第一題「影像加入高斯或胡椒鹽雜訊」

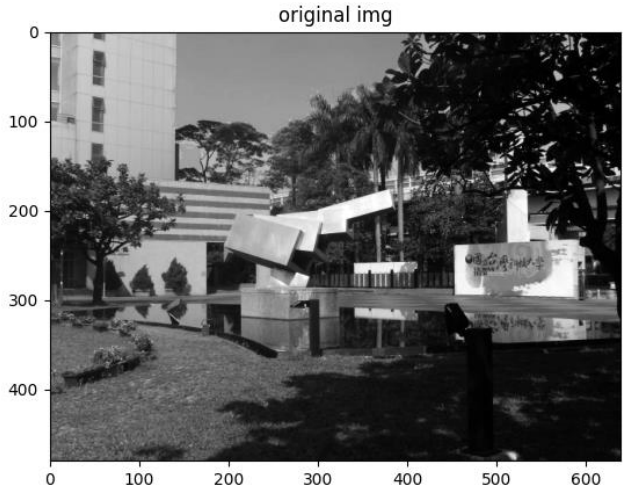
```
#匯入此作業使用到的模組
import cv2  #用於讀取附件灰階影像與儲存影像
import matplotlib.pyplot as plt #用於顯示影像
import random #用於產生亂數
import numpy as np #建立和影像相同大小的空白新影像矩陣與計算高斯分布所用 x 雜訊值的範圍
import math #使用 e.pi 常數與開根號

#定義加入高斯雜訊函式，img 為影像，sigma 為控制高斯分布離散程度的標準差
def add_gaussian_noise(img, sigma):
    #使用變數取得影像的高與寬
    height = img.shape[0]
    width = img.shape[1]

    #建立和影像相同大小的空白新影像矩陣
    gaussian = np.array([[0 for j in range(width)] for i in range(height)])

    #利用公式計算高斯分布，建立從-100 到 100 的範圍 x 值
    x = np.arange(-100, 101)
    exponent = -1 * ((x ** 2) / (2 * (sigma ** 2))) #高斯分布指數
    y = (1 / (sigma * math.sqrt(2 * math.pi))) * (math.e ** exponent) #計算高斯分布 y 值

    #計算機率直方圖
    sum_y = 0
    for i in y:
        sum_y += i #計算 y 的總和
    y = y / sum_y #正規化 y 值

    #累積直方圖
    cumsum_sum = 0
    len_y = 0
    for i in y:
        len_y += 1 #計算 y 的長度
    y_cum = [0] * len_y #建立一個一維陣列存放累計機率

    for i in range(len_y): #利用 y 長度累加機率值
        cumsum_sum += y[i] #累加當前機率值
        y_cum[i] = cumsum_sum #將累積的值存入陣列

    #用雙層迴圈添加高斯雜訊
    for i in range(height):
        for j in range(width):
            rand = random.uniform(0, 1) #產生 0~1 之間的隨機數
```

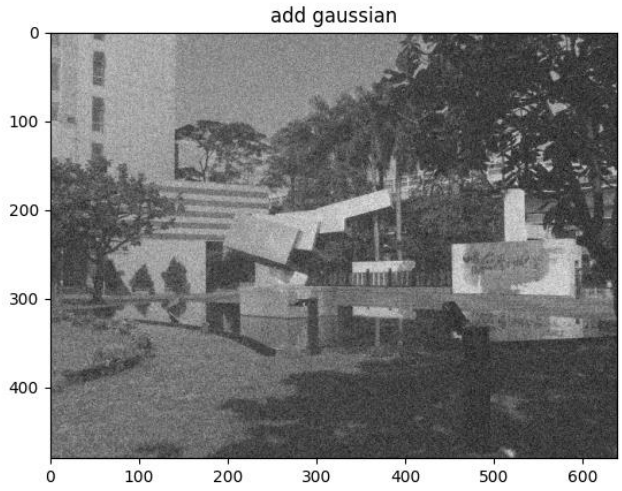
```
            for k in range(len_y): #搜尋累積高斯曲線對應的雜訊數值
                if y_cum[k] > rand:
                    x_noise = x[k]  #選擇對應的 x 雜訊值
                    break
            gaussian[i][j] = img[i][j] #保留原影像像素值
            gaussian[i][j] += x_noise   #添加雜訊到像素值

    return gaussian #返回加入高斯雜訊的影像

#定義加入胡椒鹽雜訊函式，img 為影像，sap_ratio 為胡椒鹽雜訊比例
def add_saltandpepper_noise(img, sap_ratio):
    #使用變數取得影像的高與寬
    height = img.shape[0]
    width = img.shape[1]
    #建立和影像相同大小的空白新影像矩陣
    saltandpepper = np.array([[0 for j in range(width)] for i in range(height)])

    #用雙層迴圈添加胡椒鹽雜訊
    for i in range(height):
        for j in range(width):
            rand = random.uniform(0, 100) #產生 0~100 的隨機數
            if rand > sap_ratio: #若隨機數大於雜訊比例，則保留原影像像素值
                saltandpepper[i][j] = img[i][j]
            else: #若隨機數小於雜訊比例，則隨機決定黑或白點
                if (random.uniform(0, 1) < 0.5): #產生 0~1 隨機數判斷
                    saltandpepper[i][j] = 255 #若隨機數<0.5 則設為白點(鹽)
                else:
                    saltandpepper[i][j] = 0 #若隨機數>0.5 則設為黑點(胡椒)
    return saltandpepper #返回加入胡椒鹽雜訊的影像
if __name__  == "__main__":
    #讀取附件的 8-bit 灰階影像
    img = 'ntust_gray.jpg'
    original_img = cv2.imread(img, cv2.IMREAD_GRAYSCALE) #以灰階讀取影像

    #顯示輸入影像
    plt.imshow(original_img, cmap='gray') #用灰階模式顯示影像
    plt.title('original img')
    plt.show()

    #輸入雜訊類型選項
    option = int(input("輸入雜訊類型選項 1.高斯雜訊 2.胡椒鹽雜訊 : "))

    #根據選項輸入雜訊的參數：高斯分配的標準差，或胡椒鹽雜訊的濃度
```

```
    if option == 1:
        sigma = int(input("輸入高斯分配的標準差參數 : "))
        gaussian_noise_img = add_gaussian_noise(original_img, sigma) #加入高斯雜訊
        cv2.imwrite('add Gaussian.jpg', gaussian_noise_img) #儲存結果影像
        plt.imshow(gaussian_noise_img, cmap='gray') #顯示加入雜訊後的影像
        plt.title('add gaussian')
        plt.show()
    elif option == 2:
        sap_ratio = int(input("輸入胡椒鹽雜訊的濃度參數 : "))
        saltandpepper_noise_img = add_saltandpepper_noise(original_img,  sap_ratio) #加入胡椒
鹽雜訊
        cv2.imwrite('add Salt and Pepper.jpg', saltandpepper_noise_img) #儲存結果影像
        plt.imshow(saltandpepper_noise_img, cmap='gray') #顯示加入雜訊後的影像
        plt.title('add Salt and Pepper')
        plt.show()
```

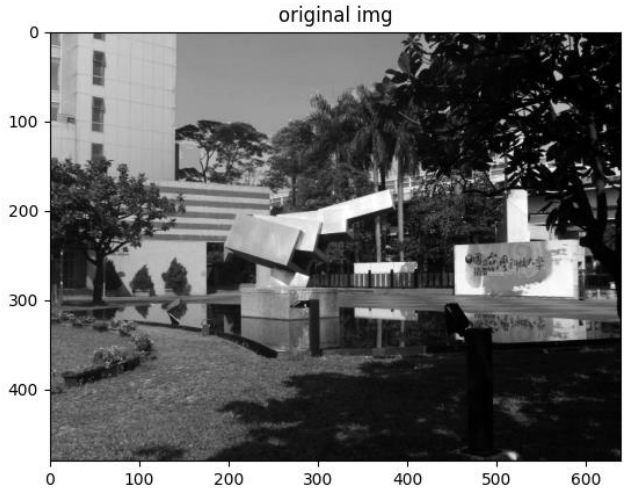
執行結果:高斯雜訊、胡椒鹽雜訊

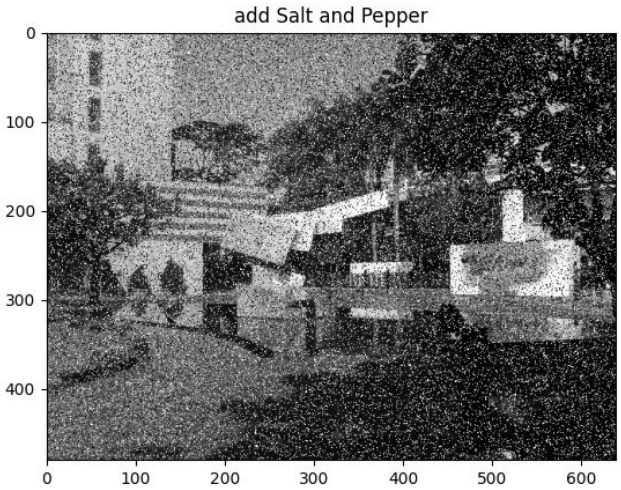
```
PS C:\Users\Xuan\Desktop\Color Image Processing\HW1> & C:/Users/Xuan/anaconda3/envs/py39/python.exe "c:/Users/Xuan/Desktop/Color Image Processing/HW1/HW1_1.py"
輸入雜訊類型選項 1.高斯雜訊 2.胡椒鹽雜訊 : 1
輸入高斯分配的標準差參數 : 30
```






```
PS C:\Users\Xuan\Desktop\Color Image Processing\HW1> & C:/Users/Xuan/anaconda3/envs/py39/python.exe "c:/Users/Xuan/Desktop/Color Image Processing/HW1/HW1_1.py"
輸入雜訊類型選項 1.高斯雜訊 2.胡椒鹽雜訊 : 1
輸入高斯分配的標準差參數 : 15
```





第二題「影像均值濾波或中值濾波」

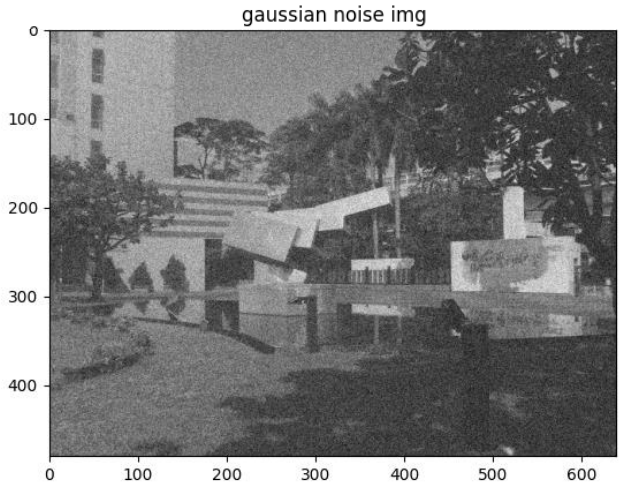
```
#匯入此作業使用到的模組
import cv2 #用於讀取附件灰階影像與儲存影像
import matplotlib.pyplot as plt #用於顯示影像
import numpy as np #建立和影像相同大小的空白新影像矩陣

#定義均值濾波函式，img 為影像，對影像每個像素應用 3X3 均值濾波
def mean_filter(img):
    #使用變數取得影像的高與寬
    height = img.shape[0]
    width = img.shape[1]

    #建立和影像相同大小的空白新影像矩陣
    new_image = np.array([[0 for j in range(width)] for i in range(height)])

    #用雙層迴圈依序對(x,y)像素為中心的 3×3 區域，點對點乘上數值皆為 1/9 的 3×3 濾鏡，將加總的數值填入輸出影像的(x,y)位置
    for i in range(height):
        for j in range(width):
            filter = [[0 for j in range(3)] for i in range(3)] #建立 3x3 的濾波器
            #計算 3x3 範圍內的均值
            for ni in range(-1, 2):
                for nj in range(-1, 2):
                    if 0 <= i + ni < height and 0 <= j + nj < width: #檢查範圍是否超過影像
                        filter[1 + ni][1 + nj] = img[i + ni][j + nj] #若沒有超過範圍則加入影像像素值
                    else:
                        filter[1 + ni][1 + nj] = 0 #若超過範圍設成 0
            mean_value = 0
            #計算 3x3 區域內像素值的總和
            for k in range(3):
                for v in range(3):
                    mean_value += filter[k][v]
            new_image[i][j] = mean_value * (1 / 9)  #計算均值並填入新影像像素值

    return new_image #返回經過均值濾波後的新影像

#定義中值濾波函式，img 為影像，對影像每個像素應用 3x3 中值濾波
def median_filter(img):
    #使用變數取得影像的高與寬
    height = img.shape[0]
    width = img.shape[1]
```

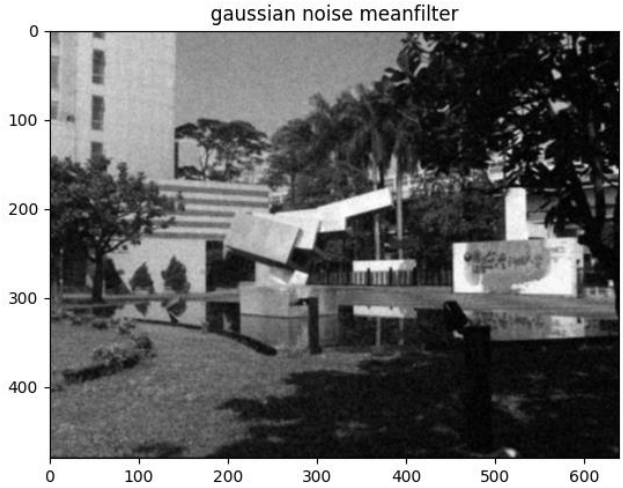
```
    #建立和影像相同大小的空白新影像矩陣
    new_image = np.array([[0 for j in range(width)] for i in range(height)])

    #用雙層迴圈，依序對以(x,y)像素為中心的 3×3 區域，做灰階值排序。將排序後的中位數，填入輸出影像的(x,y)位置
    for i in range(height):
        for j in range(width):
            filter = [[0 for j in range(3)] for i in range(3)] #建立 3x3 的濾波器
            #取得 3x3 範圍內的像素值
            for ni in range(-1, 2):
                for nj in range(-1, 2):
                    if 0 <= i + ni < height and 0 <= j + nj < width: #檢查範圍是否超過影像
                        filter[1 + ni][1 + nj] = img[i + ni][j + nj] #若沒有超過範圍則加入影像像素值
                    else:
                        filter[1 + ni][1 + nj] = 0 #若超過範圍設成 0

            # 將 3x3 範圍內的值轉成一維陣列
            filter_1D = [filter[k][v] for v in range(3) for k in range(3)]

            #進行灰階值排序
            for k in range(9):
                for v in range(0, 9 - k - 1):
                    if filter_1D[v + 1] < filter_1D[v]: #如果第 v 項大於 v+1 項則交換
                        temp = filter_1D[v + 1]
                        filter_1D[v + 1] = filter_1D[v]
                        filter_1D[v] = temp
            median_value = filter_1D[4] #取得排序後的中位數
            new_image[i][j] = median_value #將中位數值填入新影像像素值

    return  new_image #返回經過中值濾波後的新影像

if __name__ == "__main__":
    #輸入空間濾波選項
    option = int(input("輸入空間濾鏡選項 1.均值濾波 2.中值濾波 : "))

    if option == 1:
        #讀取並顯示含有高斯雜訊的影像
        img1 = 'add Gaussian.jpg'
        gaussian_noise_img = cv2.imread(img1, cv2.IMREAD_GRAYSCALE) #以灰階讀取影像
        plt.imshow(gaussian_noise_img, cmap='gray') #用灰階模式顯示影像
        plt.title('gaussian noise img')
        plt.show()
```

```
        #對影像應用均值濾波並顯示結果
        gaussian_noise_mean_filter_img = mean_filter(gaussian_noise_img)
        cv2.imwrite('mean filter img.jpg', gaussian_noise_mean_filter_img) #儲存結果影像
        plt.imshow(gaussian_noise_mean_filter_img, cmap='gray') #顯示經過均值濾波後的影像
        plt.title('gaussian noise meanfilter')
        plt.show()
    elif option == 2:
        #讀取並顯示含有胡椒鹽雜訊的影像
        img2 = 'add Salt and Pepper.jpg'
        saltandpepper_noise_img = cv2.imread(img2, cv2.IMREAD_GRAYSCALE) #以灰階讀取影像
        plt.imshow(saltandpepper_noise_img, cmap='gray') #用灰階模式顯示影像
        plt.title('saltandpepper noise img')
        plt.show()
        #對影像應用中值濾波並顯示結果
        saltandpepper_noise_median_filter_img = median_filter(saltandpepper_noise_img)
        cv2.imwrite('median filter img.jpg', saltandpepper_noise_median_filter_img) #儲存結果影像
        plt.imshow(saltandpepper_noise_median_filter_img, cmap='gray') #顯示經過中值濾波後的影像
        plt.title('saltandpepper noise meanfilter')
        plt.show()
```

執行結果:均值濾波

```
PS C:\Users\Xuan\Desktop\Color Image Processing\HW1> & C:/Users/Xuan/anaconda3/envs/py39/python.exe "c:/Users/Xuan/Desktop/Color Image Processing/HW1/HW1_2.py"
輸入空間濾鏡選項 1.均值濾波 2.中值濾波 : 1
```





執行結果:中值濾波

```
PS C:\Users\Xuan\Desktop\Color Image Processing\HW1> & C:/Users/Xuan/anaconda3/envs/py39/python.exe "c:/Users/Xuan/Desktop/Color Image Processing/HW1/HW1_2.py"
輸入空間濾鏡選項 1.均值濾波 2.中值濾波 : 2
```