Computer Vision
Homework 8 Report

林義聖
B03902048

December 6, 2016

1 Introduction

In this homework assignment, we're going to generate some noisy image and apply filters to remove the noise. I use *Python* as my programming language and *Pillow* as my image library. It is a fork of *PIL*, which is the original image library of *Python*. And I use *Pillow* for reading input image, and transfering image data into *List* of *Python*.

Figure 1: lena.bmp as Benchmark Image

2 Generate Noisy Image

2.1 Additive White Gaussian Noise

To generate additive white gaussian noise on given image, we use the following function and let $amplitude = 10$ or 30,

$$I(nim, i, j) = I(im, i, j) + amplitude \cdot N(0, 1)$$

```
1 def additive_white(data, amplitude):
2   ret = []
3   for pix in data:
4     ret.append(pix + amplitude * random.gauss(0, 1))
5   return ret
```

(a) original picture (b) Amplitude = 10

Figure 2: Additive white gaussian noise (amplitude = 10)

(a) original picture (b) Amplitude = 30

Figure 3: Additive white gaussian noise (amplitude = 30)

2.2 Salt-and-Pepper Noise

To generate salt-and-pepper noise on given image, we use the following function and let $threshold = 0.1$ or 0.05,

$$I(nim, i, j) = \begin{cases} 0 & \text{if uniform(0,1)} < \dfrac{\text{threshold}}{2} \\ 255 & \text{if uniform(0,1)} > 1 - \dfrac{\text{threshold}}{2} \\ I(im, i, j) & \text{otherwise} \end{cases}$$

```
1  def salt_and_pepper(data, threshold):
2    ret = []
3    threshold = threshold / 2
4    for pix in data:
5      uni = random.uniform(0, 1)
6      if uni < threshold:
7        ret.append(0)
8      elif uni > (1 - threshold):
9        ret.append(255)
10     else:
11       ret.append(pix)
12   return ret
```

(a) original picture (b) Threshold = 0.05

Figure 4: Salt-and-pepper noise (threshold = 0.05)

(a) original picture (b) Threshold = 0.1

Figure 5: Salt-and-pepper noise (threshold = 0.1)

3 Noise Removal

3.1 Box Filter

To use box filter, I just calculate the average of 3×3 or 5×5 neighbors of each pixel. And for border pixel, I just ignore the pixels outside the boundaries. The following is the 3×3 box filter.

```python
def box_filter3x3(data, hei, wid):
  ret = []
  for y in range(hei):
    for x in range(wid):
      s = 0
      t = 0
      for off_x, off_y in product(range(-1,2), range(-1,2)):
        kx, ky = x + off_x, y + off_y
        if 0 <= kx < wid and 0 <= ky < hei:
          s += data[ky * wid + kx]
          t += 1
      ret.append(s / t)
  return ret
```

(a) additive white (amplitude = 10) (b) box filter 3×3 applied

Figure 6: Apply box filter 3×3

(a) additive white (amplitude = 10) (b) box filter 5×5 applied

Figure 7: Apply box filter 5×5

(a) additive white (amplitude = 30) (b) box filter 3×3 applied

Figure 8: Apply box filter 3×3

(a) additive white (amplitude = 30)

(b) box filter 5×5 applied

Figure 9: Apply box filter 5×5

(a) salt-and-pepper (threshold = 0.05)

(b) box filter 3×3 applied

Figure 10: Apply box filter 3×3

(a) salt-and-pepper (threshold = 0.05)

(b) box filter 5×5 applied

Figure 11: Apply box filter 5×5

(a) salt-and-pepper (threshold = 0.1)

(b) box filter 3×3 applied

Figure 12: Apply box filter 3×3

(a) salt-and-pepper (threshold = 0.1) (b) box filter 5×5 applied

Figure 13: Apply box filter 5×5

3.2 Median Filter

To use median filter, I just find the median of 3×3 or 5×5 neighbors of each pixel. And for border pixel, I just ignore the pixels outside the boundaries. The following is the 3×3 median filter.

```
1  def median_filter3x3(data, hei, wid):
2    ret = []
3    for y in range(hei):
4      for x in range(wid):
5        l = []
6        for off_x, off_y in product(range(-1,2), range(-1,2)):
7          kx, ky = x + off_x, y + off_y
8          if 0 <= kx < wid and 0 <= ky < hei:
9            l.append(data[ky * wid + kx])
10       ret.append(median(l))
11   return ret
```

(a) additive white (amplitude = 10) (b) median filter 3×3 applied

Figure 14: Apply median filter 3×3

(a) additive white (amplitude = 10) (b) median filter 5×5 applied

Figure 15: Apply median filter 5×5

(a) additive white (amplitude = 30) (b) median filter 3×3 applied

Figure 16: Apply median filter 3×3

(a) additive white (amplitude = 30) (b) median filter 5×5 applied

Figure 17: Apply median filter 5×5

(a) salt-and-pepper (threshold = 0.05)

(b) median filter 3×3 applied

Figure 18: Apply median filter 3×3

(a) salt-and-pepper (threshold = 0.05)

(b) median filter 5×5 applied

Figure 19: Apply median filter 5×5

(a) salt-and-pepper (threshold = 0.1) (b) median filter 3×3 applied

Figure 20: Apply median filter 3×3

(a) salt-and-pepper (threshold = 0.1) (b) median filter 5×5 applied

Figure 21: Apply median filter 5×5

3.3 Opening-then-Closing

To doing opening followed by closing, I just apply what I have done in the previous assignment.

```
1 def opening(data, hei, wid):
2   return dilation( erosion(data, hei, wid), hei, wid )
3
4 def closing(data, hei, wid):
5   return erosion( dilation(data, hei, wid), hei, wid )
```

```
6
7 ...
8 data = opening(pixellist, hei, wid)
9 data = closing(data, hei, wid)
```

(a) additive white (amplitude = 10) (b) opening-then-closing applied

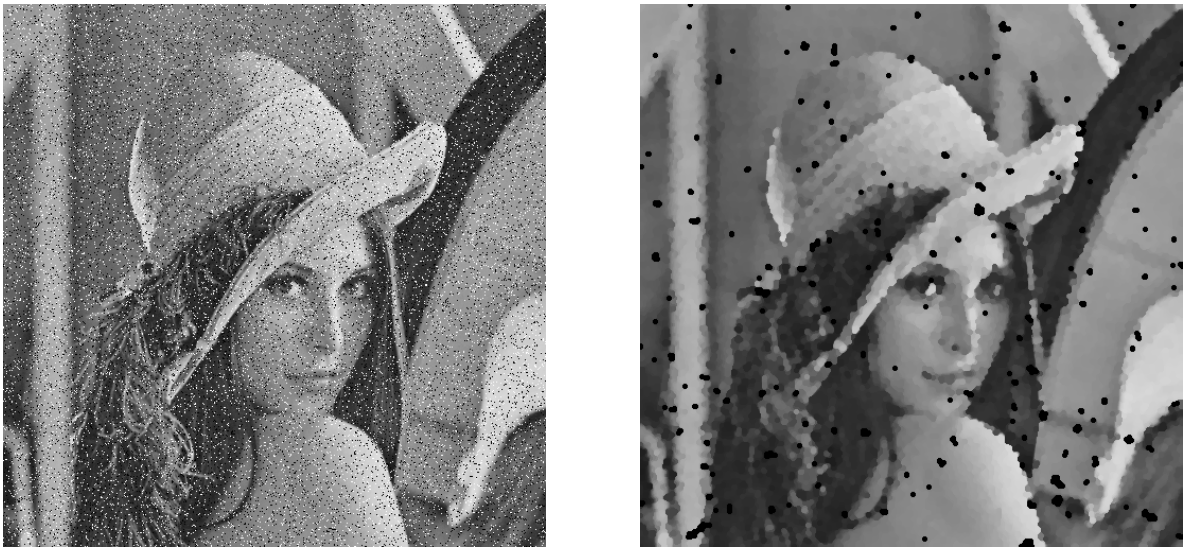
Figure 22: Apply opening-then-closing

(a) additive white (amplitude = 30) (b) opening-then-closing applied

Figure 23: Apply opening-then-closing

(a) salt-and-pepper (threshold = 0.05) (b) opening-then-closing applied

Figure 24: Apply opening-then-closing



(a) salt-and-pepper (threshold = 0.1) (b) opening-then-closing applied

Figure 25: Apply opening-then-closing

3.4 Closing-then-Opening

To doing closing followed by opening, I just apply what I have done in the previous assignment.

```
1 def opening(data, hei, wid):
2   return dilation( erosion(data, hei, wid), hei, wid )
3
4 def closing(data, hei, wid):
5   return erosion( dilation(data, hei, wid), hei, wid )
```

```
6
7 ...
8 data = closing(pixellist, hei, wid)
9 data = opening(data, hei, wid)
```

(a) additive white (amplitude = 10) (b) closing-then-opening applied

Figure 26: Apply closing-then-opening

(a) additive white (amplitude = 30) (b) closing-then-opening applied

Figure 27: Apply closing-then-opening

(a) salt-and-pepper (threshold = 0.05) (b) closing-then-opening applied

Figure 28: Apply closing-then-opening

(a) salt-and-pepper (threshold = 0.1) (b) closing-then-opening applied

Figure 29: Apply closing-then-opening

3.5 Signal-to-ratio

```
1 # ori_pixellist is the data list from original image
2 # proc_pixellist is the data list from processed image
3
4 # calculate mu, mu_n
5 mu = sum(ori_pixellist) / len(ori_pixellist)
6 mu_n = sum([ (proc_pixellist[i] - ori_pixellist[i]) for i in range(len(
      proc_pixellist)) ]) / len(proc_pixellist)
7
```

```
8  # calculate VS
9  vs = sum( [(p - mu)**2 for p in ori_pixellist] ) / len(ori_pixellist)
10 vn = sum( [(proc_pixellist[i] - ori_pixellist[i] - mu_n)**2 for i in range(
       len(proc_pixellist))] ) / len(proc_pixellist)
11
12 # calculate SNR
13 snr = 10 * log10(vs / vn)
```

And the result of the calculation is,

```
SNR of lena.bmp(original) and images/additive-white-10-lena.bmp(processed)
    = 13.599534419205314
SNR of lena.bmp(original) and images/additive-white-30-lena.bmp(processed)
    = 4.168365462765479
SNR of lena.bmp(original) and images/box-filter3x3-additive-white-10-lena.
    bmp(processed) = 17.750910028719368
SNR of lena.bmp(original) and images/box-filter3x3-additive-white-30-lena.
    bmp(processed) = 12.64272803901579
SNR of lena.bmp(original) and images/box-filter3x3-salt-and-pepper-0.05-
    lena.bmp(processed) = 12.12819615866225
SNR of lena.bmp(original) and images/box-filter3x3-salt-and-pepper-0.1-lena
    .bmp(processed) = 9.434489873418514
SNR of lena.bmp(original) and images/box-filter5x5-additive-white-10-lena.
    bmp(processed) = 14.860280590188308
SNR of lena.bmp(original) and images/box-filter5x5-additive-white-30-lena.
    bmp(processed) = 13.304770057934672
SNR of lena.bmp(original) and images/box-filter5x5-salt-and-pepper-0.05-
    lena.bmp(processed) = 12.839450012779476
SNR of lena.bmp(original) and images/box-filter5x5-salt-and-pepper-0.1-lena
    .bmp(processed) = 11.134016148173638
SNR of lena.bmp(original) and images/closing-opening-additive-white-10-lena
    .bmp(processed) = 13.548920190706756
SNR of lena.bmp(original) and images/closing-opening-additive-white-30-lena
    .bmp(processed) = 11.191502796932436
SNR of lena.bmp(original) and images/closing-opening-salt-and-pepper-0.05-
    lena.bmp(processed) = 11.583123010350555
SNR of lena.bmp(original) and images/closing-opening-salt-and-pepper-0.1-
    lena.bmp(processed) = 5.207227434463263
SNR of lena.bmp(original) and images/median-filter3x3-additive-white-10-
    lena.bmp(processed) = 17.6704767282444
SNR of lena.bmp(original) and images/median-filter3x3-additive-white-30-
    lena.bmp(processed) = 11.145526702974655
SNR of lena.bmp(original) and images/median-filter3x3-salt-and-pepper-0.05-
    lena.bmp(processed) = 20.236697407175093
SNR of lena.bmp(original) and images/median-filter3x3-salt-and-pepper-0.1-
    lena.bmp(processed) = 19.261098922929346
SNR of lena.bmp(original) and images/median-filter5x5-additive-white-10-
    lena.bmp(processed) = 15.996660533250076
SNR of lena.bmp(original) and images/median-filter5x5-additive-white-30-
    lena.bmp(processed) = 12.900513044867646
SNR of lena.bmp(original) and images/median-filter5x5-salt-and-pepper-0.05-
    lena.bmp(processed) = 16.604577305951246
SNR of lena.bmp(original) and images/median-filter5x5-salt-and-pepper-0.1-
    lena.bmp(processed) = 16.3418665816682
SNR of lena.bmp(original) and images/opening-closing-additive-white-10-lena
    .bmp(processed) = 13.232635090133504
SNR of lena.bmp(original) and images/opening-closing-additive-white-30-lena
    .bmp(processed) = 11.075501994102645
```

```
SNR of lena.bmp(original) and images/opening-closing-salt-and-pepper-0.05-
    lena.bmp(processed) = 11.229403076481663
SNR of lena.bmp(original) and images/opening-closing-salt-and-pepper-0.1-
    lena.bmp(processed) = 5.709451684942338
SNR of lena.bmp(original) and images/salt-and-pepper-0.05-lena.bmp(
    processed) = 3.909796100756556
SNR of lena.bmp(original) and images/salt-and-pepper-0.1-lena.bmp(processed
    ) = 0.9183721178590989
```

4 How to Use

There are some executable program in my submission, they are

- *generate_noise.py*
- *noise_removal.py*
- *calc_SNR.py*

To use *generate_noise.py*, you need to type "`./generate_noise.py [input image]`". To use *noise_removal.py*, you need to type "`./noise_removal.py --filter=[box|median] [input image]`". To use *calc_SNR.py*, you need to type `./calc_SNR.py [original image] [processed image]`.