# Computer vision homework 8

B04902028 資工三 洪浩翔

Original Lena image

## 1. Gaussian noise:

Amp = 10

Amp = 30

```
def gaussian(original_image , amp):
  print 'gaussian amp:' , amp
  (h , w) = original_image.size
  gaussian_image = original_image.copy()
  for i in range(h):
    for j in range(w):
      value = original_image.getpixel((i , j)) + amp * random.gauss(0 , 1)
      if value >= 255:
        value = 255
      gaussian_image.putpixel((i , j) , int(value))

  return gaussian_image
```

Code of Gaussian noise

## 2. Salt and pepper noise:

Threshold = 0.05

threshold = 0.1

```
def saltandpepper(original_image , thres):
  print 'salt and pepper threshold:' , thres
  (h , w) = original_image.size
  salt_image = original_image.copy()
  for i in range(h):
    for j in range(w):
      value = random.uniform(0 , 1)
      if value < thres:
        salt_image.putpixel((i , j) , 0)
      elif value > (1 - thres):
        salt_image.putpixel((i , j) , 255)

  return salt_image
```

Code of salt and pepper noise

## 3. Box filter:

3x3 with Gaussian(amp = 10)

5x5 with Gaussian(amp = 10)

3x3 with Gaussian(amp = 30)

5x5 with Gaussian(amp = 30)

3x3 with salt(threshold = 0.05)

5x5 with salt(threshold = 0.05)

3x3 with salt(threshold = 0.1)  5x5 with salt(threshold = 0.1)

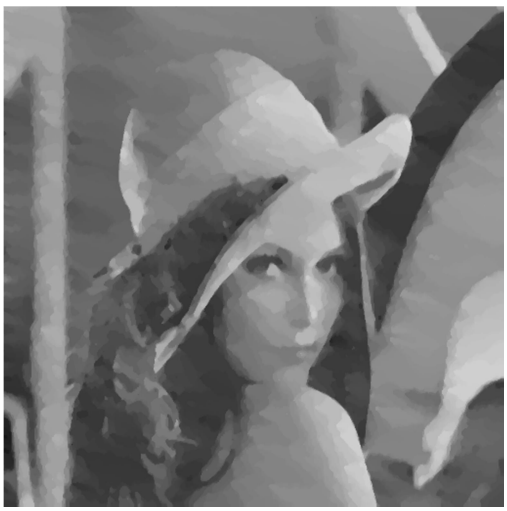
```
def box_filter(noise_image , b_w , b_h):
  (h , w) = noise_image.size
  box_filt_image = noise_image.copy()
  for i in range(h):
    for j in range(w):
      counter = 0
      sum_pixel = 0
      for k in range(-(b_h/2) , (b_h/2)+1):
        for l in range(-(b_w/2) ,( b_w/2)+1):
          if (i + k) >= 0 and (i + k) < h and (j + l) >= 0 and (j + l) < w:
            counter += 1
            sum_pixel += box_filt_image.getpixel((i+k , j+l))
      box_filt_image.putpixel((i , j) , sum_pixel / counter)

  return box_filt_image
```

Code of box filter

## 4. Median filter:

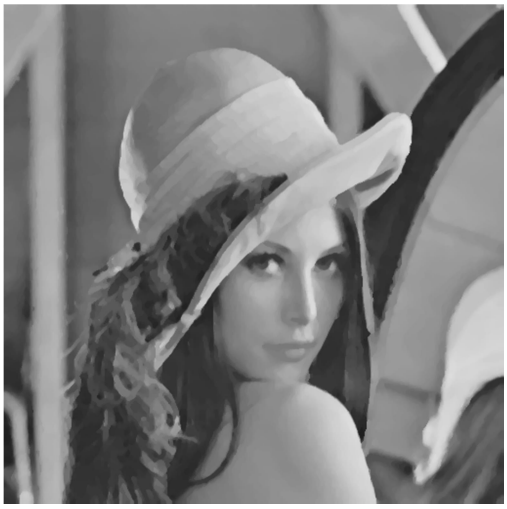
3x3 with Gaussian(amp = 10)  5x5 with Gaussian(amp = 10)

3x3 with Gaussian(amp = 30)



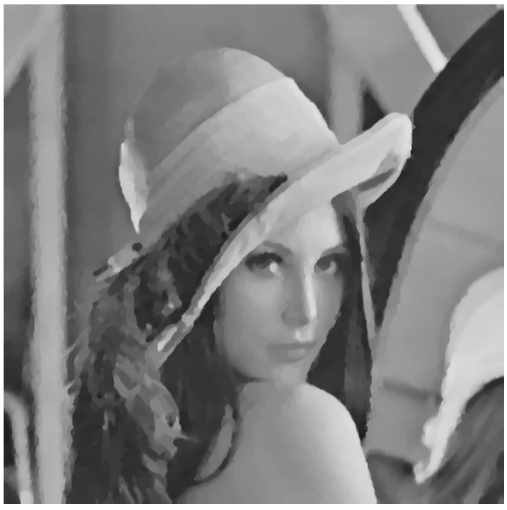
5x5 with Gaussian(amp = 30)

3x3 with salt(threshold = 0.05)



5x5 with salt(threshold = 0.05)

3x3 with salt(threshold = 0.1)



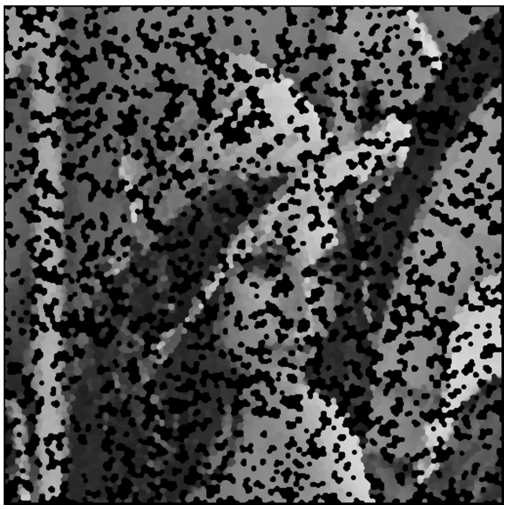
5x5 with salt(threshold = 0.1)

```
def median_filter(noise_image , b_w , b_h):
  (h , w) = noise_image.size
  median_filt_image = noise_image.copy()
  for i in range(h):
    for j in range(w):
      list_arr = []
      for k in range(-(b_h/2) , (b_h/2)+1):
        for l in range(-(b_w/2) , (b_w/2)+1):
          if (i + k) >= 0 and (i + k) < h and (j + l) >= 0 and (j + l) < w:
            list_arr.append(median_filt_image.getpixel((i+k , j+l)))
      list_arr.sort()
      value = 0
      place = len(list_arr)/2
      if len(list_arr) % 2 == 0:
        value = (list_arr[place-1] + list_arr[place]) / 2
      else:
        value = list_arr[place]
      median_filt_image.putpixel((i , j) , value)

  return median_filt_image
```

## 5. Opening then closing:

On Gaussian with amp = 10

On Gaussian with amp = 30



On salt with threshold = 0.05 On salt with threshold = 0.1

```
def dilation(image):
  (h , w) = image.size
  map_record = [[-1 for i in range(h+4)] for j in range(w+4)]
  for i in range(h):
    for j in range(w):
      map_record[i+2][j+2] = image.getpixel((i , j))
  dilation = image.copy()
  for i in range(h):
    for j in range(w):
      tmp = max(map_record[i+1][j] , map_record[i+3][j] , map_record[i+2][j] , map_record[i][j+1] , map_record[i+1][j+1] , map_record[i+2][j+1] ,
                map_record[i+3][j+1] , map_record[i+4][j+1] , map_record[i][j+2] , map_record[i+1][j+2] , map_record[i+2][j+2] , map_record[i+3][j+2] ,
                map_record[i+4][j+2] , map_record[i][j+3] , map_record[i+1][j+3] , map_record[i+2][j+3] , map_record[i+3][j+3] , map_record[i+4][j+3] ,
                map_record[i+1][j+4] , map_record[i+3][j+4] , map_record[i+2][j+4])
      dilation.putpixel((i , j) , tmp)
  return dilation

def erosion(image):
  (h , w) = image.size
  map_record = [[-1 for i in range(h+4)] for j in range(w+4)]
  for i in range(h):
    for j in range(w):
      map_record[i+2][j+2] = image.getpixel((i , j))
  erosion = image.copy()
  for i in range(h):
    for j in range(w):
      tmp = min(map_record[i+1][j] , map_record[i+3][j] , map_record[i+2][j] , map_record[i][j+1] , map_record[i+1][j+1] , map_record[i+2][j+1] ,
                map_record[i+3][j+1] , map_record[i+4][j+1] , map_record[i][j+2] , map_record[i+1][j+2] , map_record[i+2][j+2] , map_record[i+3][j+2] ,
                map_record[i+4][j+2] , map_record[i][j+3] , map_record[i+1][j+3] , map_record[i+2][j+3] , map_record[i+3][j+3] , map_record[i+4][j+3] ,
                map_record[i+1][j+4] , map_record[i+3][j+4] , map_record[i+2][j+4])
      erosion.putpixel((i , j) , tmp)

  return erosion
```

```
def opening(image):
  opening_image = image.copy()
  image = erosion(image)
  opening_image = dilation(image)
  return opening_image

def closing(image):
  closing_image = image.copy()
  image = dilation(image)
  closing_image = erosion(image)
  return  closing_image

def open_then_close(image):
  return closing(opening(image))

def close_then_open(image):
  return  opening(closing(image))

def write_snr(name , snr):
  file.write(name + ': ' + str(snr) + '\n')
```

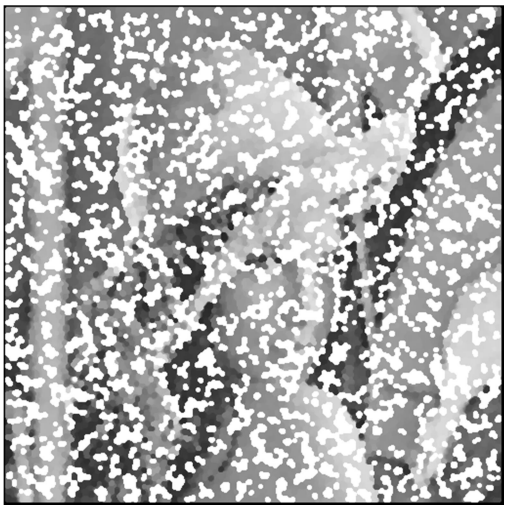
Code of opening then closing

## 6. Closing then opening:

On Gaussian with amp = 10

On Gaussian with amp = 30

On salt with threshold = 0.05



On salt with threshold = 0.1

```python
def opening(image):
  opening_image = image.copy()
  image = erosion(image)
  opening_image = dilation(image)
  return opening_image

def closing(image):
  closing_image = image.copy()
  image = dilation(image)
  closing_image = erosion(image)
  return  closing_image

def open_then_close(image):
  return closing(opening(image))

def close_then_open(image):
  return  opening(closing(image))

def write_snr(name , snr):
  file.write(name + ': ' + str(snr) + '\n')
```

Code of closing then opening

## 7. SNR:

gaussian with apt = 10: 13.615062885
gaussian with apt = 30: 4.16345011273
salt-and-pepper with threshold = 0.1: -2.09480634806
salt-and-pepper with threshold = 0.5: 0.962335873449
box filter with gaussian(apt = 10) and 3 x 3: 17.7586211403
box filter with gaussian(apt = 30) and 3 x 3: 12.6005180655
box filter with gaussian(apt = 10) and 5 x 5: 14.872644709
box filter with gaussian(apt = 30) and 5 x 5: 13.2534778522
median filter with gaussian(apt = 10) and 3 x 3: 17.6998635713
median filter with gaussian(apt = 30) and 3 x 3: 11.0857180412
median filter with gaussian(apt = 10) and 5 x 5: 16.0127050249
median filter with gaussian(apt = 30) and 5 x 5: 12.8580945551
open then close with gaussian(apt = 10): 8.60681375489
open then close with gaussian(apt = 30): 8.63171025024
close then open with gaussian(apt = 10): 7.66090486363
close then open with gaussian(apt = 30): 6.05850931253
box filter with salt and pepper(threshold = 0.1) and 3 x 3: 6.36526871914

box filter with salt and pepper(threshold = 0.05) and 3 x 3: 9.51486144906
box filter with salt and pepper(threshold = 0.1) and 5 x 5: 8.55247615075
box filter with salt and pepper(threshold = 0.05) and 5 x 5: 11.1970926351
median filter with salt and pepper(threshold = 0.1) and 3 x 3: 15.1114750513
median filter with salt and pepper(threshold = 0.05) and 3 x 3: 19.1410327818
median filter with salt and pepper(threshold = 0.1) and 5 x 5: 15.7804135126
median filter with salt and pepper(threshold = 0.05) and 5 x 5: 16.3911094286
open then close with salt and pepper(threshold = 0.1): -2.23485612528
open then close with salt and pepper(threshold = 0.05): 4.41100712156
close then open with salt and pepper(threshold = 0.1): -2.87604166466
close then open with salt and pepper(threshold = 0.05): 4.12927427708

```
def SNR(original_image , noise_image):
  sigma = 0.0
  (h , w) = original_image.size
  for i in range(h):
    for j in range(w):
      sigma += original_image.getpixel((i , j))
  mius = float(sigma / (h*w))
  print 'mius:' , mius

  sigma = 0.0
  for i in range(h):
    for j in range(w):
      sigma += ((original_image.getpixel((i , j)) - mius)**2)
  vs = float(sigma / (h*w))
  print 'vs:' , vs

  sigma = 0.0
  for i in range(h):
    for j in range(w):
      sigma += (noise_image.getpixel((i , j)) - original_image.getpixel((i , j)))
  miun = float(sigma / (h*w))
  print 'miun:' , miun

  sigma = 0.0
  for i in range(h):
    for j in range(w):
      sigma += ((noise_image.getpixel((i , j)) - original_image.getpixel((i , j)) - miun)**2)
  vn = float(sigma / (w*h))
  snr = 20 * math.log((math.sqrt(vs) / math.sqrt(vn)) , 10)
  print 'SNR:' , snr
  return snr
```

Code of SNR