CV Home Work 8 資研一 R07922003 劉濬慶

- 將原圖分成兩種不同 Noise with 兩個不同參數:

  1. Gaussian Noise

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

$N(0,1)$:Gaussian random variable with zero mean and st. dev. 1

amplitude determines signal-to-noise ratio, try 10, 30

  2. Salt And Pepper Noise

$$I(nim,i,j) = 0 \;\; if \;\; uniform(0,1) < 0.05$$

$$I(nim,i,j) = 255 \;\; if \;\; uniform(0,1) > 1 - 0.05$$

$$I(nim,i,j) = I(im,i,j) \;\; otherwise$$

$uniform(0,1) : random\ variable\ uniformly\ distributed\ over\ [0,1]$

$try\ both\ 0.05\ and\ 0.1$

- 再將四個 Noise Image 去做四種 Filter

  1. box filter:

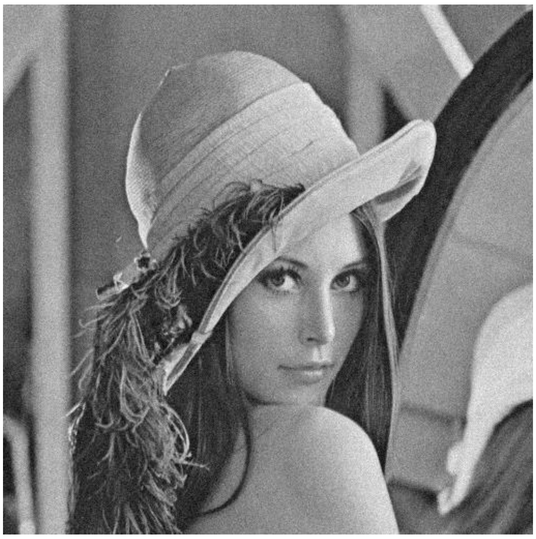
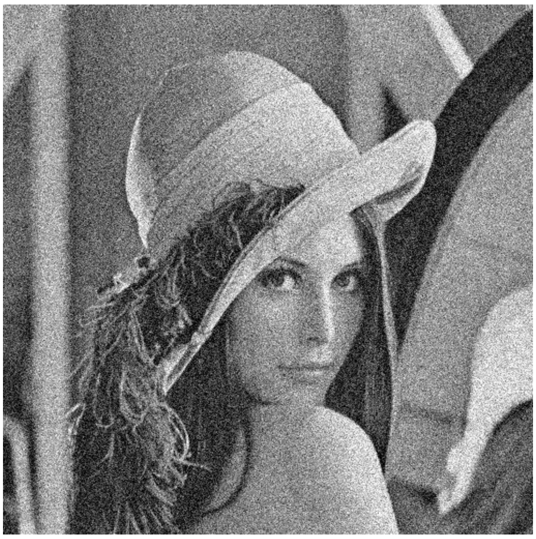
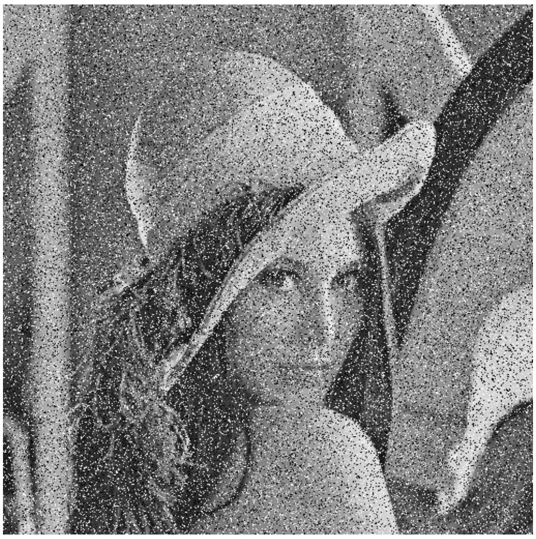
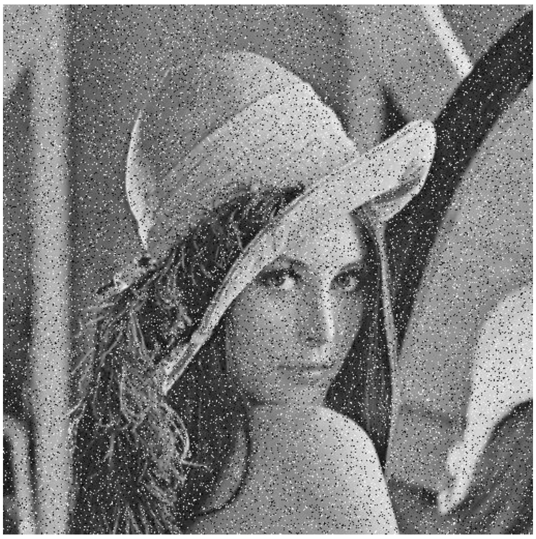
     掃過的那點 pixel，照著那點看成 3x3 的 pixel 矩陣，將那點 pixel 設為 3x3 pixel 矩陣的平均值

  2. median filter:

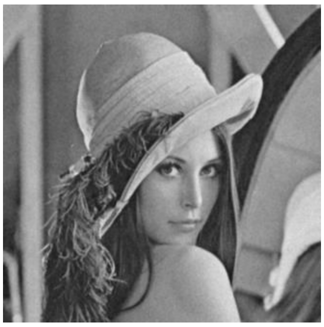
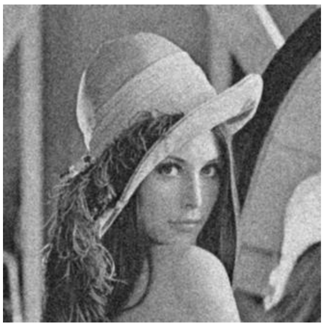
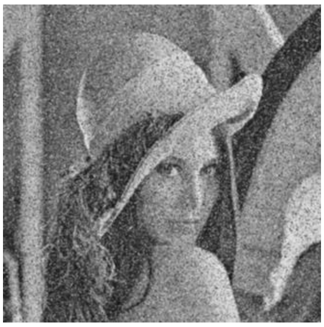
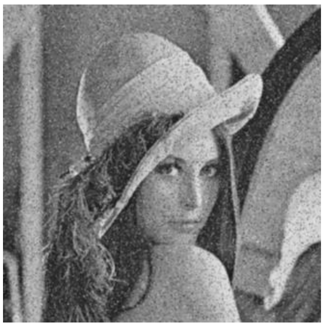
     掃過那點 pixel，照著那點看成 3x3 的 pixel 矩陣，

     將那點 pixel 設為 3x3 pixel 矩陣的中位數

  3. opening-then-closing
  4. closing-then opening

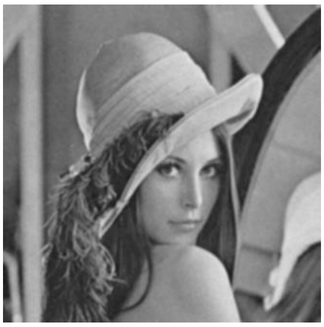
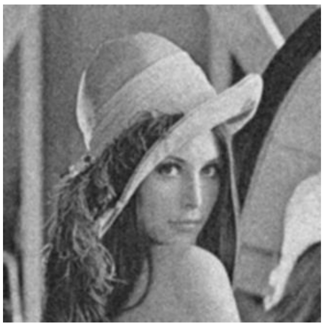
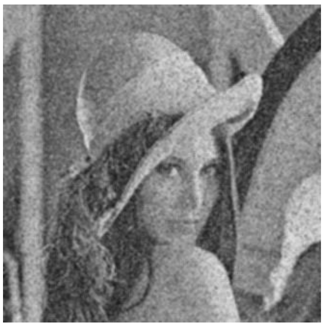
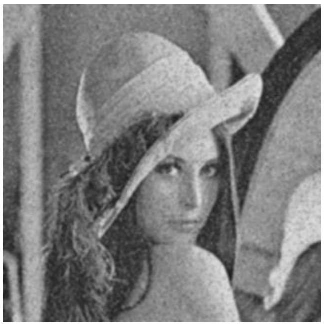
Gaussian_10	Gaussian_30
	
Salt and Pepper 0.1	Salt and Pepper 0.05
	

- Box filter 3x3

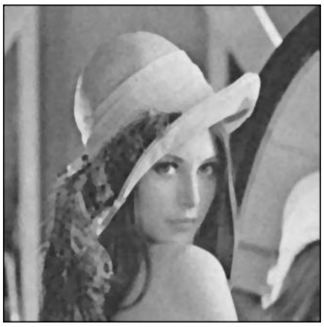
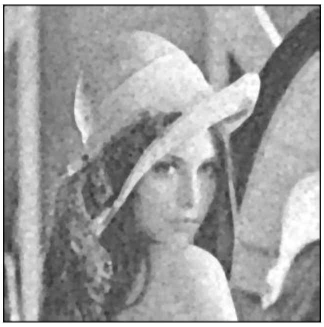
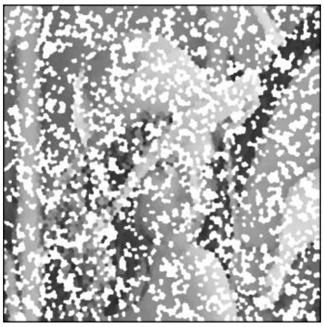
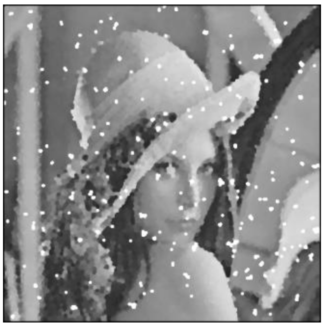
GS 10	GS 30	S&P 0.1	S&P 0.05
			

- Box filter 5x5

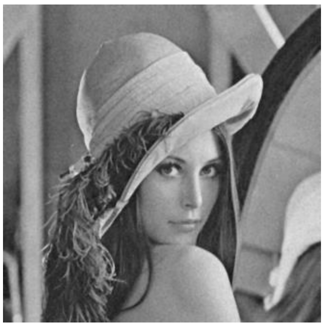
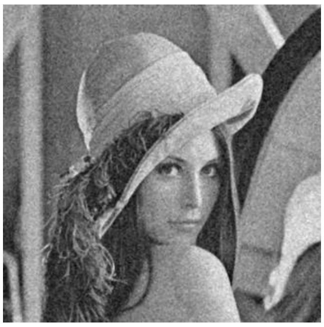
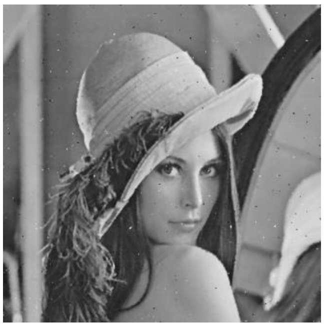
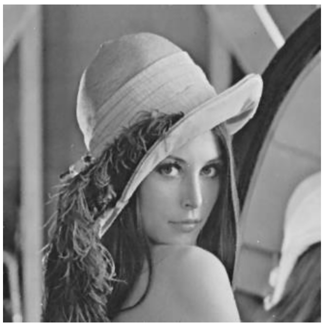
GS 10	GS 30	S&P 0.1	S&P 0.05

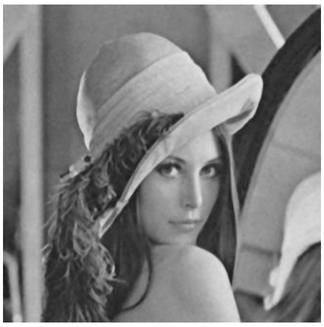
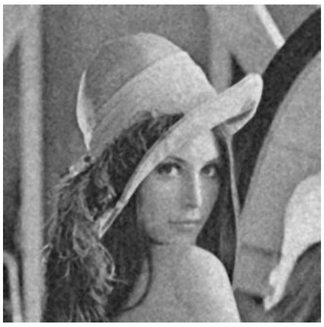
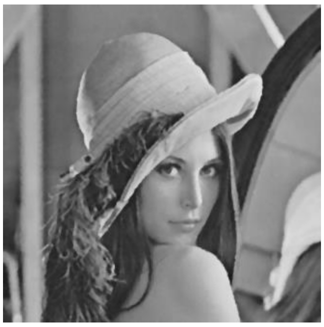
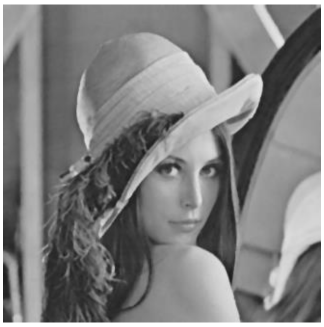
- Close-Open

GS 10	GS 30	S&P 0.1	S&P 0.05
			

- Median 3

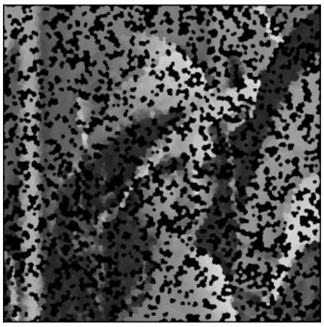
GS 10	GS 30	S&P 0.1	S&P 0.05
			

- Median 5

GS 10	GS 30	S&P 0.1	S&P 0.05
			

- Open-Close

GS 10	GS 30	S&P 0.1	S&P 0.05

SNR

```
GS_10_box_3 ： 13.132161646862954
GS_10_box_5 ： 11.572168146895429
GS_10_median_3 ： 13.27631615310952
GS_10_median_5 ： 12.100599897080322
GS_10_close_open ： 3.6945950492001565
GS_10_open_close ： 4.136362133470649
GS_30_box_3 ： 5.883109108752843
GS_30_box_5 ： 5.358117227626292
GS_30_median_3 ： 5.656054991366601
GS_30_median_5 ： 5.330359200107821
GS_30_close_open ： 2.14640323133101
GS_30_open_close ： 3.012827389645247
S&P_005_box_3 ： 3.5090408586412805
S&P_005_box_5 ： 3.092290603894414
S&P_005_median_3 ： 3.051813938604048
S&P_005_median_5 ： 2.9498254216201203
S&P_005_close_open ： 0.7598616933033864
S&P_005_open_close ： 1.1032935636094314
S&P_01_box_3 ： 1.9505945538037295
S&P_01_box_5 ： 1.592999807303005
S&P_01_median_3 ： 1.4311950233094124
S&P_01_median_5 ： 1.3615386678530694
S&P_01_close_open ： -1.926262016939736
S&P_01_open_close ： -1.0983081749830665
```

程式碼

```
from PIL import Image, ImageDraw
import numpy as np

def Gaussian_noise(img, amp):
    pixel = img.load()
    Gaussian_noise_img = Image.new(img.mode, img.size)

    for i in range(0,512,1):
        for j in range(0,512,1):
            Gaussian_noputpixelise_img.((i,j),int( pixel[i,j] + amp *
np.random.normal(0,1)) )

    return Gaussian_noise_img

def salt_and_pepper_noise(img, threshold):
    pixel = img.load()
    salt_and_pepper_noise_img = Image.new(img.mode, img.size)

    for i in range(0,512,1):
        for j in range(0,512,1):
            rand = np.random.sample()
            if rand < threshold:
                salt_and_pepper_noise_img.putpixel((i,j),0)
            elif rand > 1 - threshold:
                salt_and_pepper_noise_img.putpixel((i,j),255)
            else:
                salt_and_pepper_noise_img.putpixel((i,j),pixel[i,j])
    return salt_and_pepper_noise_img

def box_and_median_filter(img, box_size):
    pixel = img.load()
    img_box = Image.new(img.mode, img.size)
    img_median = Image.new(img.mode, img.size)

    x_start = int(box_size/2)
    y_start = int(box_size/2)
    for i in range(0,512,1):
        for j in range(0,512,1):
            box_collect = []
            for x in range(0,box_size,1):
                for y in range(0,box_size,1):
                    try:
                        box_collect.append( pixel[i+x-x_start, j+y-y_start] )
                    except:
                        pass
            img_box.putpixel((i,j) , int(np.mean(np.array(box_collect))) )
            img_median.putpixel((i,j) ,int( np.median(np.array(box_collect))) )

    return img_box, img_median
```

```
def dilation(img, kernel):
    pixel = img.load()
    coulmn,row=img.size
    img_new = Image.new(img.mode, img.size)

    for i in range(0,coulmn,1):
        for j in range(0,row,1):
            if pixel[i,j] > 0:
                dil_pix_list = []
                for y in range(-2,3,1):
                    for x in range(-2,3,1):
                        if kernel[y+2,x+2] == 1:
                            if (i+x < coulmn) and (j+y < row) and (i+x >= 0) and (j+y >=
0):
                                dil_pix_list.append(pixel[i+x,j+y])

                max_pix = max(dil_pix_list)
                for y in range(-2,3,1):
                    for x in range(-2,3,1):
                        if kernel[y+2,x+2] == 1:
                            if (i+x < coulmn) and (j+y < row) and (i+x >= 0) and (j+y >=
0):
                                img_new.putpixel((i+x,j+y),max_pix)
    return img_new

def erosion(img, kernel):
    pixel = img.load()
    coulmn,row=img.size
    img_new = Image.new(img.mode, img.size )
    for i in range(0,coulmn,1):
        for j in range(0,row,1):
            ero_flag = True
            ero_pix_list = []
            for y in range(-2,3,1):
                for x in range(-2,3,1):
                    if kernel[y+2,x+2] == 1:
                        if (i+x < coulmn) and (j+y < row) and (i+x >= 0) and (j+y >= 0):
                            ero_pix_list.append(pixel[i+x,j+y])
                            if pixel[i+x,j+y] == 0:
                                ero_flag = False
                        else:
                            ero_flag = False
            min_pix = min(ero_pix_list)
            if ero_flag :
                img_new.putpixel((i,j),min_pix)

    return img_new

def opening(img, kernel):
    img_ero = erosion(img, kernel)
    img_new = dilation(img_ero, kernel)

    return img_new

def closing(img, kernel):
    img_dil = dilation(img, kernel)
    img_new = erosion(img_dil, kernel)

    return img_new

def SNR_calculate(img_orig, img_proc):
    pixel_orig = img_orig.load()
    pixel_proc = img_proc.load()

    orig_array = np.array((512,512))
    proc_array = np.array((512,512))
    mu = 0
    mu_n = 0
    VS = 0
    VN = 0
    for i in range(0,512,1):
        for j in range(0,512,1):
            mu += pixel_orig[i,j]
            mu_n += pixel_proc[i,j] - pixel_orig[i,j]

    mu = mu / (512*512)
    mu_n = mu_n / (512*512)

    for i in range(0,512,1):
        for j in range(0,512,1):
            VS += (pixel_orig[i,j] - mu) ** 2
            VN += (pixel_proc[i,j] - pixel_orig[i,j] - mu_n) ** 2
    VS = VS / (512*512)
    VN = VN / (512*512)

    SNR = 20 * np.log10( np.sqrt(VS) / np.sqrt(VN) )
    return SNR

def image_processing(img, file_name):
    img_box_3 , img_median_3 = box_and_median_filter(img, 3)
    img_box_5 , img_median_5 = box_and_median_filter(img, 5)

    kernel_array = np.array([[0,1,1,1,0],
                                     [1,1,1,1,1],
                                     [1,1,1,1,1],
                                     [1,1,1,1,1],
                                     [0,1,1,1,0]])
    img_open = opening(img, kernel_array)
    img_close = closing(img, kernel_array)
    img_close_open = opening(img_close, kernel_array)
    img_open_close = closing(img_open, kernel_array)

    img_box_3.save('./processed/' + file_name + '_box_3.bmp')
    img_box_5.save('./processed/' + file_name + '_box_5.bmp')
    img_median_3.save('./processed/' + file_name + '_median_3.bmp')
    img_median_5.save('./processed/' + file_name + '_median_5.bmp')
    img_close_open.save('./processed/' + file_name + '_close_open.bmp')
    img_open_close.save('./processed/' + file_name + '_open_close.bmp')

    print ( file_name + "_box_3 : " + str( SNR_calculate(img, img_box_3) ))
    print ( file_name + "_box_5 : " + str( SNR_calculate(img, img_box_5) ))
    print ( file_name + "_median_3 : " + str( SNR_calculate(img, img_median_3) ))
    print ( file_name + "_median_5 : " + str( SNR_calculate(img, img_median_5) ))
    print ( file_name + "_close_open : " + str( SNR_calculate(img, img_close_open) ))
    print ( file_name + "_open_close : " + str( SNR_calculate(img, img_open_close) ))


lena=Image.open("lena.bmp")

img_noise = Gaussian_noise(lena, 10)
img_noise.save('./noised/Gaussian_10.bmp')
image_processing(img_noise, 'GS_10')

img_noise = Gaussian_noise(lena, 30)
```

```
img_noise.save('./noised/Gaussian_30.bmp')
image_processing(img_noise, 'GS_30')

img_noise = salt_and_pepper_noise(lena, 0.05)
img_noise.save('./noised/Salt_and_Pepper_005.bmp')
image_processing(img_noise, 'S&P_005')

img_noise = salt_and_pepper_noise(lena, 0.1)
img_noise.save('./noised/Salt_and_Pepper_01.bmp')
image_processing(img_noise, 'S&P_01')
```