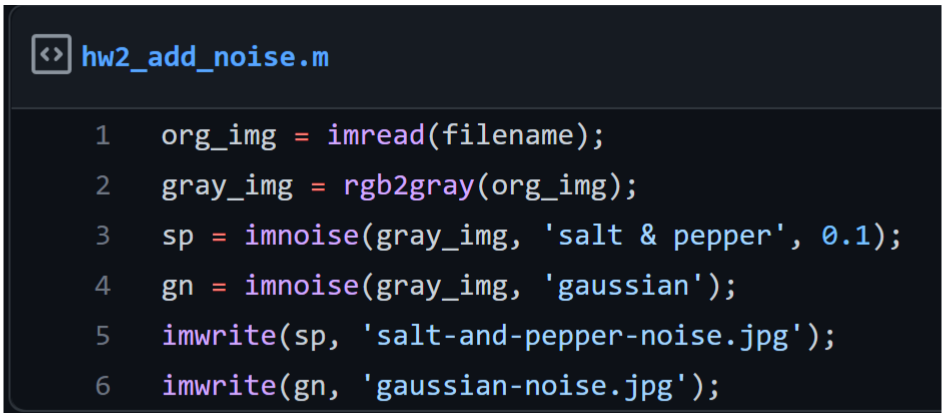
```
Original Image Pixels:  264196
Result of Pixels (3 x 3 Median Filter and Original Image):  253948
Result of Pixels (5 x 5 Median Filter and Original Image):  255795

Process finished with exit code 0
```

Figure 4: The Number Pixels of Noise Removed (2-pass 3 by 3 and 1-pass 5 by 5)

3.4 Code

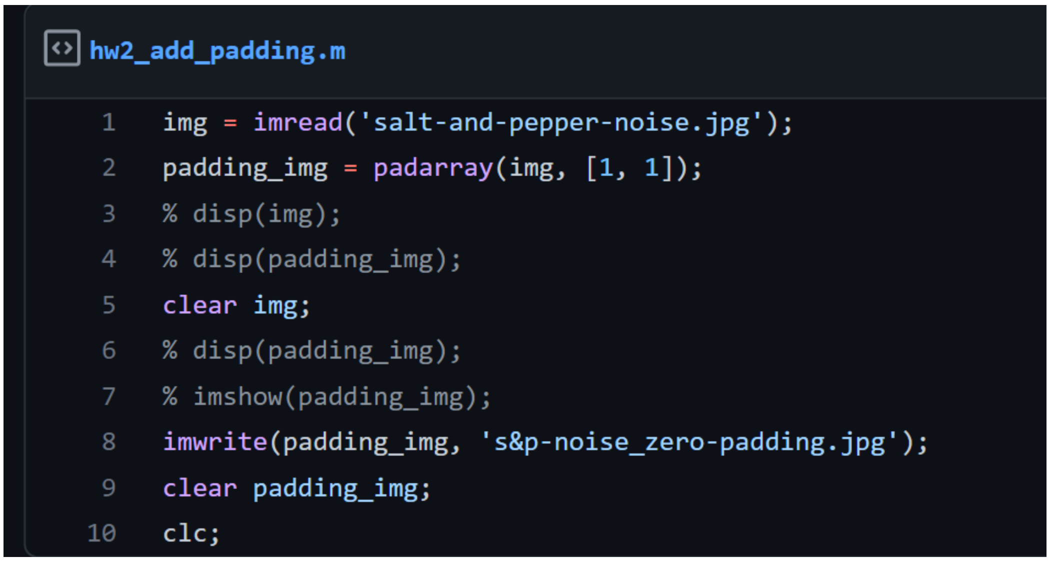
3.4.1 Salt-and-Pepper Noise Adding


hw2_add_noise.m

```matlab
org_img = imread(filename);
gray_img = rgb2gray(org_img);
sp = imnoise(gray_img, 'salt & pepper', 0.1);
gn = imnoise(gray_img, 'gaussian');
imwrite(sp, 'salt-and-pepper-noise.jpg');
imwrite(gn, 'gaussian-noise.jpg');
```

Figure 5: Matlab Code: Add 10% Salt-and-Pepper Noise to Image

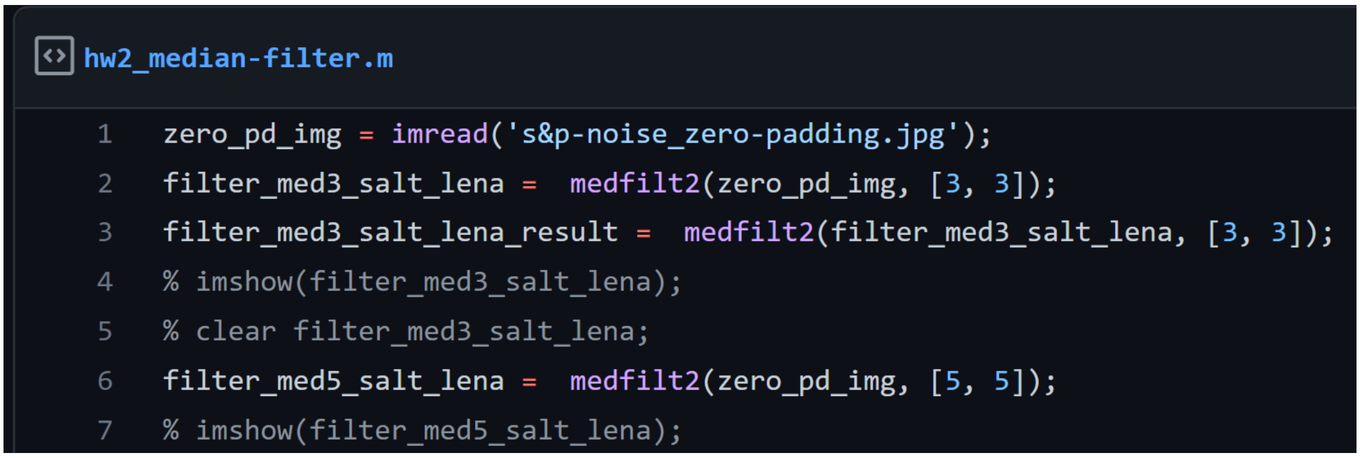
3.4.2 Zero-Padding Adding


hw2_add_padding.m

```matlab
img = imread('salt-and-pepper-noise.jpg');
padding_img = padarray(img, [1, 1]);
% disp(img);
% disp(padding_img);
clear img;
% disp(padding_img);
% imshow(padding_img);
imwrite(padding_img, 's&p-noise_zero-padding.jpg');
clear padding_img;
clc;
```

Figure 6: Matlab Code: Zero-Padding Adding

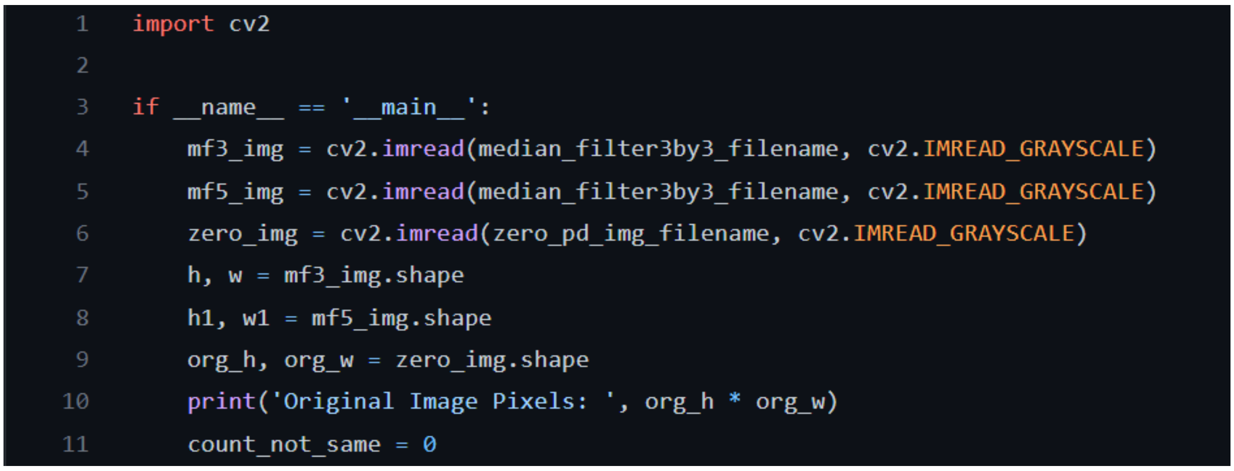
3.4.3 Median Filter Operation


hw2_median-filter.m

```
zero_pd_img = imread('s&p-noise_zero-padding.jpg');
filter_med3_salt_lena =  medfilt2(zero_pd_img, [3, 3]);
filter_med3_salt_lena_result =  medfilt2(filter_med3_salt_lena, [3, 3]);
% imshow(filter_med3_salt_lena);
% clear filter_med3_salt_lena;
filter_med5_salt_lena =  medfilt2(zero_pd_img, [5, 5]);
% imshow(filter_med5_salt_lena);
```

Figure 7: Median Filter: 2-pass 3 x 3 and 1-pass 5 x 5

3.4.4 Calculation of The Number of Noise Removed Part I

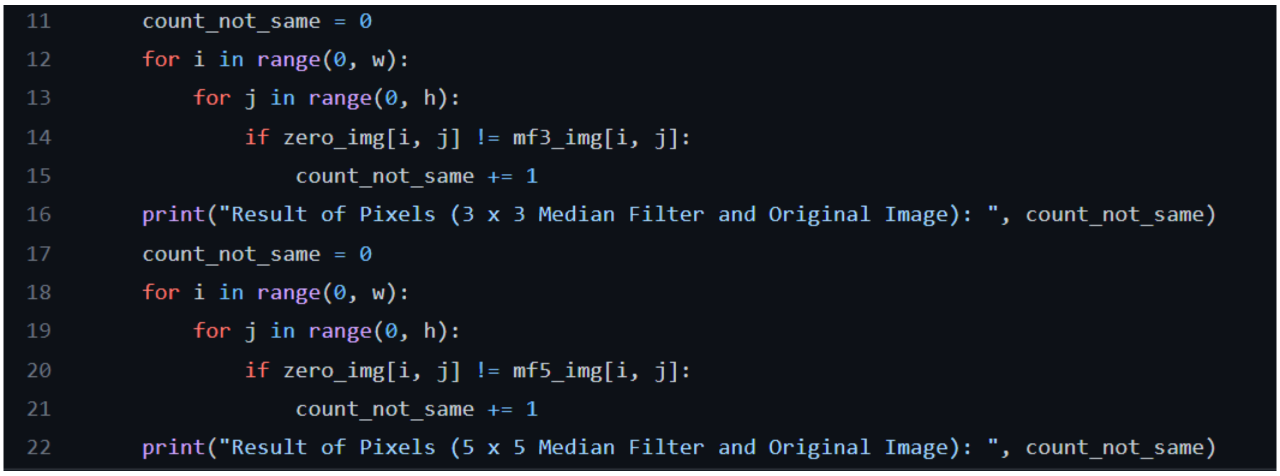


```python
import cv2

if __name__ == '__main__':
    mf3_img = cv2.imread(median_filter3by3_filename, cv2.IMREAD_GRAYSCALE)
    mf5_img = cv2.imread(median_filter3by3_filename, cv2.IMREAD_GRAYSCALE)
    zero_img = cv2.imread(zero_pd_img_filename, cv2.IMREAD_GRAYSCALE)
    h, w = mf3_img.shape
    h1, w1 = mf5_img.shape
    org_h, org_w = zero_img.shape
    print('Original Image Pixels: ', org_h * org_w)
    count_not_same = 0
```

Figure 8: Median Filter: 2-pass 3 x 3 and 1-pass 5 x 5

3.4.5 Calculation of The Number of Noise Removed Part II



```python
    count_not_same = 0
    for i in range(0, w):
        for j in range(0, h):
            if zero_img[i, j] != mf3_img[i, j]:
                count_not_same += 1
    print("Result of Pixels (3 x 3 Median Filter and Original Image): ", count_not_same)
    count_not_same = 0
    for i in range(0, w):
        for j in range(0, h):
            if zero_img[i, j] != mf5_img[i, j]:
                count_not_same += 1
    print("Result of Pixels (5 x 5 Median Filter and Original Image): ", count_not_same)
```

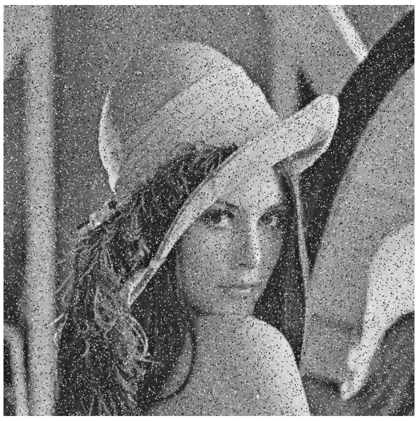
Figure 9: Median Filter: 2-pass 3 x 3 and 1-pass 5 x 5

4 Code Explanation

4.1 Salt-and-Pepper Noise Adding

In **3.4.1 section**, we propose an operation method for adding salt-and-pepper noise to the image. In **lines 1-2**, Lena image is loaded and converted into a grey scale; In **line 3**, using the Matlab function, imnoise, adds salt-and-pepper noise. In this function, **imnoise**, parameters are **imnoise(img_file, noise_name, adding_percent)** Last, we write the noise image.

Original Image: Lena



Convert into gray-scale and add salt-and-pepper noise

4.2 Zero-padding Adding

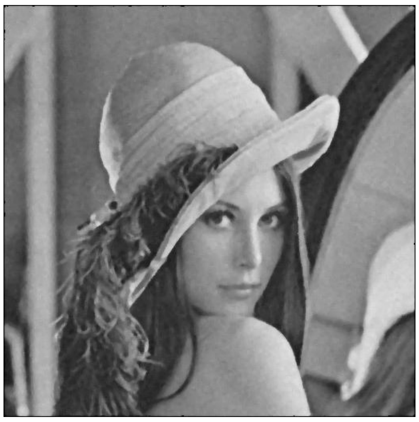
Section 3.4.2 is a zero-padding operation. That is, adding rectangular strips of zeros outside the rectangular edge of the image so that you ***have a black frame around the image***. For this purpose, using the Matlab function **padarray**, padarray parameters are padarray(image-arrat, padsize), where pads image-array with an amount of padding in each dimension specified by padsize. The padarray function pads numeric or logical images with the **value 0** and categorical images with the category <**undefined**>. By default, ***paddarray adds padding before the first element and after the last element of each dimension***.

Zero-padding, please zoom in the image to see the zero-padding result.

4.3 Median Filtering

Section 3.4.3 doing the **median filter**. In **lines 2-3**, do a **2-pass 3 x 3 median filter operation**. The first pass is in line 2, and the second is in line 3. Next, **line 6** of the program operates as a **1-pass 5 x 5 median filter**. The 2-D median filter function in Matlab is called medfilt2. It has two parameters. One is the *image array*, and the other is *M by N filter array.*

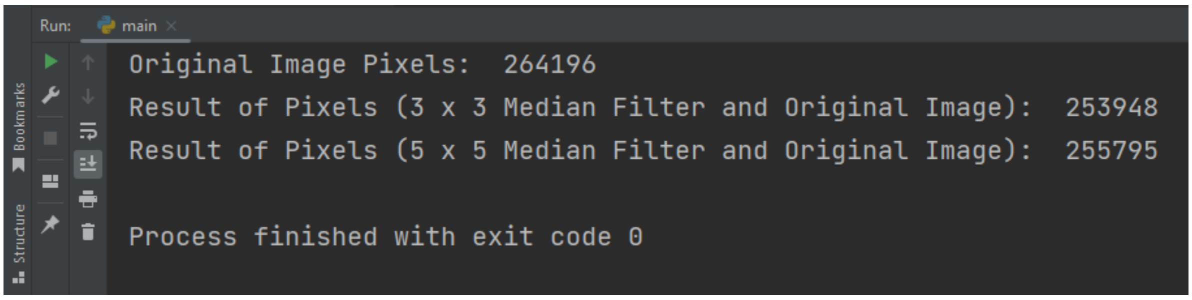
2-Pass 3 by 3 Median Filter



1-Pass 5 by 5 Median Filter

4.4 Output Results (i.e., the number of noise removed)

In the last part (**section 3.4.4** and **section 3.4.5**) of our code, we use python to calculate the output results (i.e., the number of noise removed). That is the **total amounts of pixels that have been updated or changed** In **section 3.4.4**, **lines 4-10** in the python program load in the *original salt-and-pepper noise image*, *3 x 3 2-pass median filtered image*, and *5 x 5 1-pass median filtered image*; calculated its height and width and the noise image pixels. Next, declare a variable called ***count_not_same***, which is the **statistics of the pixel value of the change after the median filter operation**. The last section (**section 3.4.5**) has a for loop with an if condition for **adding the variable**, count_not_same, **where the pixel value has been changed or updated**.



```
Original Image Pixels:  264196
Result of Pixels (3 x 3 Median Filter and Original Image):  253948
Result of Pixels (5 x 5 Median Filter and Original Image):  255795

Process finished with exit code 0
```

Figure 10: The Number Pixels of Noise Removed (2-pass 3 by 3 and 1-pass 5 by 5)

5 Image Source

The image used in this assignment is called ***Lena***. It is the most classic image in the digital image processing field. That is this image is often used in digital image processing experiments.

6 Assignment Description and Lecturer's Hint

Input a grayscale image and add 10% salt-and-pepper noise. Compare the effect of applying a 2-pass 3X3 median filter and a 1-pass 5X5 median filter. (Note: padding is required.) Hint: The number of noise points that are removed can be used to compare the effect of applying the two methods The content of the report includes:

1. The program code and program description
2. Input image
3. Output results (i.e., the number of noise removed) of the two filtering modes.

7 LaTex Code

`https://www.overleaf.com/read/zmjvtcxrdkkt`

8 Project Code - GitHub

8.1 Matlab

8.1.1 Add Salt-Pepper Noise

`https://gist.github.com/KevinTsaiCodes/c3dcd91c0d19680f6679ca13987d24b7#file-hw2_add_noise-m`

8.1.2 Zero-Padding

`https://gist.github.com/KevinTsaiCodes/c3dcd91c0d19680f6679ca13987d24b7#file-hw2_add_padding-m`

8.1.3 Median Filter

https://gist.github.com/KevinTsaiCodes/c3dcd91c0d19680f6679ca13987d24b7#file-hw2_median-filter-m

8.2 Python

8.2.1 Calculation Result

https://gist.github.com/KevinTsaiCodes/c3dcd91c0d19680f6679ca13987d24b7#file-hw2_calculate_different_pixels-py

Finished Date: November 25, 2022
Author: Wei-Chun **Kevin** Tsai