R10922123　周昱豪

Code:

1. 產生 Gaussian noise and Salt-and-Pepper 的 code

```python
def GetGaussianNoise(img, threshold):
    return img + threshold * np.random.normal(0, 1, img.shape)

def GetSaltAndPepper(img, threshold):
    res = np.copy(img)
    randomValue = np.random.uniform(0, 1, res.shape)
    row, col = img.shape
    for i in range(row):
        for j in range(col):
            if randomValue[i][j] < threshold:
                res[i][j] = 0
            elif randomValue[i][j] > 1 - threshold:
                res[i][j] = 255
            else:
                continue
    return res
```

2. Box filter and Median filter: 基本上都是跟著 PPT 上做

```python
def boxFilter(img, size):
    kernel = []
    for i in range(-size // 2, size // 2):
        for j in range(-size // 2, size // 2):
            kernel.append([i, j])
    normalize = size * size
    row, col = img.shape
    res = np.zeros(img.shape)

    for i in range(row):
        for j in range(col):
            sum = 0
            for ele in kernel:
                eleI, eleJ = ele
                if i + eleI >= 0 and i + eleI < row and j + eleJ >= 0 and j + eleJ < col:
                    sum += img[i + eleI, j + eleJ]
            res[i][j] = sum / normalize
    return res
```

```python
def medianFilter(img, size):
    kernel = []
    for i in range(-size // 2, size // 2):
        for j in range(-size // 2, size // 2):
            kernel.append([i, j])

    res = np.zeros(img.shape)
    row, col = img.shape

    for i in range(row):
        for j in range(col):
            medianSet = []
            for k in kernel:
                ki, kj = k
                if i + ki >= 0 and i + ki < row and j + kj >= 0 and j + kj < col:
                    medianSet.append(img[i+ki][j+kj])
            res[i][j] = np.median(medianSet)
    return res
```

3. SNR 及各個結果跑出來的結果: 跟著投影片做，一開始先除 255 做 normalize 不然會 overflow.

```python
def SNR(img, noiseImg):
    img = img / 255
    noiseImg = noiseImg / 255

    if img.shape != noiseImg.shape:
        print("Image size must be same.")
        return

    us = 0
    VS = 0
    uNoise = 0
    VN = 0
    row, col = img.shape

    for i in range(row):
        for j in range(col):
            us = us + img[i][j]
    us = us / (row * col)

    for i in range(row):
        for j in range(col):
            VS = VS + math.pow(img[i][j] - us, 2)
    VS = VS / (row * col)

    for i in range(row):
        for j in range(col):
            uNoise = uNoise + (noiseImg[i][j] - img[i][j])
    uNoise = uNoise / (row * col)

    for i in range(row):
        for j in range(col):
            VN = VN + math.pow(noiseImg[i][j] - img[i][j] - uNoise, 2)
    VN = VN / (row * col)

    return 20 * math.log(math.sqrt(VS) / math.sqrt(VN), 10)
```

```
gaussianImage_10_SNR = 13.586684135545916
gaussianImage_30_SNR = 4.060634967276163
saltAndPepper_0.05_SNR = 0.9109636739377763
saltAndPepper_0.1_SNR = -2.072483710069579
gaussianImage_10_BF3_SNR = 11.111537202059152
gaussianImage_10_BF5_SNR = 11.226577570238167
gaussianImage_30_BF3_SNR = 9.381058312765244
gaussianImage_30_BF5_SNR = 10.507167192147811
saltAndPepper_0.05_BF3_SNR = 7.747552096737459
saltAndPepper_0.05_BF5_SNR = 9.354122103564995
saltAndPepper_0.1_BF3_SNR = 5.525907019306557
saltAndPepper_0.1_BF5_SNR = 7.546649625596008
gaussianImage_10_MF3_SNR = 11.38268222171413
gaussianImage_10_MF5_SNR = 11.85053943159046
gaussianImage_30_MF3_SNR = 8.963151133255158
gaussianImage_30_MF5_SNR = 10.74985212490406
saltAndPepper_0.05_MF3_SNR = 11.527530775181132
saltAndPepper_0.05_MF5_SNR = 11.83658559544174
saltAndPepper_0.1_MF3_SNR = 10.536956872650471
saltAndPepper_0.1_MF5_SNR = 11.73739453542163
gaussianImage_10_OC_SNR = 13.271450007185388
gaussianImage_30_OC_SNR = 11.172346418330017
gaussianImage_10_CO_SNR = 13.589686807035338
gaussianImage_30_CO_SNR = 11.117000794482175
saltAndPepper_0.05_OC_SNR = 5.7638790575066645
saltAndPepper_0.1_OC_SNR = -2.056008954060036
saltAndPepper_0.05_CO_SNR = 5.49777282451031
saltAndPepper_0.1_CO_SNR = -2.447931602576144
```

Result:

1. Gaussian noise with threshold 10 與他們被 filter 弄出來後的結果

Gaussian noise with threshold 10, SNR = 13.586684135545916

	
Box filter 3x3, SNR = 11.111537202059152	Box filter 5x5, SNR = 11.226577570238167
	
Median filter 3x3, SNR = 11.38268222171413	Median filter 5x5, SNR = 11.85053943159046
	
Closing-then-Opening, SNR = 13.589686807035338	Opening-then-Closing, SNR = 13.271450007185388

2. Gaussian noise with threshold 30

Gaussian noise with threshold 30, SNR = 4.060634967276163

Box filter 3x3, SNR = 9.381058312765244

Box filter 5x5, SNR = 10.507167192147811

Median filter 3x3, SNR = 8.963151133255158

Median filter 5x5, SNR = 10.74985212490406

Closing-then-Opening, SNR = 11.117000794482175

Opening-then-Closing, SNR = 11.172346418330017

3. Salt-and-Pepper with probability 0.05

Salt-and-Pepper with probability 0.05, SNR = 0.9109636739377763

Box filter 3x3, SNR = 7.747552096737459

Box filter 5x5, SNR = 9.354122103564995

Median filter 3x3, SNR = 11.527530775181132

Median filter 5x5, SNR = 11.83658559544174

Closing-then-Opening, SNR = 5.49777282451031

Opening-then-Closing, SNR = 5.7638790575066645

4. Salt-and-Pepper with probability 0.1

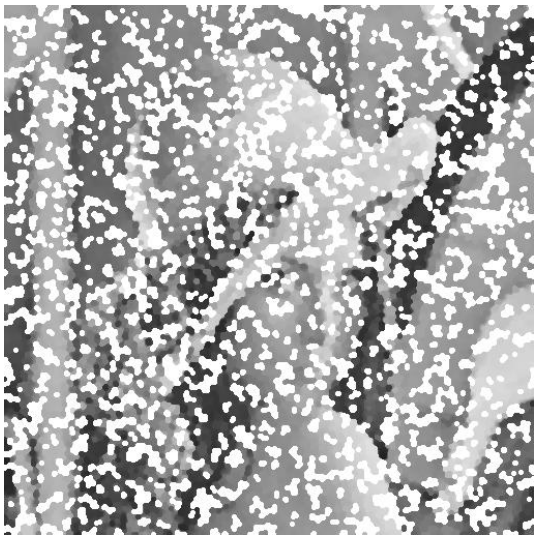
Salt-and-Pepper with probability 0.1, SNR = -2.072483710069579

Box filter 3x3, SNR = 5.525907019306557

Box filter 5x5, SNR = 7.546649625596008

Median filter 3x3, SNR = 10.536956872650471

Median filter 5x5, SNR = 11.73739453542163



Closing-then-Opening, SNR = -2.447931602576144

Opening-then-Closing, SNR = -2.056008954060036