2021 CV HW8 Report

Gaussian Noise
Amplitude = 10
SNR = 13.5829

Box 3x3
SNR = 17.7523

Median 3x3
SNR = 17.6634

Opening-then-Closing
SNR = 13.2994

Box 5x5
SNR = 14.8773

Median 5x5
SNR = 16.0050

Closing-then-Opening
SNR = 13.5968

Gaussian Noise
Amplitude = 30
SNR = 4.1575

Box 3x3
SNR = 12.6090

Median 3x3
SNR = 11.0682

Opening-then-Closing
SNR = 11.2466

Box 5x5
SNR = 13.3249

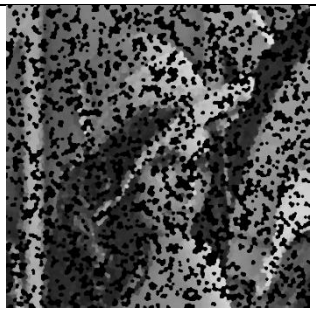
Median 5x5
SNR = 12.9147

Closing-then-Opening
SNR = 11.2374

Salt-and-Pepper Noise
Probability = 0.1
SNR = -2.1098

Box 3x3
SNR = 6.3343

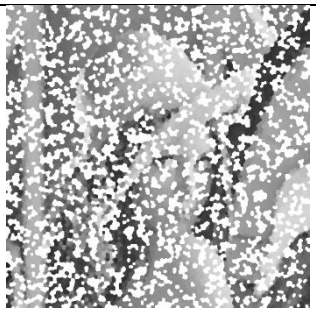
Median 3x3
SNR = 14.9065



Opening-then-Closing
SNR = -2.1658

Box 5x5
SNR = 8.5137

Median 5x5
SNR = 15.7722



Closing-then-Opening
SNR = -2.6901

Salt-and-Pepper Noise
Probability = 0.05
SNR = 0.8953

Box 3x3
SNR = 9.4260

Median 3x3
SNR = 19.1243

Opening-then-Closing
SNR = 5.6342

Box 5x5
SNR = 11.1306

Median 5x5
SNR = 16.3645

Closing-then-Opening
SNR = 5.3967

Noise Removal

(a) Generate noisy images with gaussian noise (amplitude of 10 and 30)

利用 np.random.normal() 函式產生高斯分布（μ = 0, σ = 1）的隨機數。

```
mu, sigma = 0, 1
s1 = np.random.normal(mu, sigma, (h, w))
```

將 pixel value 加上 amplitude 乘上該位置的隨機數。(若運算結果不在 0~255 中，需做處理)

```
temp = img[i][j] + 10 * s1[i][j] # amplitude = 10
```

(b) Generate noisy images with salt-and-pepper noise (probability 0.1 and 0.05)

利用 np.random.uniform() 函式產生均勻分布的隨機數（介在 0~1 之間）。

```
s2 = np.random.uniform(0, 1, (h, w))
```

以 Probability = 0.05 為例，若該位置的隨機數<0.05，pixel value 設為 0；隨機數>0.95，設為 255；其餘不變。

```
if s2[i][j] < 0.05:
        salt_05[i][j] = 0
    elif s2[i][j] > 0.95:
        salt_05[i][j] = 255
    else:
        salt_05[i][j] = img[i][j]
```

(c) Use the 3x3, 5x5 box filter on images generated by (a)(b)

以 3x3 為例，對 Image 做完 padding 後，用 separable box filter recursive 的方法去做 noise cleaning。

```
for j in range(w):
    IBUF[j] = int(noisePadding[1][j])+int(noisePadding[2][j])+int(noisePadding[3][j])

for i in range(2, h-2):
    for j in range(2, w-2):
        box33[i-2][j-2] = IBUF[j-1] + IBUF[j] + IBUF[j+1]
        box33[i-2][j-2] /= 9
    # update IBUF to next row
    for j in range(w):
        IBUF[j] = IBUF[j] - int(noisePadding[i-1][j]) + int(noisePadding[i+2][j])
```

(d) Use 3x3, 5x5 median filter on images generated by (a)(b)

以每個 pixel 作為中心，周圍 3x3 和 5x5 鄰居的中位數作為新的 value。

```
def medianFiter(img, k, i, j):
    l = []
    if k == 3: # 3x3
        for a in (-1, 0, 1):
            for b in (-1, 0, 1):
                l.append(img[i+a][j+b])
        l.sort()
        return l[4]
    elif k == 5: # 5x5
        for a in (-2, -1, 0, 1, 2):
            for b in (-2, -1, 0, 1, 2):
                l.append(img[i+a][j+b])
        l.sort()
        return l[12]
```

(e) Use both opening-then-closing and closing-then opening filter (using the octagonal 3-5-5-5-3 kernel, value = 0) on images generated by (a)(b)

用 cv2.morphologyEx() 函式**先對圖片做 Opening 再做 Closing** 以及**先對圖片做 Closing 再做 Opening**。

```
# opening-then-closing
output = cv2.morphologyEx(value, cv2.MORPH_OPEN, kernel)
output = cv2.morphologyEx(output, cv2.MORPH_CLOSE, kernel)
# closing-then-opening
output = cv2.morphologyEx(value, cv2.MORPH_CLOSE, kernel)
output = cv2.morphologyEx(output, cv2.MORPH_OPEN, kernel)
```

SNR 計算

先將 pixel 值 0~255 normalize 到 0~1。

```
    origin = np.divide(origin, 255)
    noise = np.divide(noise, 255)
```

去計算原圖和雜訊圖片的平均 μ，接著算出標準差 σ。

```
    # compute original image's Mean
    sum = 0
    for i in range(h):
        for j in range(w):
            sum += origin[i][j]
    mu = sum / (h * w)
    # compute original image's Standard Deviation
    sum = 0
    for i in range(h):
        for j in range(w):
            sum += (origin[i][j] - mu)**2
    sigma = sum / (h * w)
    # compute noise image's Mean
    sum = 0
    for i in range(h):
        for j in range(w):
            sum += noise[i][j] - origin[i][j]
    mu_noise = sum / (h * w)
    # compute noise image's Standard Deviation
    sum = 0
    for i in range(h):
        for j in range(w):
            sum += (noise[i][j] - origin[i][j] - mu_noise)**2
    sigma_noise = sum / (h * w)
```

最後將算出的標準差 σ 帶入公式中即可求出 SNR。

```
    snr = 10 * (math.log10(sigma) - math.log10(sigma_noise))
```