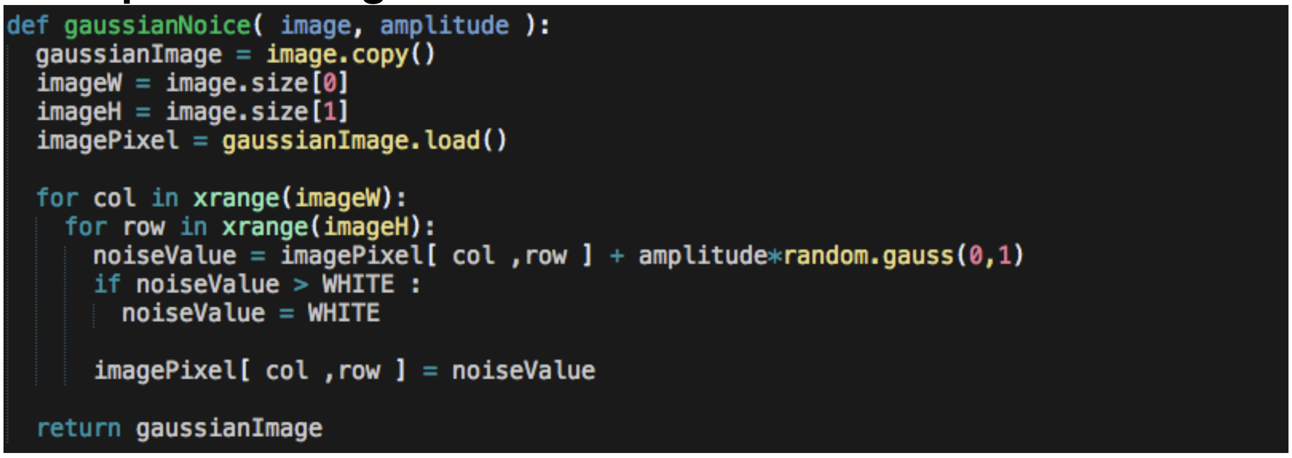
R02944010 李柏緯

Computer Vision Homework 8

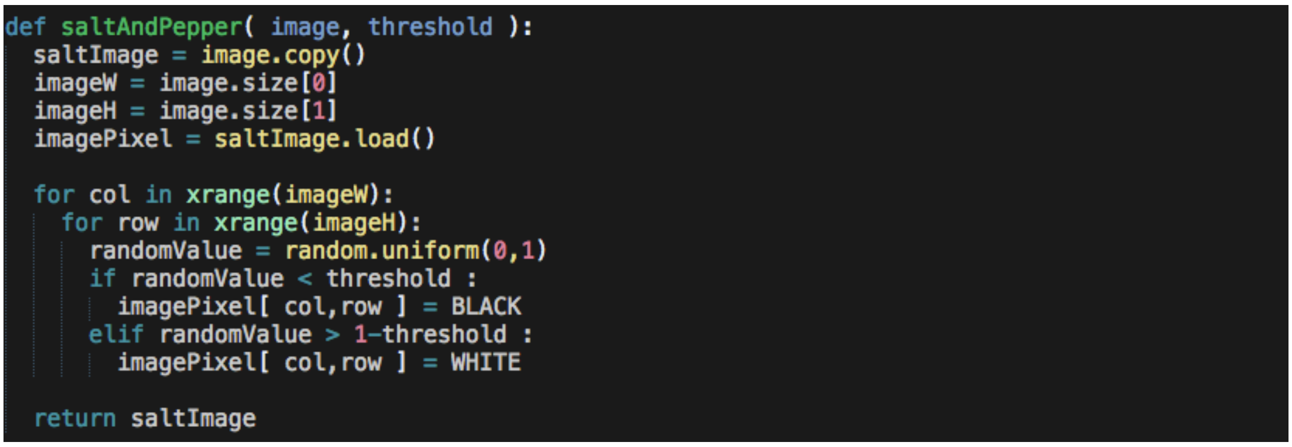
How to run it

```
python image.py lena.bmp
```

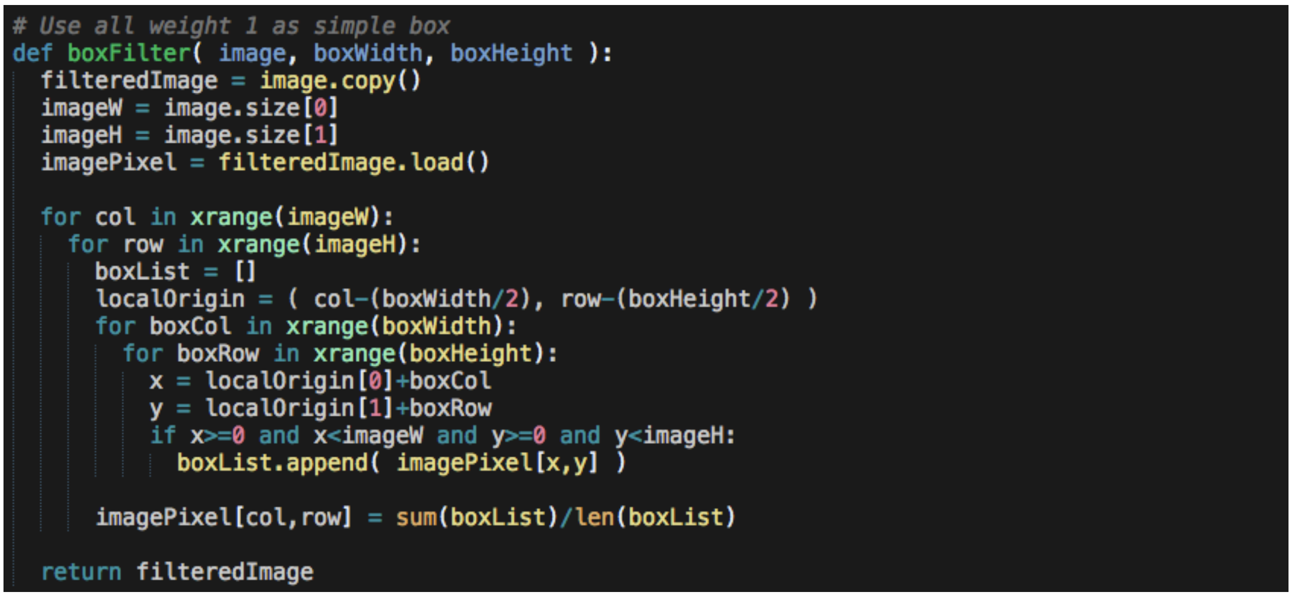
Principal code fragment



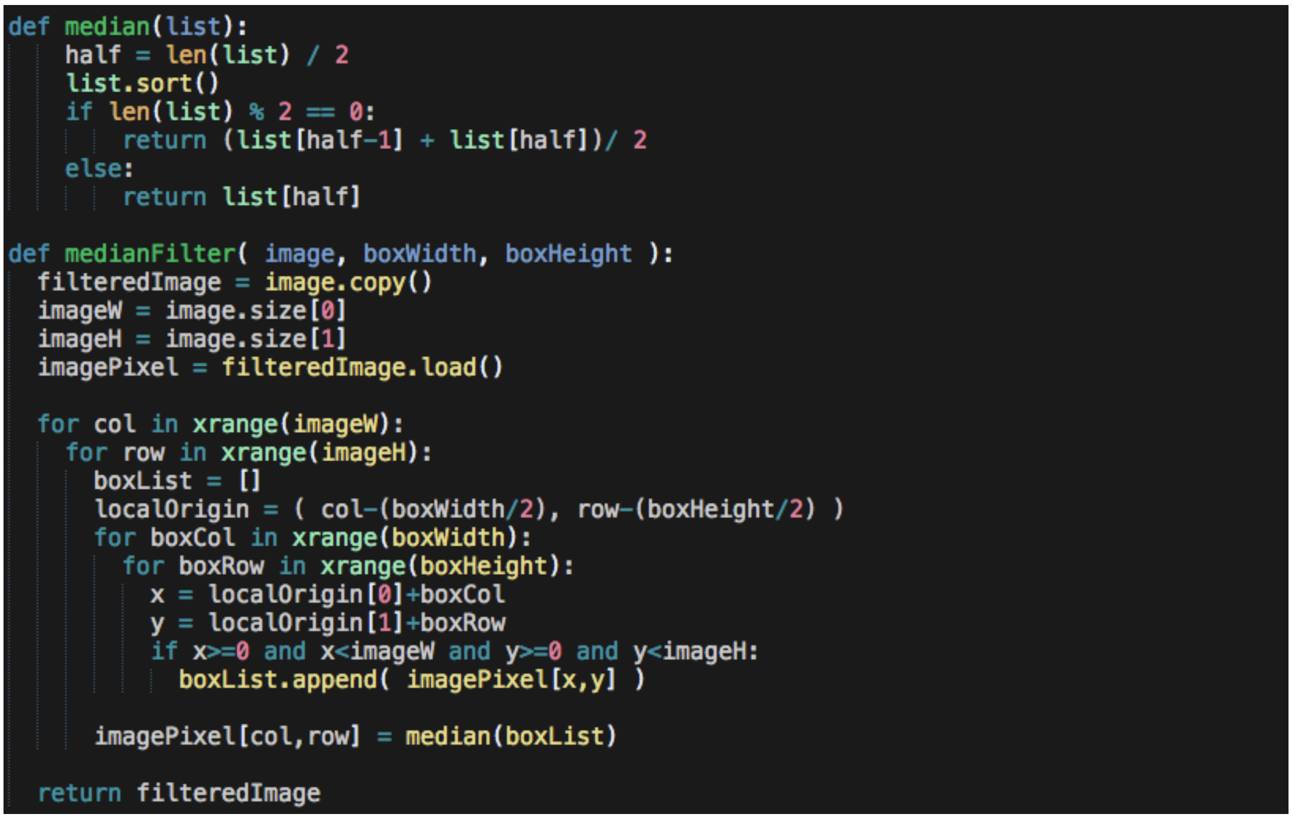
```python
def gaussianNoice( image, amplitude ):
  gaussianImage = image.copy()
  imageW = image.size[0]
  imageH = image.size[1]
  imagePixel = gaussianImage.load()

  for col in xrange(imageW):
    for row in xrange(imageH):
      noiseValue = imagePixel[ col ,row ] + amplitude*random.gauss(0,1)
      if noiseValue > WHITE :
        noiseValue = WHITE

      imagePixel[ col ,row ] = noiseValue

  return gaussianImage
```



```python
def saltAndPepper( image, threshold ):
  saltImage = image.copy()
  imageW = image.size[0]
  imageH = image.size[1]
  imagePixel = saltImage.load()

  for col in xrange(imageW):
    for row in xrange(imageH):
      randomValue = random.uniform(0,1)
      if randomValue < threshold :
        imagePixel[ col,row ] = BLACK
      elif randomValue > 1-threshold :
        imagePixel[ col,row ] = WHITE

  return saltImage
```



```python
# Use all weight 1 as simple box
def boxFilter( image, boxWidth, boxHeight ):
  filteredImage = image.copy()
  imageW = image.size[0]
  imageH = image.size[1]
  imagePixel = filteredImage.load()

  for col in xrange(imageW):
    for row in xrange(imageH):
      boxList = []
      localOrigin = ( col-(boxWidth/2), row-(boxHeight/2) )
      for boxCol in xrange(boxWidth):
        for boxRow in xrange(boxHeight):
          x = localOrigin[0]+boxCol
          y = localOrigin[1]+boxRow
          if x>=0 and x<imageW and y>=0 and y<imageH:
            boxList.append( imagePixel[x,y] )

      imagePixel[col,row] = sum(boxList)/len(boxList)

  return filteredImage
```



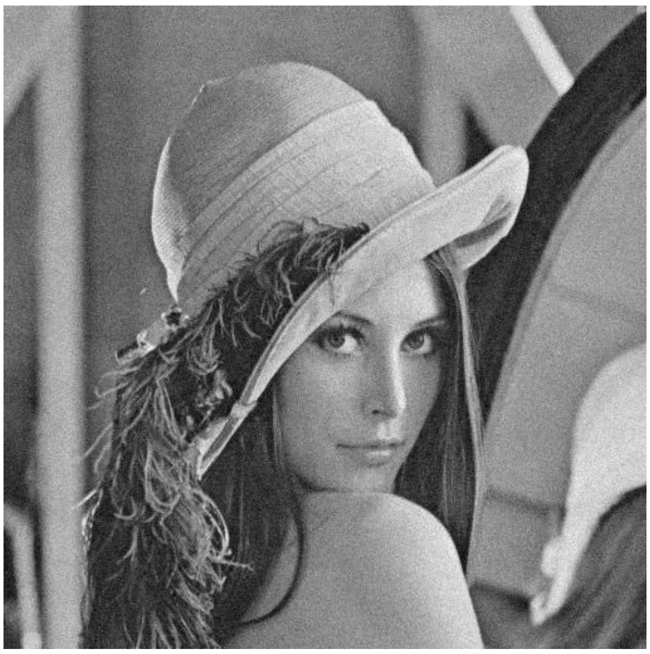
```python
def median(list):
    half = len(list) / 2
    list.sort()
    if len(list) % 2 == 0:
        return (list[half-1] + list[half])/ 2
    else:
        return list[half]

def medianFilter( image, boxWidth, boxHeight ):
  filteredImage = image.copy()
  imageW = image.size[0]
  imageH = image.size[1]
  imagePixel = filteredImage.load()

  for col in xrange(imageW):
    for row in xrange(imageH):
      boxList = []
      localOrigin = ( col-(boxWidth/2), row-(boxHeight/2) )
      for boxCol in xrange(boxWidth):
        for boxRow in xrange(boxHeight):
          x = localOrigin[0]+boxCol
          y = localOrigin[1]+boxRow
          if x>=0 and x<imageW and y>=0 and y<imageH:
            boxList.append( imagePixel[x,y] )

      imagePixel[col,row] = median(boxList)

  return filteredImage
```

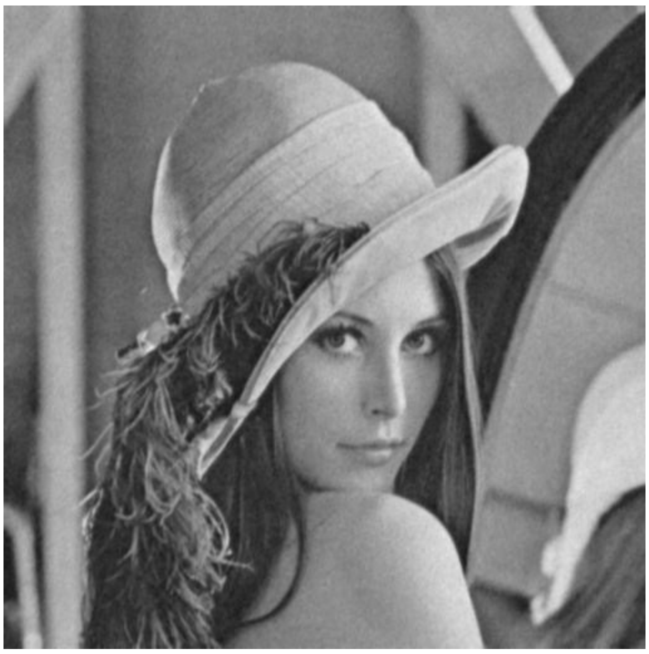
Description

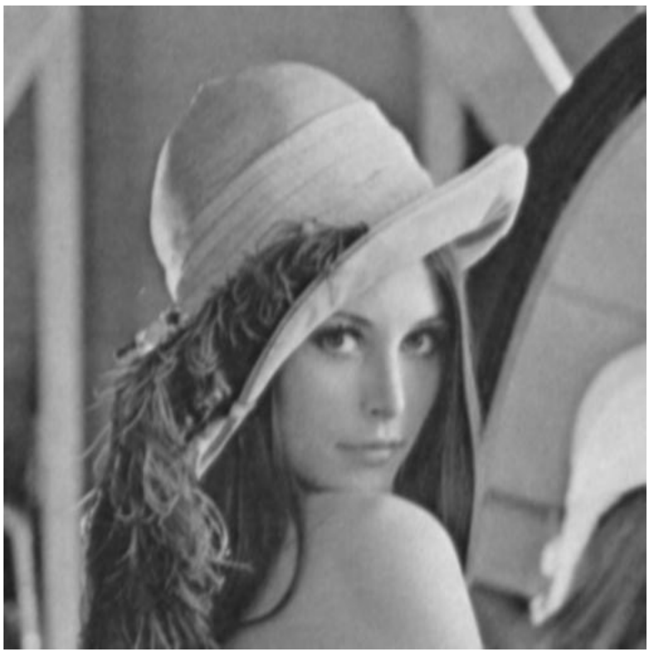
Generate gaussian noise with amplitude of 10 and 30 and salt-and-pepper noise with probability 0.1 and 0.05 first. Then use the 3x3, 5x5 box filter and median filter, opening-then- closing and closing-then opening filter (using the octagonal 3-5-5-5-3 kernel, value = 0) on those images.

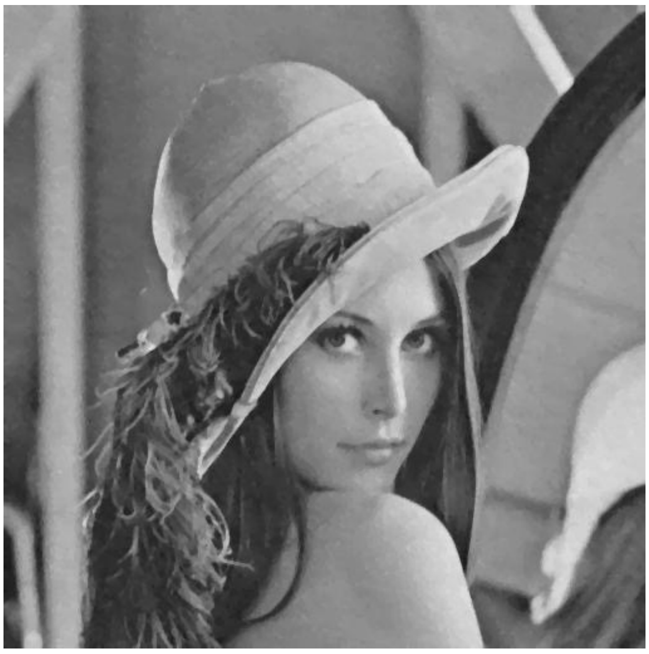
Result

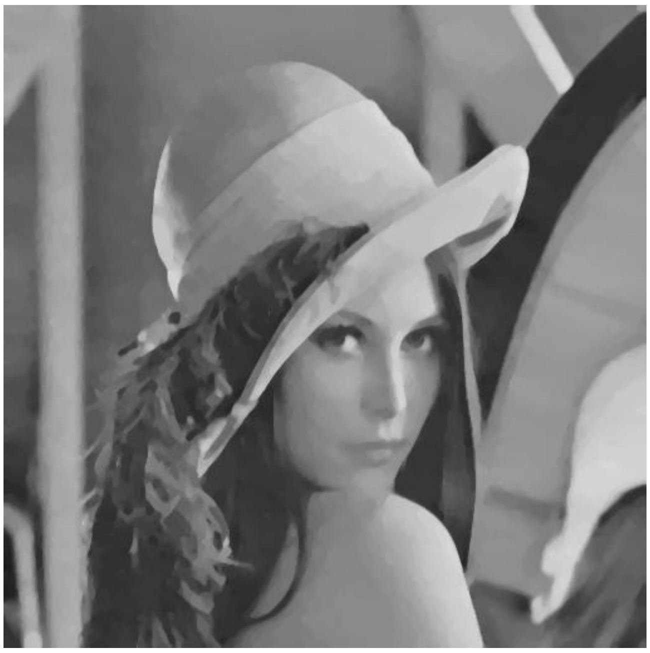
Gaussian Noise (Amplitude=10)

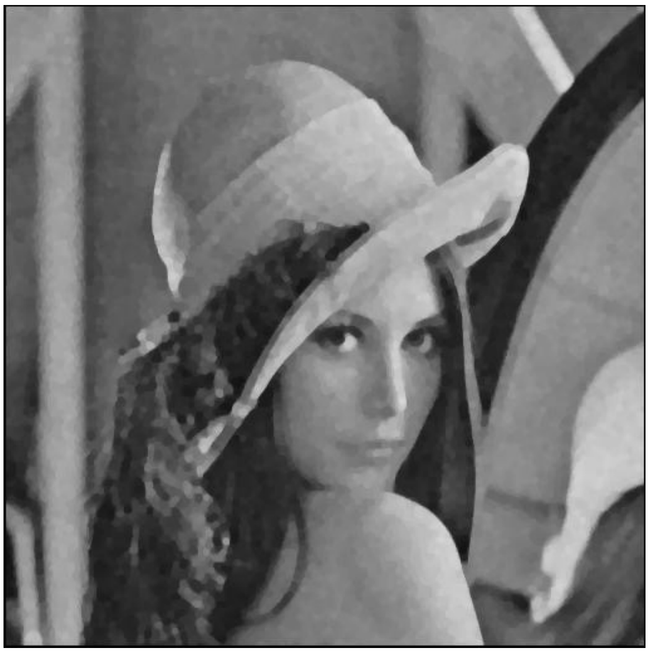

Gaussian Noise Image

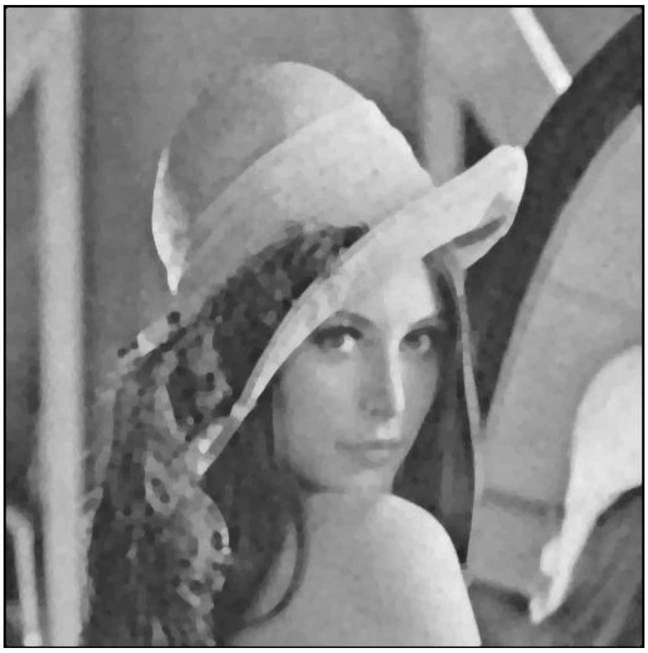

Box Filter 3x3


Box Filter 5x5

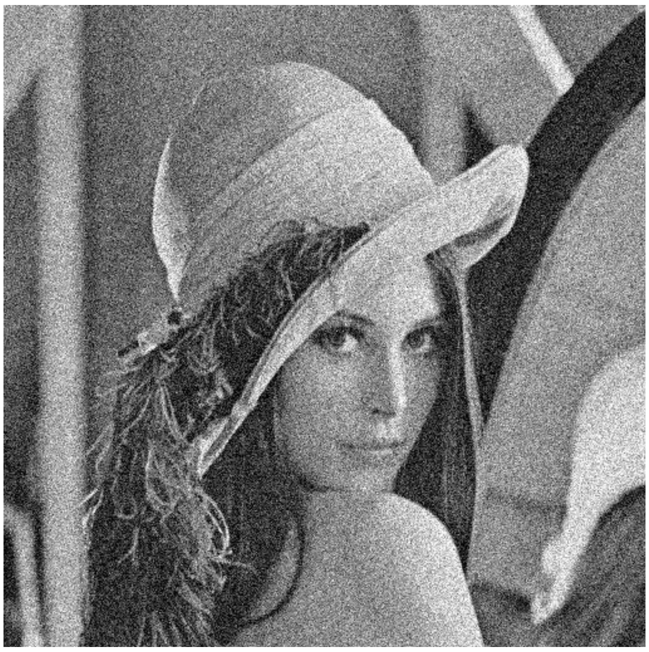

Median Filter 3x3

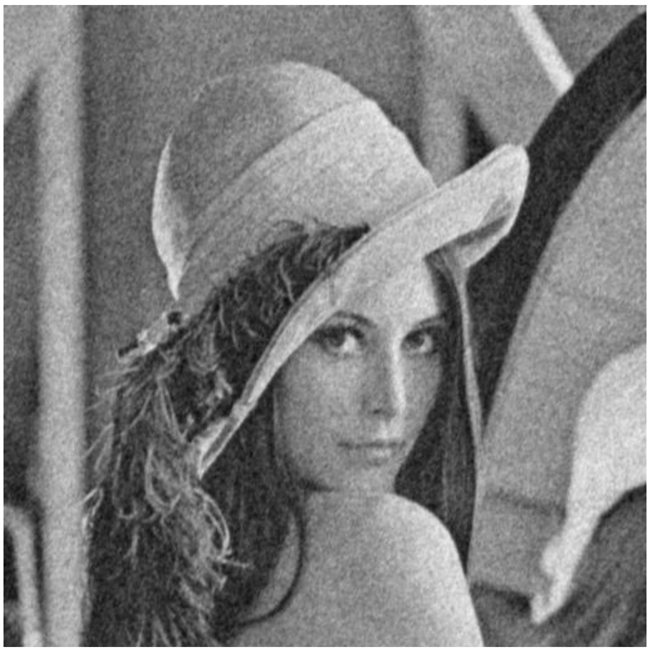

Median Filter 5x5

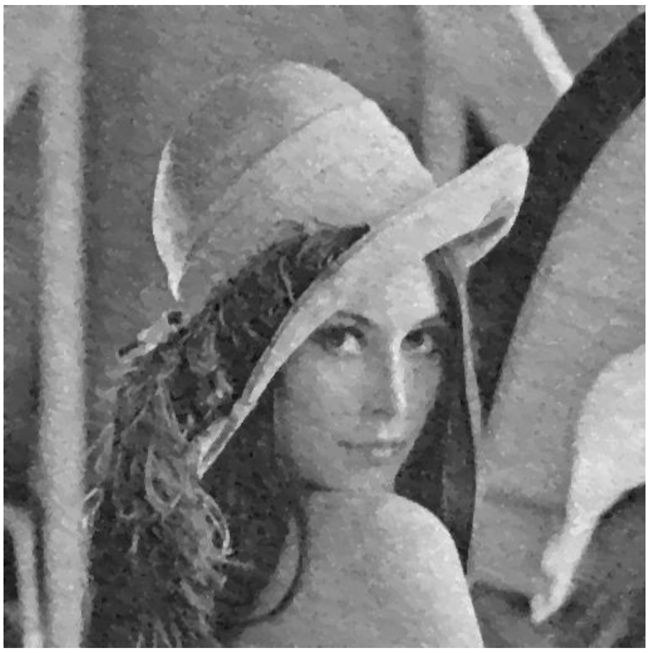

Opening then Closing

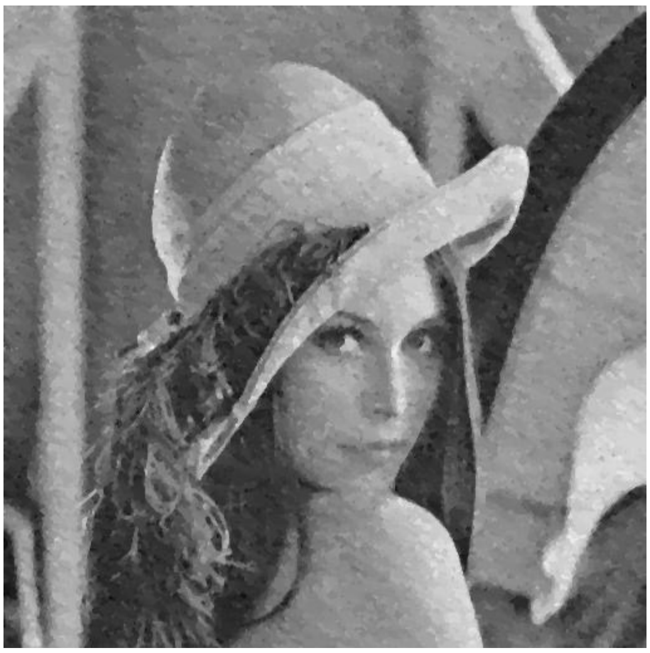

Closing the Opening

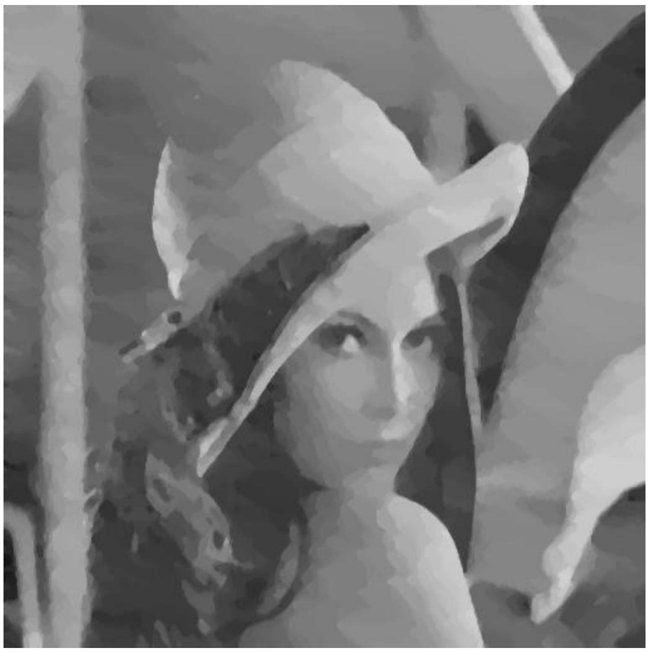
Gaussian Noise (Amplitude=30)

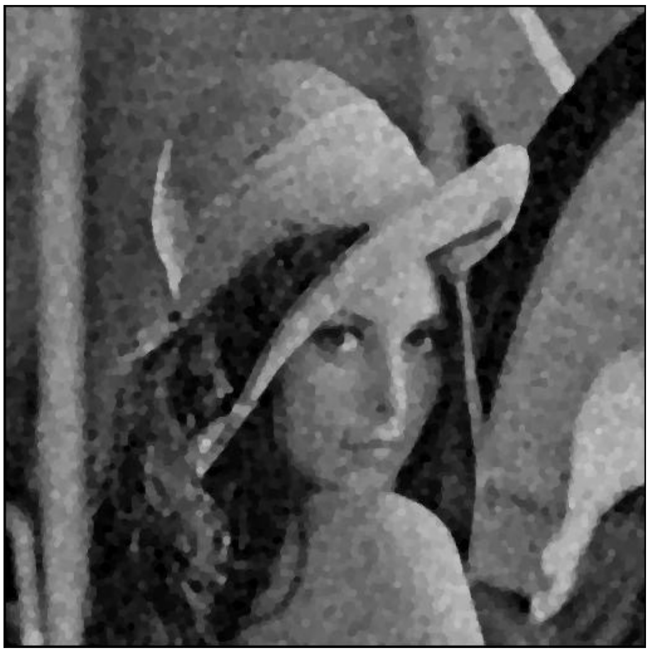

Gaussian Noise Image

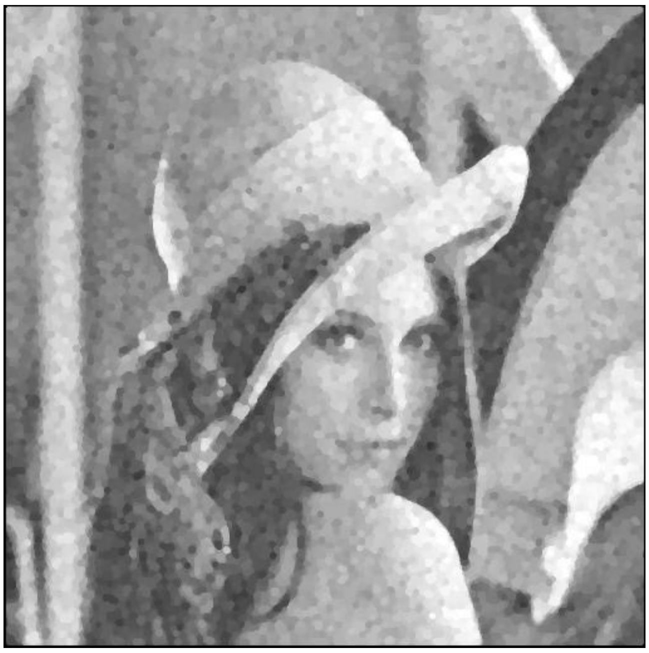

Box Filter 3x3


Box Filter 5x5


Median Filter 3x3

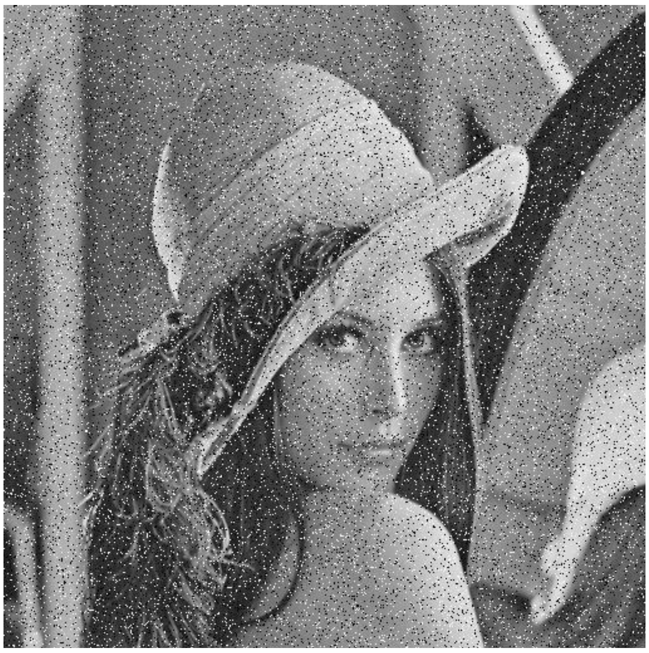

Median Filter 5x5

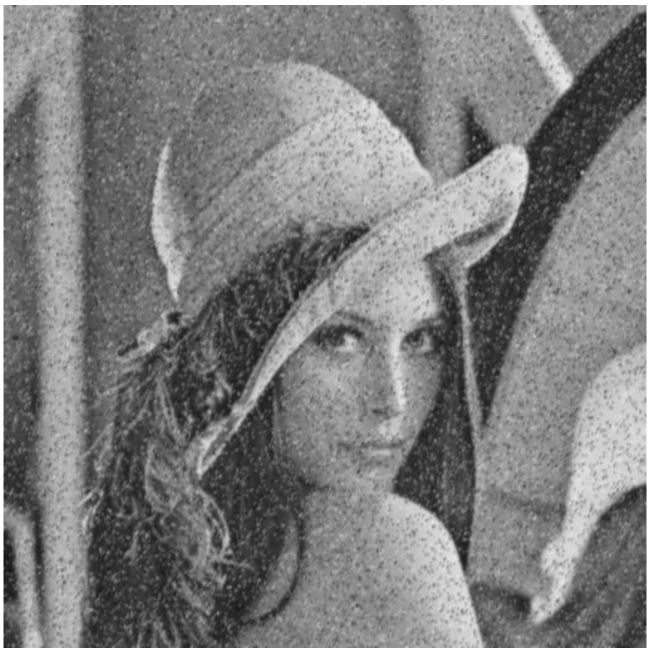

Opening then Closing

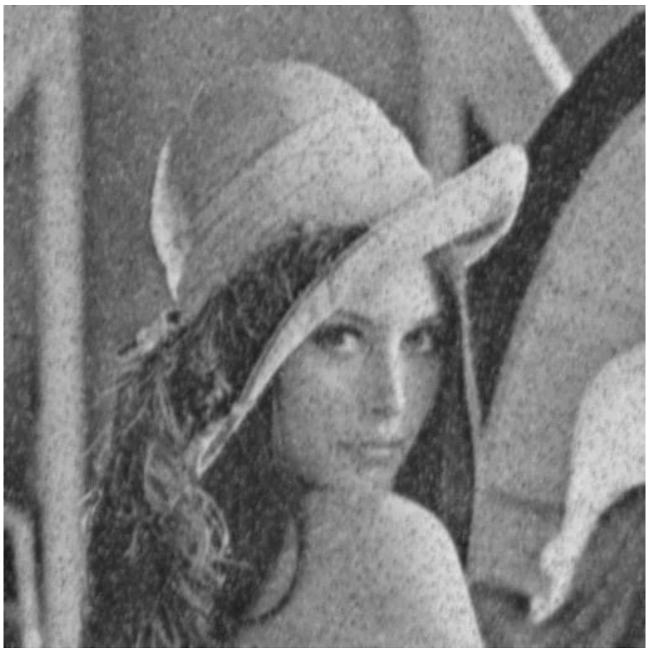

Closing the Opening

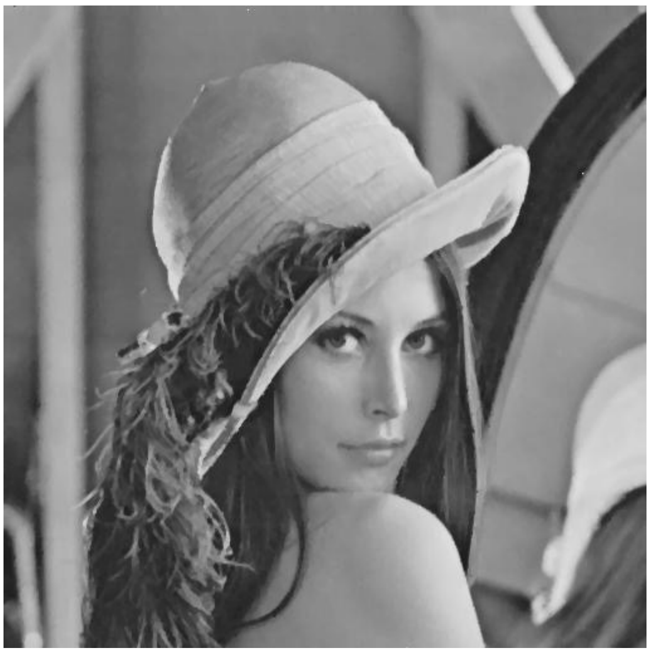
R02944010 李柏緯

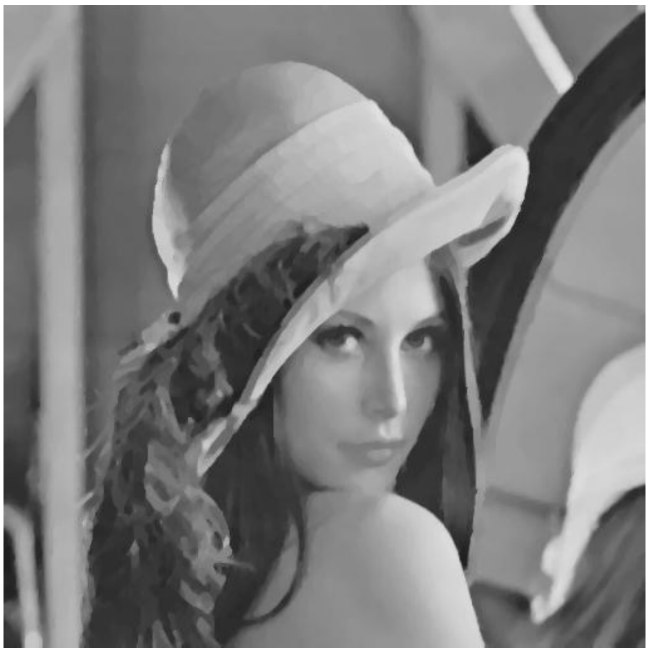
SaltAndPepper (threshold = 0.05)

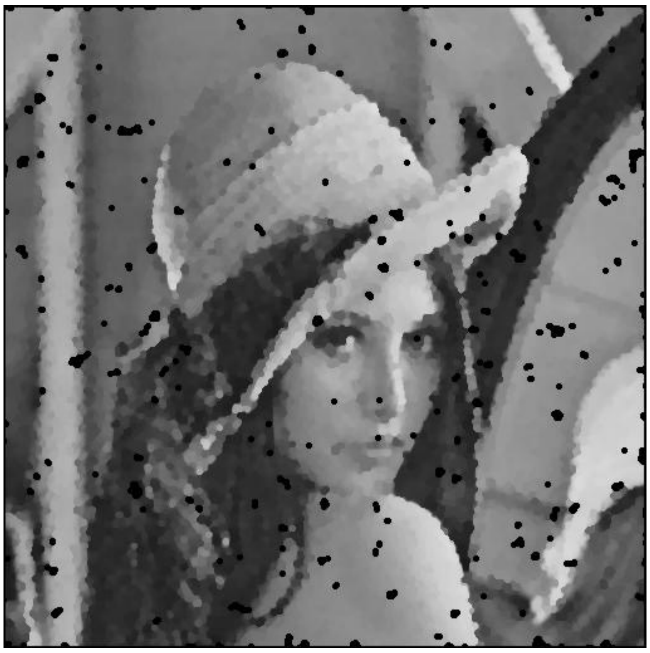

Salt And Peper Noise Image

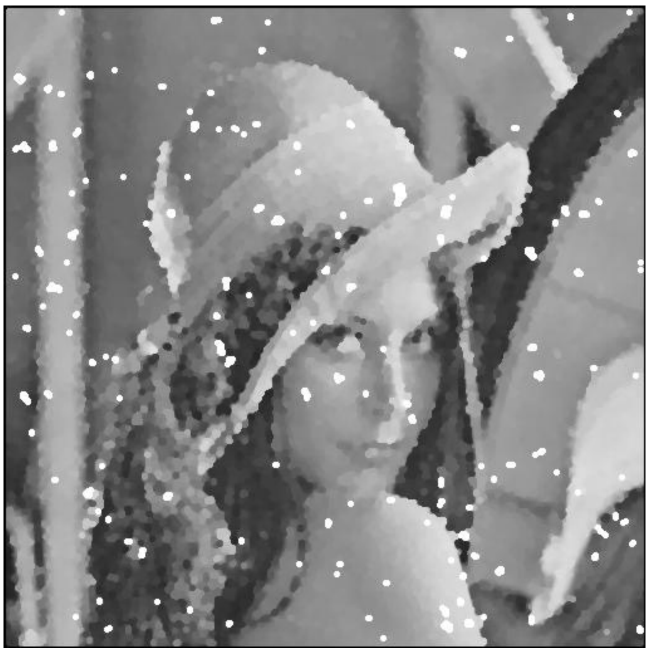

Box Filter 3x3


Box Filter 5x5

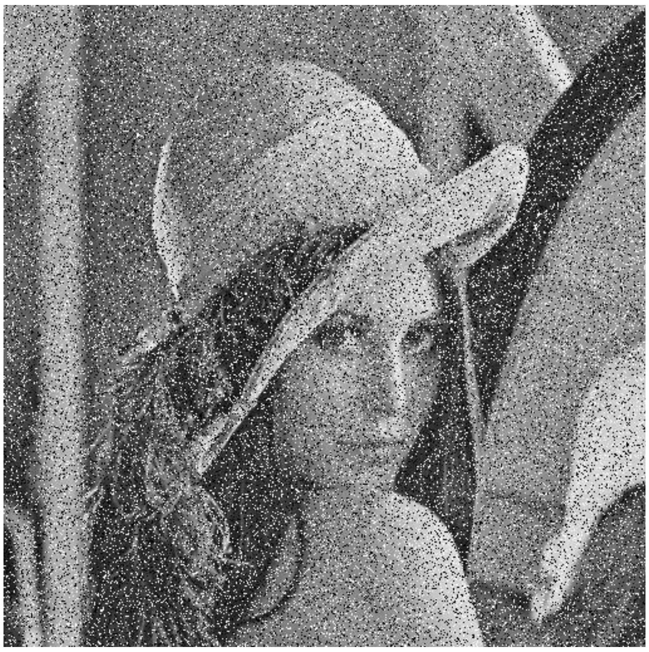

Median Filter 3x3


Median Filter 5x5

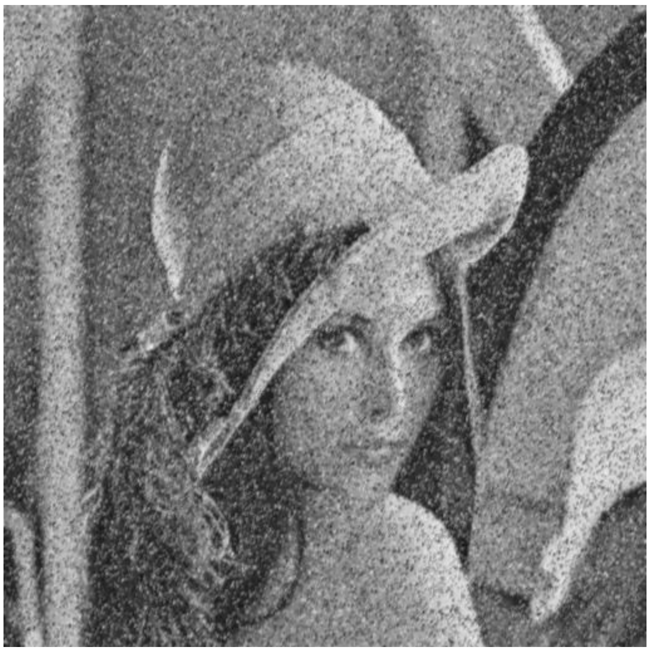

Opening then Closing


Closing the Opening

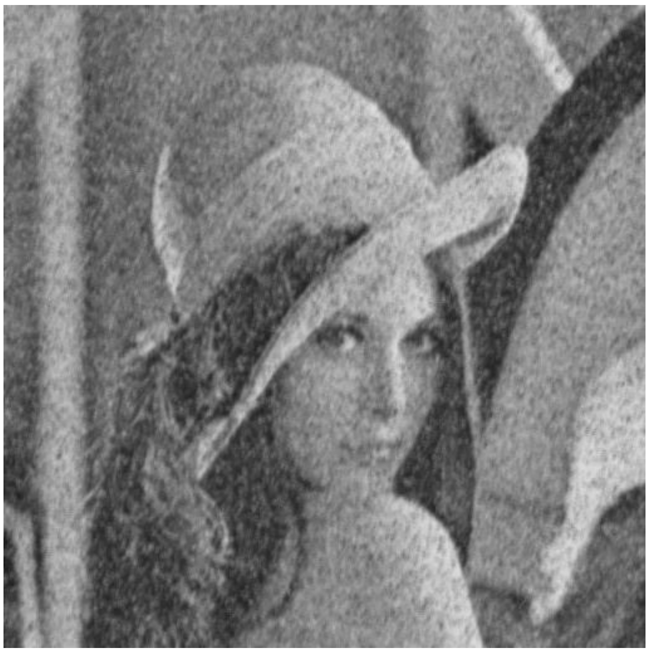
SaltAndPepper (Threshold =0.1)



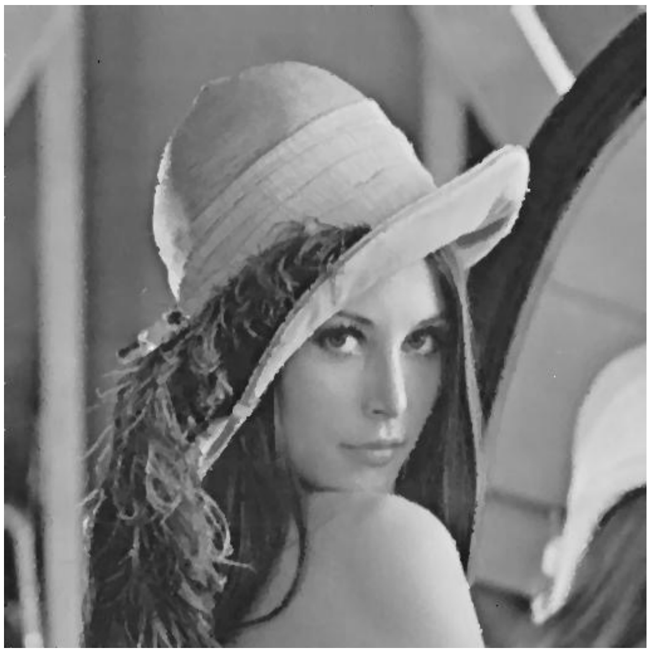
Salt And Peper Noise Image



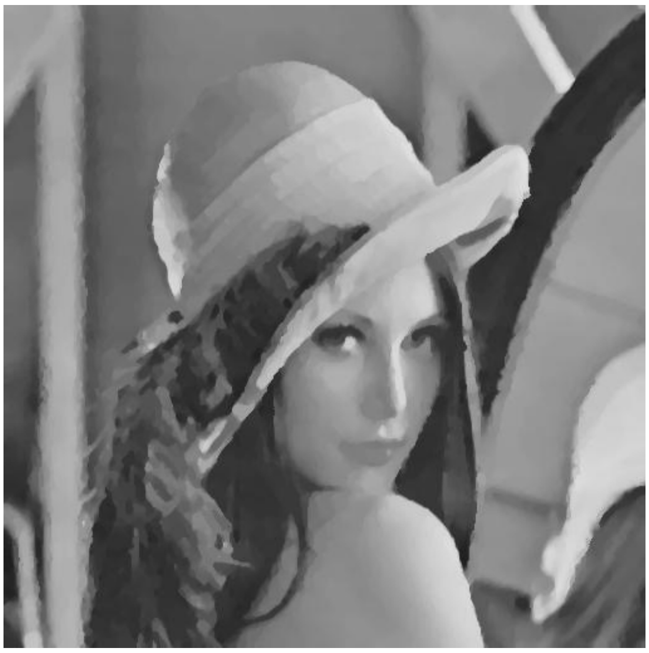
Box Filter 3x3

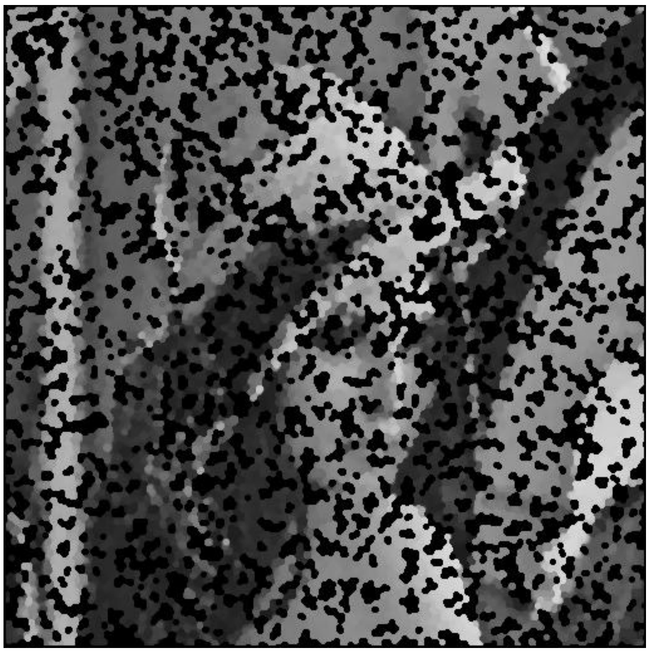


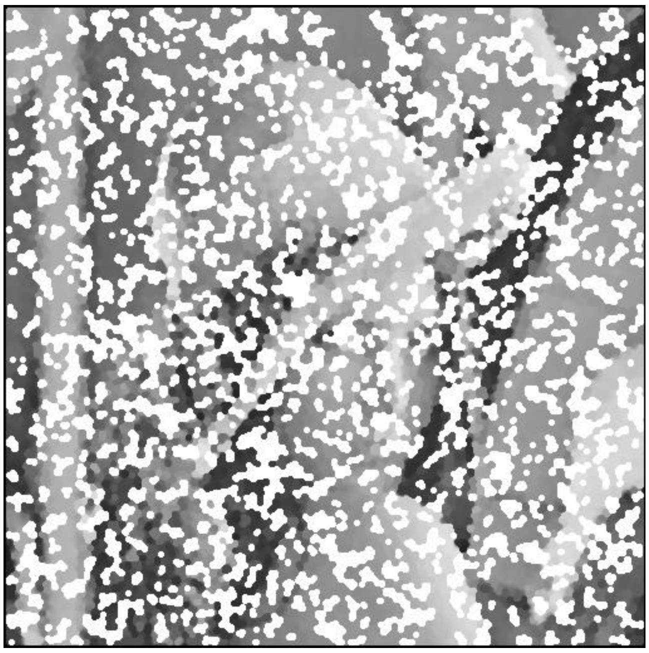
Box Filter 5x5



Median Filter 3x3


Median Filter 5x5


Opening then Closing


Closing the Opening

R02944010 李柏緯

signal-to-ratio

box3x3_gaussian_10: 17.31732322
box3x3_gaussian_30: 12.8043570325
box3x3_salt_005: 9.72821962591
box3x3_salt_01: 6.60503969026

box5x5_gaussian_10: 14.2579558435
box5x5_gaussian_30: 13.0604080344
box5x5_salt_005: 11.0331085902
box5x5_salt_01: 8.60037010007

median3x3_gaussian_10: 17.0803243863
median3x3_gaussian_30: 12.4221762718
median3x3_salt_005: 18.2112960287
median3x3_salt_01: 16.2926425449

median5x5_gaussian_10: 13.4484159886
median5x5_gaussian_30: 8.95821879091
median5x5_salt_005: 14.0309666975
median5x5_salt_01: 12.1838006325

closingThenOpening_gaussian_10: 7.65065256202
closingThenOpening_gaussian_30: 6.07646246604
closingThenOpening_salt_005: 4.26816437106
closingThenOpening_salt_01: -3.09444839174

openingThenClosing_gaussian_10: 8.61005452256
openingThenClosing_gaussian_30: 8.65881476768
openingThenClosing_salt_005: 4.29056873901
openingThenClosing_salt_01: -2.24689641533