hw3	4109037012 資工四 李曉旻

Coding part :

讀取一張圖像，將其轉換為灰度圖像

```
image = cv2.imread('/content/IMG_6.JPG')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray_image.shape
```

將交錯顏色的文字"NCHU CSE"添加到圖像上，先設定文字的字體、大小、粗細及初始位置(左上角)，再用迴圈，依次在圖像上繪製每個字母，並使用黑白交替顏色。最後，顯示並返回修改後的圖像如下。


```
def add_interlaced_text(image):
    # image = cv2.imread(image_path)
    if image is None:
        print("圖片讀取失敗，請檢查圖片")
        return None

    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 1
    thickness = 1
    x_start = 10
    y_start = 40
    letter_spacing = 30

    text = "NCHU CSE"
    colors = [(0, 0, 0), (255, 255, 255)]

    for i, letter in enumerate(text):
        color = colors[i % 2]
        cv2.putText(image, letter, (x_start + i * letter_spacing, y_start),
font, font_scale, color, thickness, cv2.LINE_AA)

    cv2_imshow(image)
    return image

# add_NCHU_text
image_with_text = add_interlaced_text(gray_image)
```

這段程式碼模擬了 impulse noise，即鹽和胡椒雜訊，透過在灰階圖像的隨機位置添加白色和黑色像素。將鹽的機率設為 1/3，胡椒的機率設為 1/4，根據圖像的大小計算了添加鹽和胡椒雜訊的像素數量，然後在隨機生成的座標上將對應的像素值設置為白色和黑色。最後返回添加了雜訊的圖像。

```
def saltAndPepperNoise(image):
    if len(image.shape) != 2:
        raise ValueError("Image should be grayscale (2D array).")

    rows, cols = image.shape
    total_pixels = image.size
    noisy_image = np.copy(image)
    # 計算應添加鹽雜訊和胡椒雜訊的像素數
    num_salt = np.ceil(1/3 * image.size).astype(int)
    num_pepper = np.ceil(1/4 * image.size).astype(int)

    # 生成隨機座標
    salt_coords = (np.random.randint(0, rows, num_salt),
                np.random.randint(0, cols, num_salt))
    noisy_image[salt_coords] = 255

    pepper_coords = (np.random.randint(0, rows, num_pepper),
                  np.random.randint(0, cols, num_pepper))
    noisy_image[pepper_coords] = 0
    # cv2_imshow(noisy_image)
    return noisy_image

# Impulse Noise
noisy_image = saltAndPepperNoise(image_with_text)
cv2_imshow(noisy_image)
noisy_image.shape
```

這段程式碼實現了中值濾波器，它對每個像素的周圍區域進行排序，並將中間值（第 5 個值）作為該像素的新值，從而消除了圖像中的鹽和胡椒雜訊。

```
def medianFilter(img_noisy):
    if len(img_noisy.shape) != 2:
        raise ValueError("Image should be grayscale (2D array).")
```

```
    m, n = img_noisy.shape
    img_new = np.zeros((m, n), dtype=np.uint8)

    for i in range(1, m-1):
        for j in range(1, n-1):
            temp = [
                img_noisy[i-1, j-1], img_noisy[i-1, j], img_noisy[i-1, j+1],
                img_noisy[i, j-1],   img_noisy[i, j],   img_noisy[i, j+1],
                img_noisy[i+1, j-1], img_noisy[i+1, j], img_noisy[i+1, j+1]
            ]

            temp = sorted(temp)
            img_new[i, j] = temp[4]

    cv2_imshow(img_new)
    # cv2.imwrite('medianFilter.jpg', img_new)
    return img_new

filtered_image1 = medianFilter(noisy_image)
```

這段程式碼實現的是一種改進型的中值濾波器，稱為"switching median filter"。與原始的中值濾波器相比，它多了一個可調節的閾值參數。當像素值與中值的差異超過閾值時，這個改進型濾波器會將該像素值替換為中值。這樣可以更好地處理噪聲，特別是對於強烈的噪聲有較好的效果。而原始的中值濾波器則是將每個像素的周圍區域的像素值排序後取中值，直接使用中間值作為該像素的新值。

```
def switching_median_filter(img_noisy, kernel_size, threshold=30):
    pad_width = kernel_size // 2
    padded_image = np.pad(img_noisy, pad_width=pad_width, mode='reflect')

    output_image = np.copy(img_noisy)
    m, n = img_noisy.shape

    for i in range(1, m-1):
        for j in range(1, n-1):
            temp = [
                img_noisy[i-1, j-1], img_noisy[i-1, j], img_noisy[i-1, j+1],
                img_noisy[i, j-1],   img_noisy[i, j],   img_noisy[i, j+1],
```

```
                img_noisy[i+1, j-1], img_noisy[i+1, j], img_noisy[i+1, j+1]
            ]
            temp = sorted(temp)
            # output_image[i, j] = temp[4]

            if np.abs(int(img_noisy[i, j]) - int(temp[4])) > threshold:
                output_image[i, j] = temp[4]

    cv2_imshow(output_image)
    return output_image

filtered_image2 = switching_median_filter(noisy_image, kernel_size=3)
```

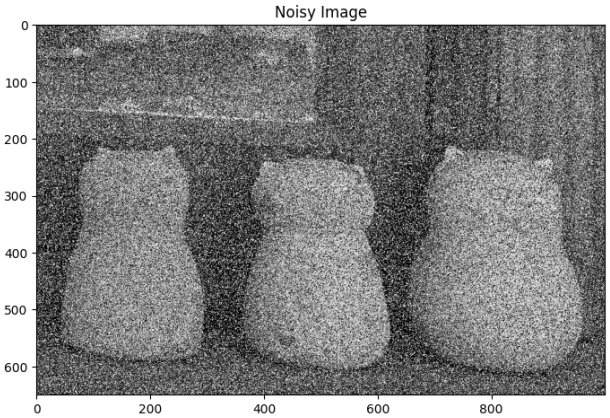
計算經過 Filter 處理後的 PSNR

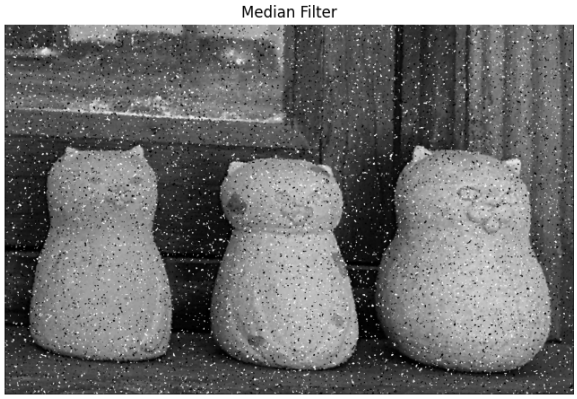
```
def Psnr(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return float('inf')
    # 假設圖像的最大像素值是 255
    return 10 * np.log10((255.0 ** 2) / mse)

# 計算 PSNR
print('Median Filter PSNR:', round(Psnr(gray_image, filtered_image1), 2))
print('Switching Median Filter PSNR:', round(Psnr(gray_image,
filtered_image2), 2))
print('PSNR between filter1 and filter2:', round(Psnr(filtered_image1,
filtered_image2), 2))
```

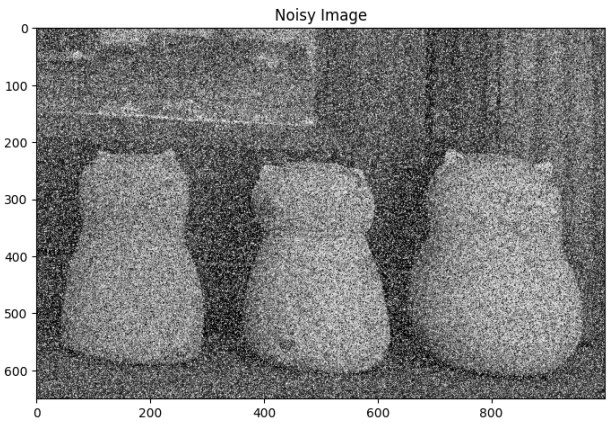
Result:

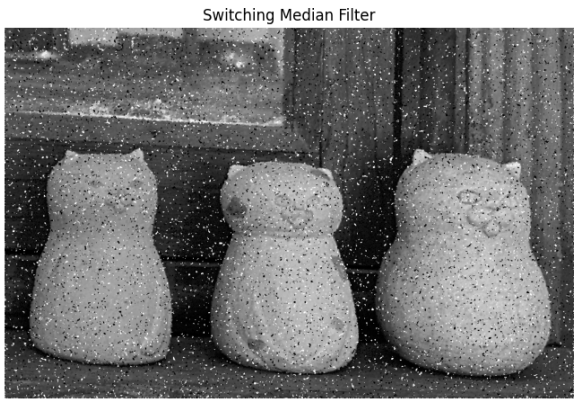
圖組一、

Median Filter PSNR: 34.54
Switching Median Filter PSNR: 35.74
PSNR between filter1 and filter2: 38.5



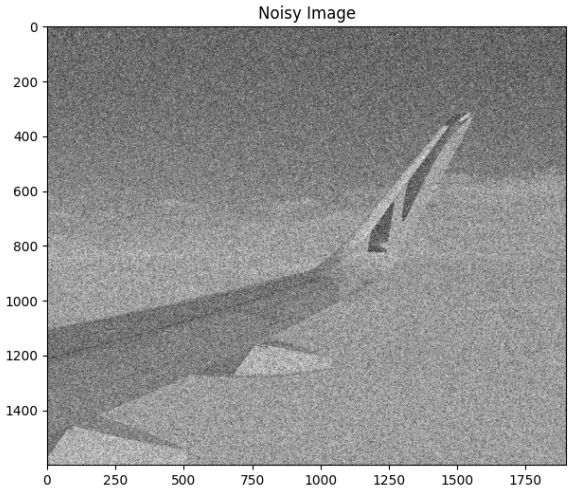


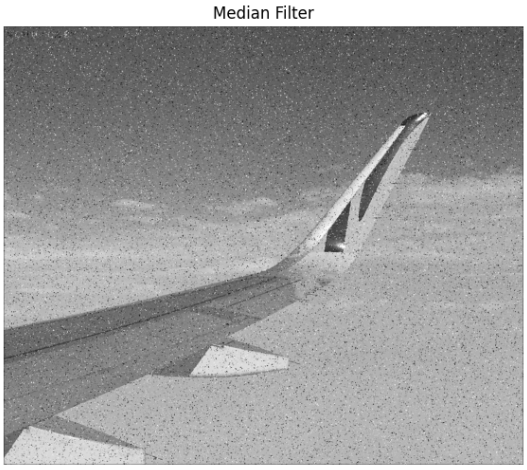


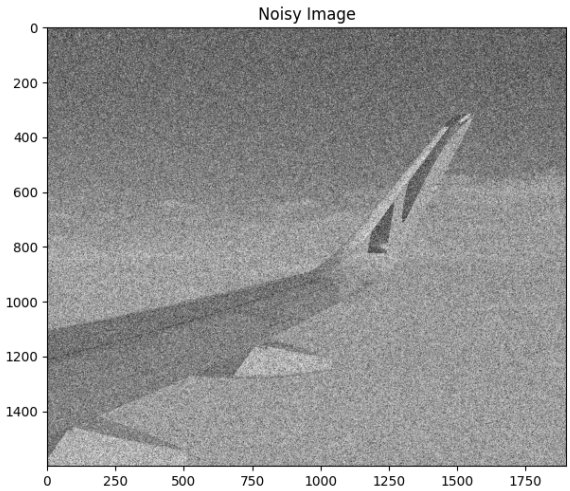


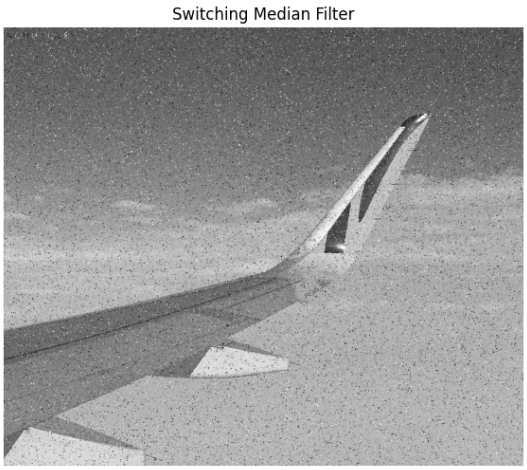
圖組二、

Median Filter PSNR: 38.0
Switching Median Filter PSNR: 38.77
PSNR between filter1 and filter2: 45.63



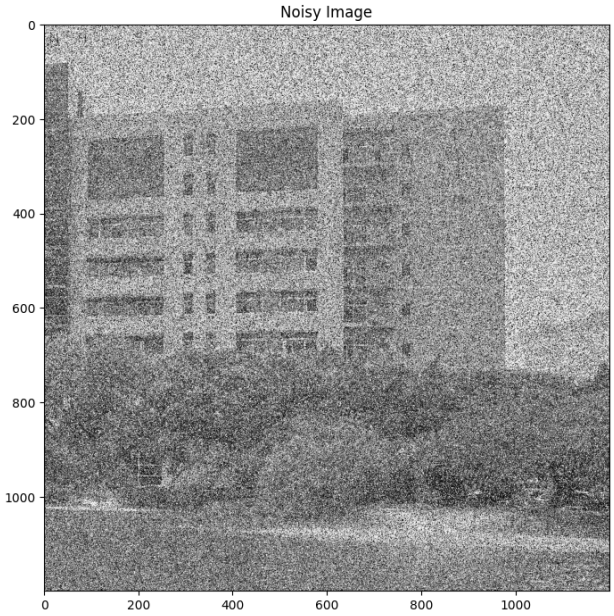


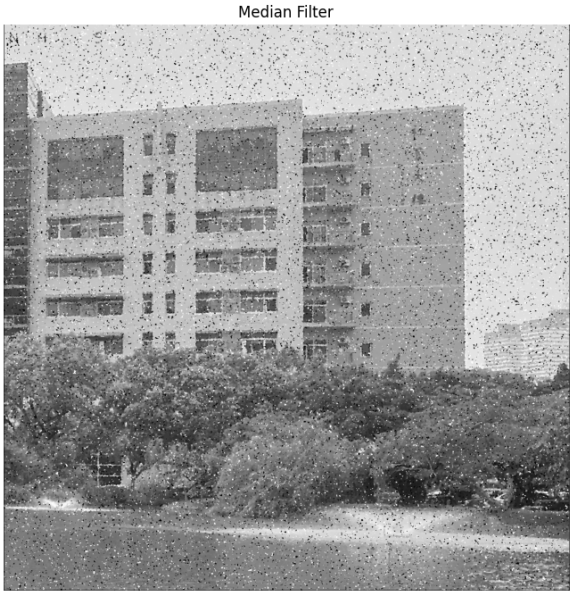


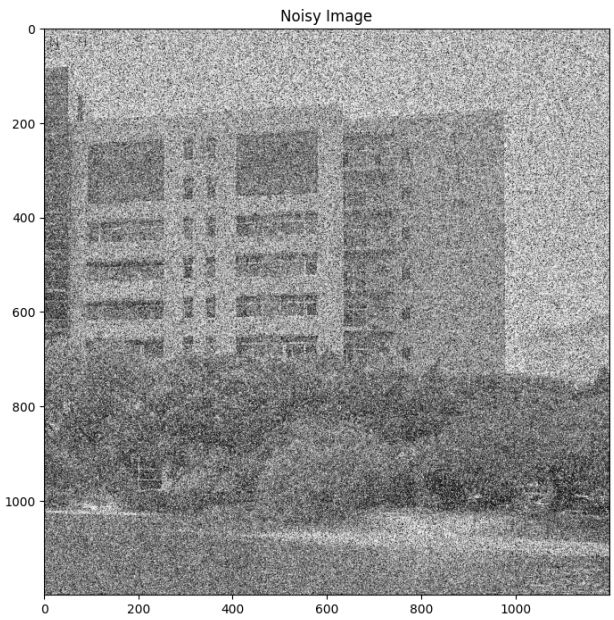


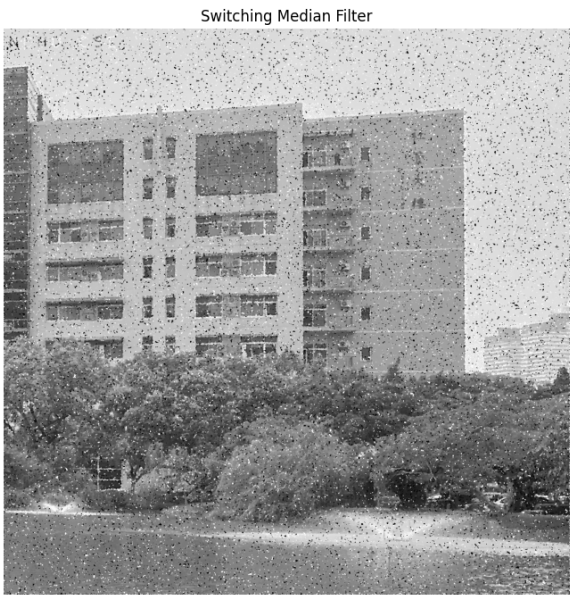
圖組三、

Median Filter PSNR: 33.96
Switching Median Filter PSNR: 35.63
PSNR between filter1 and filter2: 38.57

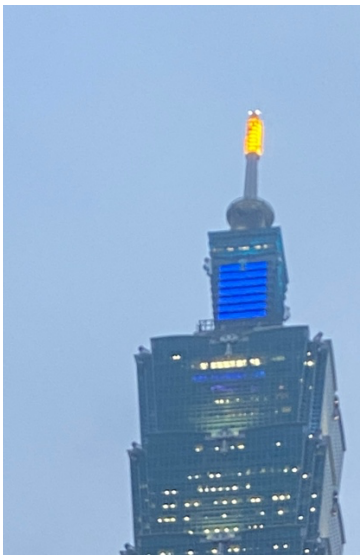




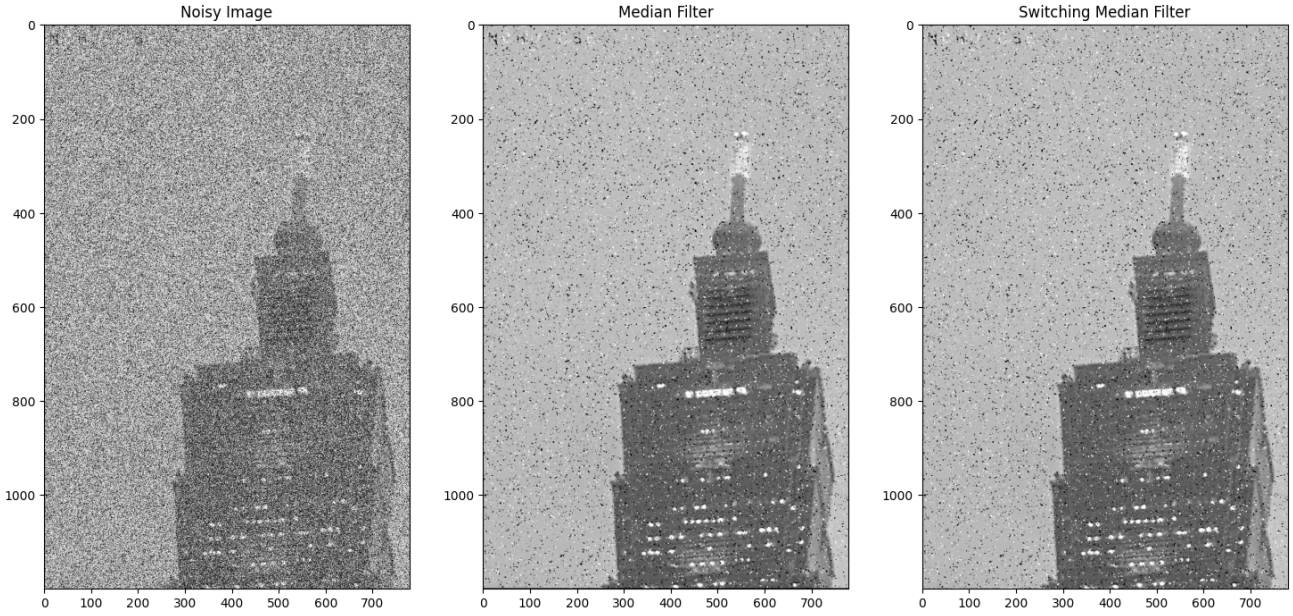




圖組四、



Median Filter PSNR: 37.84
Switching Median Filter PSNR: 38.84
PSNR between filter1 and filter2: 44.53



Discussion:

1. 經過標準中值濾波器(Median Filter)處理後，大部分雜訊被有效去除，文字輪廓基本恢復，但細節仍有些模糊。使用改進的切換中值濾波器(Switching Median Filter)後，不僅去除了大部分雜訊，文字輪廓也相較更加清晰、銳利。兩種濾波器處理效果的對比表明，Switching Median Filter 在保留文字細節方面具有一定優勢。
2. 通過計算 PSNR 值，量化地比較了原始圖像與經過不同濾波器處理後的圖像品質變化，Switching Median Filter 的 PSNR 值比 Median Filter 還要高。
3. 報告中模擬添加的是鹽胡椒雜訊，雖然是一種強烈的脈衝噪聲，但噪聲密度並不算太高。適度的噪聲水平使得中值濾波和改進濾波器都能有效去除大部分噪點，從而使處理後的圖像與原圖的差異較小，導致 PSNR 值能超過 30。