Homework 8

Noise Removal

R09921119 許凱荃

- Write a program which does:
  - (a) Generate noisy images with gaussian noise (amplitude of 10 and 30)
  - (b) Generate noisy images with salt-and-pepper noise (probability 0.1 and 0.05)
  - (c) Use the 3x3, 5x5 box filter on images generated by (a)(b)
  - (d) Use 3x3, 5x5 median filter on images generated by (a)(b)
  - (e) Use both opening-then-closing and closing-then opening filter (using the octogonal 3-5-5-5-3 kernel, value = 0) on images generated by (a)(b)
  - You must calculate the signal-to-ratio (SNR) for each instance(4 noisy images and 24 processed images)

Outline:

- Method
- Result
- SNR Table

Method

- Gaussian Noise : 每個 pixel 加上高斯函數所產生之變量。

```python
def generate_gaussian_noise_img(img, amplitude):
    """
    :type img: Image(numpy 2d)
    :type amplitude: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    for x in range(w):
        for y in range(h):
            new_pixel_value = img[x][y] + amplitude * random.gauss(0, 1)

            if new_pixel_value < 0:
                new_pixel_value = 0
            if new_pixel_value > 255:
                new_pixel_value = 255

            new_img[x][y] = new_pixel_value
    return new_img
```

- Salt-and-Pepper Noise : 每個 pixel 隨機設定成 Salt or Pepper。

```python
def generate_salt_and_pepper_img(img, probability):
    """
    :type img: Image(numpy 2d)
    :type probability: double
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    for x in range(w):
        for y in range(h):
            rnd_value = random.uniform(0, 1)

            if rnd_value <= probability:
                new_img[x][y] = 0
            elif rnd_value >= 1 - probability:
                new_img[x][y] = 255
            else:
                continue
    return new_img
```

- Box Filter : 正方形內的 pixel 做平均取其值。
  數值若有差異，可能與邊界有關。

```python
def box_filter(img, filter_size):
    """
    :type img: Image(numpy 2d)
    :type filter_size: pair
    :return type: Image
    """

    w, h = img.shape
    fw, fh = filter_size
    new_img = img.copy()

    for x in range( w ):
        for y in range( h ):
            sum_val = 0
            cnt = 0
            for a in range( -fw//2+1, fw//2+1 ):
                for b in range( -fh//2+1, fh//2+1 ):
                    tx, ty = x+a, y+b
                    if 0 <= tx < w and 0 <= ty < h:
                        sum_val += img[tx][ty]
                        cnt += 1
            avg_pixel_value = sum_val//(cnt)

            # Notice: Non-recurrsive
            new_img[x][y] = avg_pixel_value

    return new_img
```

- Median Filter : 正方形內的 pixel 找其中位數取其值。

```python
def median_filter(img, filter_size):
    """
    :type img: Image(numpy 2d)
    :type filter_size: pair
    :return type: Image
    """

    w, h = img.shape
    fw, fh = filter_size
    new_img = img.copy()

    for x in range( w ):
        for y in range( h ):
            tmp_arr = []
            for a in range( -fw//2+1, fw//2+1 ):
                for b in range( -fh//2+1, fh//2+1 ):
                    tx, ty = x+a, y+b
                    if 0 <= tx < w and 0 <= ty < h:
                        tmp_arr.append(img[tx][ty])
            median_value = sorted(tmp_arr)[len(tmp_arr)//2]

            # Notice: Non-recurrsive
            new_img[x][y] = median_value
    return new_img
```

- Calculate SNR :

  註：假設 Noise image 和 Source Image 同個 Size。

```python
def calc_snr(img, noise_img):
    """
    :type img: Image(numpy 2d)
    :type noise_img: Image(numpy 2d)
    :return type: double
    """

    w, h = img.shape # img and noise_img are the same size.

    vs, mu_s, vn, mu_n = 0, 0, 0, 0

    for x in range(w):
        for y in range(h):
            mu_s += img[x][y]
            mu_n += (int(noise_img[x][y]) - int(img[x][y]))

    mu_s = mu_s / (w*h)
    mu_n = mu_n / (w*h)

    for x in range(w):
        for y in range(h):
            vs += math.pow((img[x][y] - mu_s), 2)
            vn += math.pow((int(noise_img[x][y]) - int(img[x][y]) - int(mu_n)), 2)

    vs = vs / (w*h)
    vn = vn / (w*h)
    return 20 * math.log10( math.sqrt(vs) / math.sqrt(vn) )
```

Image Result

- **Gaussian noise, amplitude of 10**

Gaussian noise, amplitude=10, SNR = 13.581

Box_3x3, SNR = 17.630

Box_5x5, SNR = 14.812

Median_3x3, SNR = 17.604

Median_5x5, SNR = 15.950

Opening-then-closing, SNR = 13.235

Closing-then-opening, SNR = 13.605

- **Gaussian noise, amplitude of 30**

Gaussian noise, amplitude=30, SNR = 4.168

Box_3x3, SNR = 12.566

Box_5x5, SNR = 13.256

Median_3x3, SNR = 11.070

Median_5x5, SNR = 12.874

Opening-then-closing, SNR = 11.155

Closing-then-opening, SNR = 11.234

- **Salt-and-pepper noise, probability=0.1**

Salt-and-pepper noise, probability=0.1, SNR = -2.109

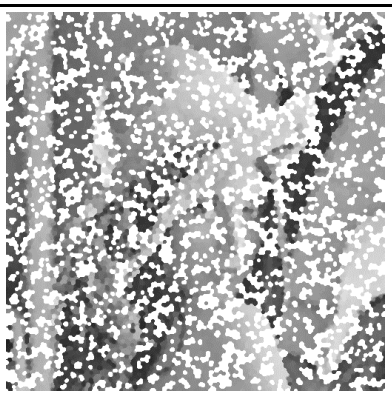
Box_3x3, SNR = 6.337

Box_5x5, SNR = 8.515

Median_3x3, SNR = 15.278

Median_5x5, SNR = 15.684

Opening-then-closing, SNR = -2.057



Closing-then-opening, SNR = -2.569

- **Salt-and-pepper noise, probability=0.05**

Salt-and-pepper noise, probability=0.05, SNR = 0.906

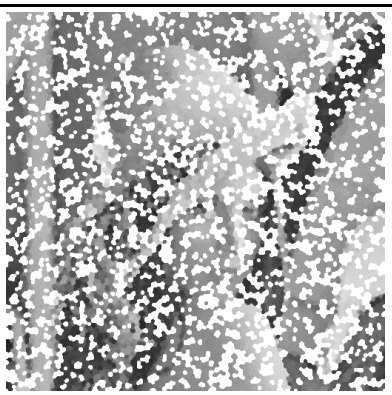
Box_3x3, SNR = 9.430

Box_5x5, SNR = 11.149

Median_3x3, SNR = 18.947

Median_5x5, SNR = 16.359

Opening-then-closing, SNR = 5.242



Closing-then-opening, SNR = 5.270

SNR Table

Filter		Gaussian noise	
		Amplitude = 10	amplitude = 30
No Filter		13.581	4.168
Box	3x3	17.630	12.566
	5x5	14.812	13.256
Median	3x3	17.604	11.070
	5x5	15.950	12.874
Opening-then-closing		13.235	11.155
Closing-then-opening		13.605	11.234

Filter		Salt-and-pepper noise	
		Probability = 0.1	Probability = 0.05
No Filter		-2.109	0.906
Box	3x3	6.337	9.430
	5x5	8.515	11.149
Median	3x3	15.278	18.947
	5x5	15.684	16.359
Opening-then-closing		-2.057	5.242
Closing-then-opening		-2.569	5.270

助教辛苦了 :)