Computer Vision I

Homework 8 - Noise Removal B04902090 資工四 施長元

Usage of the full code:

`python3 main.py [Image_path]`

After the code exit, output file will be in the directory where you execute the code.

Environment: Python3.7 on Windows Linux Subsystem (Ubuntu 18.04.1)

Contents:

Write the following programs:

1. Generate additive white Gaussian noise

   Add an array random by gaussian distribution(mean = 0, sigma = 1) and multiply by amplitude.

```python
# Generate additive white Gaussian noise
def gaussian(img_o, amplitude):
    img_t = img_o + (amplitude * np.random.normal(0, 1, img_o.shape))
    print("\tGaussian noise with amplitude %d: %.4f" % (amplitude, snr(img_t)))
    return img_t
```

	
Amplitude: 10; SNR = 13.5843	Amplitude: 30; SNR = 4.0546

2. Generate salt-and-pepper noise

   Random a value using uniform distribution, and decide the pixel value with this random value and threshold.

```python
# Generate salt-and-pepper noise
def saltpepper(img_o, threshold):
    img_t = np.array(img_o)
    for i in range(img_o.shape[0]):
        for j in range(img_o.shape[1]):
            value = random.uniform(0, 1)
            if value < threshold: img_t[i, j] = 0
            elif value > (1 - threshold): img_t[i, j] = 255
    print("\tsalt-and-pepper noise with threshold %.2f: %.4f" % (threshold, snr(img_t)))
    return img_t
```

	
Threshold: 0.1; SNR = -1.4859	Threshold: 0.05; SNR = 1.0846

Below is the code for box filter and median filter:

```
# Run box(True)/median(False) filter (3X3, 5X5) on all noisy images
def filters(img_o, size, box = True):
  img_t = np.array(img_o)
  img_o_n = np.pad(img_o, pad_width=2, mode='constant', constant_values=0)
  for i in range(img_t.shape[0]):
    for j in range(img_t.shape[1]):
      if box: img_t[i, j] = np.mean(img_o_n[2+i-size : 2+i+size, 2+j-size : 2+j+size])
      else: img_t[i, j] = np.median(img_o_n[2+i-size : 2+i+size, 2+j-size : 2+j+size])
  if box: print("\tBox_filter %dx%d: %.4f" % (size*2+1, size*2+1, snr(img_t)))
  else: print("\tMedian_filter %dx%d: %.4f" % (size*2+1, size*2+1, snr(img_t)))
  return img_t
```

3. Run box filter (3X3, 5X5) on all noisy images

   Take the 3x3 kernel's **mean** as the pixel value.

	
3x3, gaussian 10; SNR = 14.0580	3x3, gaussian 30; SNR = 9.0096

	
3x3, salt-and-pepper 0.1; SNR = -7.0458	3x3, salt-and-pepper 0.05; SNR = -7.6168
	
5x5, gaussian 10; SNR = 13.6921	5x5, gaussian 30; SNR = 11.9256
	
5x5, salt-and-pepper 0.1; SNR = -7.6168	5x5, salt-and-pepper 0.05; SNR = -7.7857

4. Run median filter (3X3, 5X5) on all noisy images

   Same as the code above, but use median as pixel value

	
3x3, gaussian 10; SNR = 13.6400	3x3, gaussian 30; SNR = 8.2984
	
3x3, salt-and-pepper 0.1; SNR = -7.9080	3x3, salt-and-pepper 0.05; SNR = -8.1907
	
5x5, gaussian 10; SNR = 13.9323	5x5, gaussian 30; SNR = 11.1284

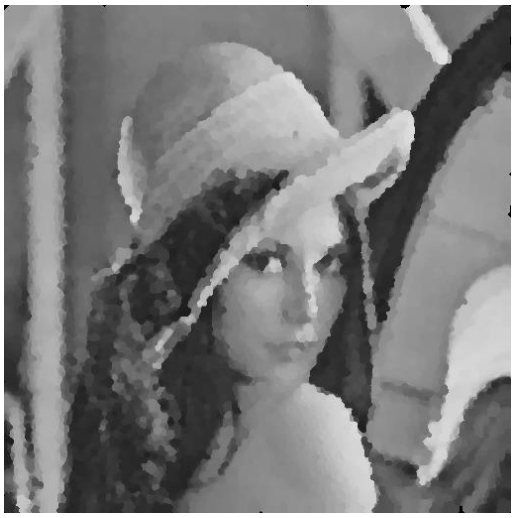
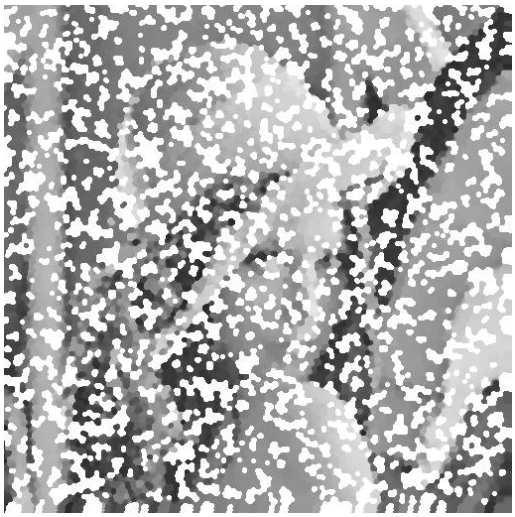
	
5x5, salt-and-pepper 0.1; SNR = -8.1540	5x5, salt-and-pepper 0.05; SNR = -8.1981

5. Run opening followed by closing and closing followed by opening

   Include the code of hw5, and call the opening and closing functions.

```
# Opening then closing(True),  Closing then opening(False)
def openclose(img_o, order = True):
  if order:
    img_t = erosion(dilation(dilation(erosion(img_o))))
    print("\tOpen then close: %.4f" % snr(img_t))
  else:
    img_t = dilation(erosion(erosion(dilation(img_o))))
    print("\tClose then open: %.4f" % snr(img_t))
  return img_t
```

	
Open then close, gaussian 10; SNR = 3.9481	Open then close, gaussian 30; SNR = 4.1072

	
Open then close, salt-and-pepper 0.1; SNR = 3.3421	Open then close, salt-and-pepper 0.05; SNR = 4.4235
	
Close then open, gaussian 10; SNR = 3.8580	Close then open, gaussian 30; SNR = 3.7736
	
Close then open, salt-and-pepper 0.1; SNR = -3.0800	Close then open, salt-and-pepper 0.05; SNR = 2.1562

Reference: SNR

```
# Signal-to-noise ratio
def snr(img_t):
  VS = np.std(img)
  VN = np.std(img_t - img)
  return 20 * math.log((VS/VN), 10)
```