Computer and Robot Vision Homework 8
Noise Removal
資工碩一 張家源 R07922102

使用語言： **Python**

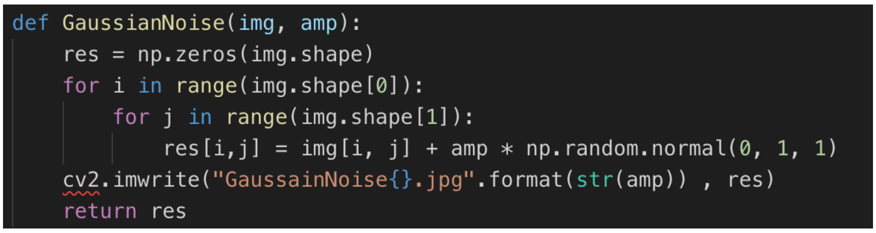
Gaussian Noise

根據下列公式，更新每個Pixel。

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

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

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



```python
def GaussianNoise(img, amp):
    res = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            res[i,j] = img[i, j] + amp * np.random.normal(0, 1, 1)
    cv2.imwrite("GaussainNoise{}.jpg".format(str(amp)) , res)
    return res
```

Salt-And-Pepper Noise

根據下列公式，更新每個Pixel。

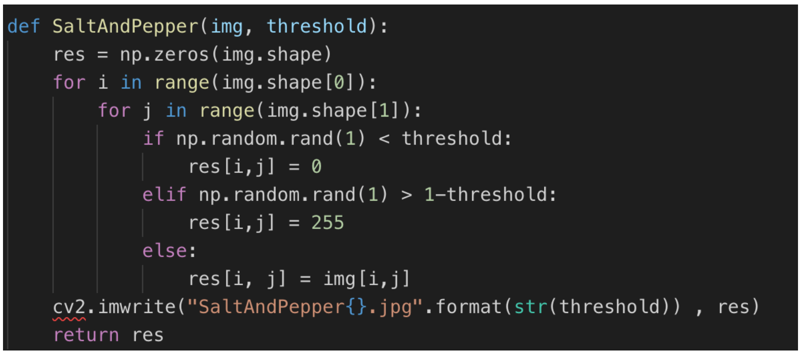
$$I(nim,i,j) = 0 \quad if\ uniform(0,1) < 0.05$$

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

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

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

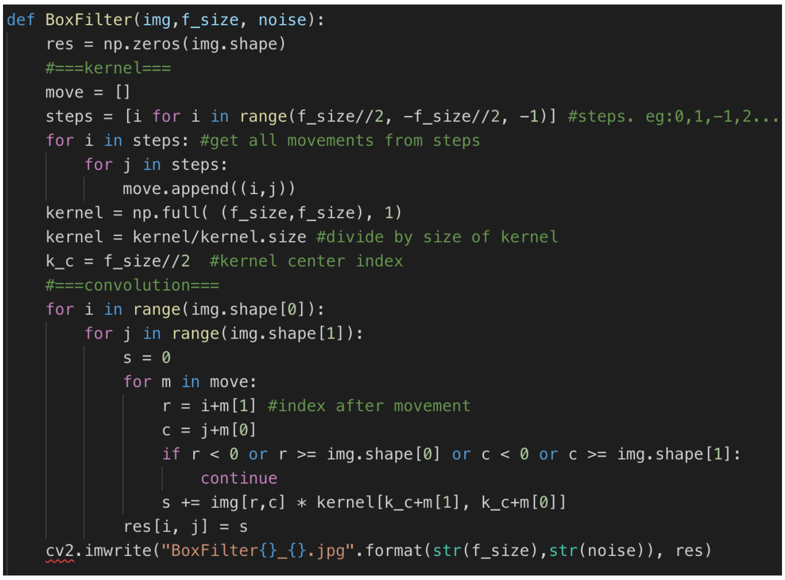
$$try\ both\ 0.05\ and\ 0.1$$



```python
def SaltAndPepper(img, threshold):
    res = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if np.random.rand(1) < threshold:
                res[i,j] = 0
            elif np.random.rand(1) > 1-threshold:
                res[i,j] = 255
            else:
                res[i, j] = img[i,j]
    cv2.imwrite("SaltAndPepper{}.jpg".format(str(threshold)) , res)
    return res
```

Box Filter

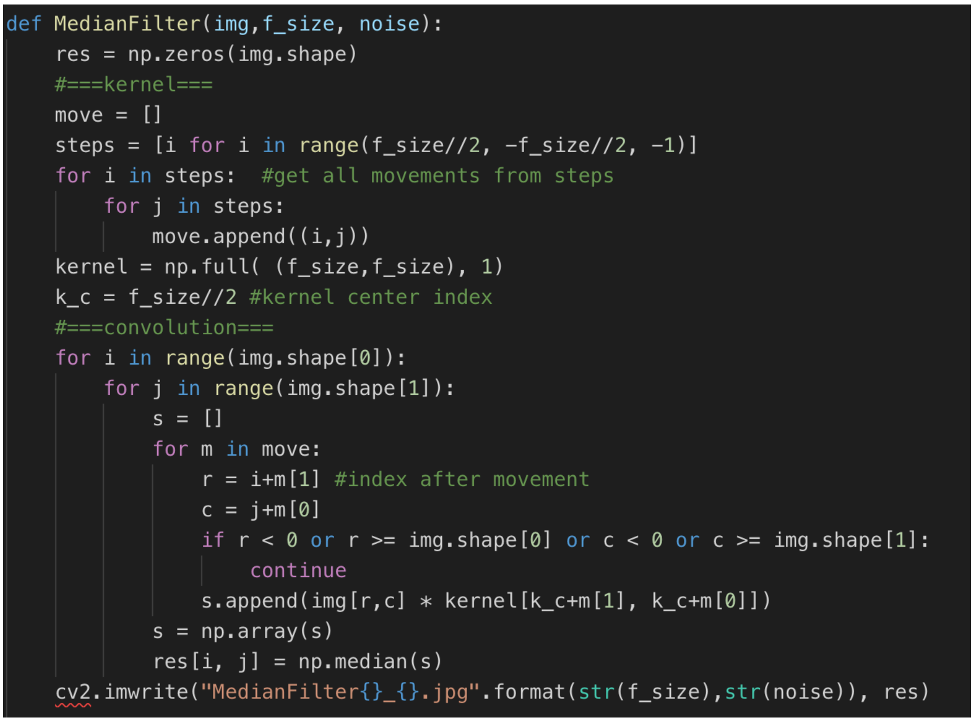
以mask範圍內的平均值更新Pixel。



```python
def BoxFilter(img,f_size, noise):
    res = np.zeros(img.shape)
    #===kernel===
    move = []
    steps = [i for i in range(f_size//2, -f_size//2, -1)] #steps. eg:0,1,-1,2...
    for i in steps: #get all movements from steps
        for j in steps:
            move.append((i,j))
    kernel = np.full( (f_size,f_size), 1)
    kernel = kernel/kernel.size #divide by size of kernel
    k_c = f_size//2  #kernel center index
    #===convolution===
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            s = 0
            for m in move:
                r = i+m[1] #index after movement
                c = j+m[0]
                if r < 0 or r >= img.shape[0] or c < 0 or c >= img.shape[1]:
                    continue
                s += img[r,c] * kernel[k_c+m[1], k_c+m[0]]
            res[i, j] = s
    cv2.imwrite("BoxFilter{}_{}.jpg".format(str(f_size),str(noise)), res)
```

Median Filter

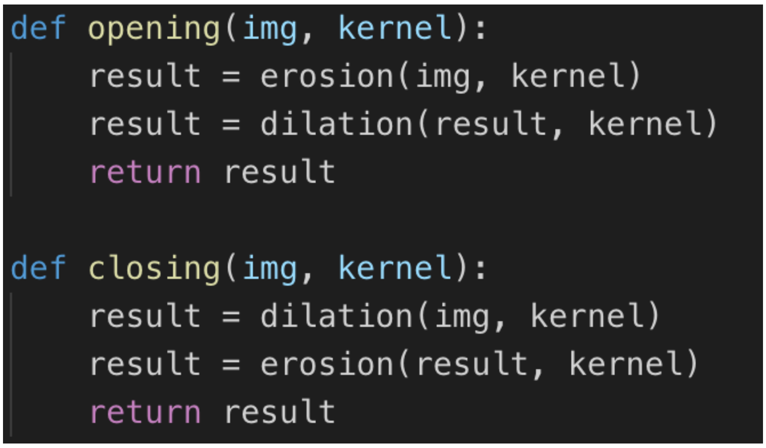
以mask範圍內的中位數來更新Pixel。



```python
def MedianFilter(img,f_size, noise):
    res = np.zeros(img.shape)
    #===kernel===
    move = []
    steps = [i for i in range(f_size//2, -f_size//2, -1)]
    for i in steps:  #get all movements from steps
        for j in steps:
            move.append((i,j))
    kernel = np.full( (f_size,f_size), 1)
    k_c = f_size//2 #kernel center index
    #===convolution===
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            s = []
            for m in move:
                r = i+m[1] #index after movement
                c = j+m[0]
                if r < 0 or r >= img.shape[0] or c < 0 or c >= img.shape[1]:
                    continue
                s.append(img[r,c] * kernel[k_c+m[1], k_c+m[0]])
            s = np.array(s)
            res[i, j] = np.median(s)
    cv2.imwrite("MedianFilter{}_{}.jpg".format(str(f_size),str(noise)), res)
```

Opening and Closing

根據HW5的Dilation & Erosion實作Opening and Closing。



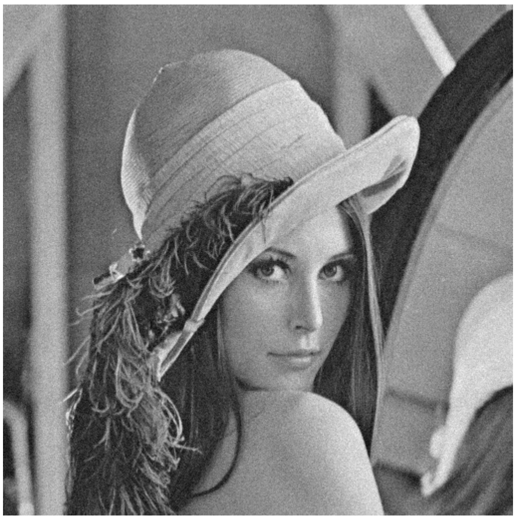
```python
def opening(img, kernel):
    result = erosion(img, kernel)
    result = dilation(result, kernel)
    return result

def closing(img, kernel):
    result = dilation(img, kernel)
    result = erosion(result, kernel)
    return result
```

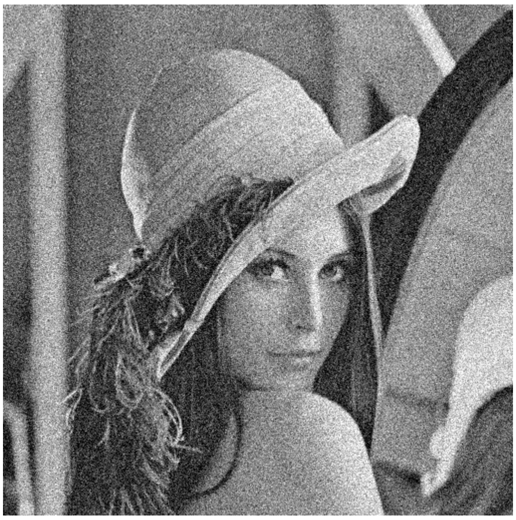
Result

Noise

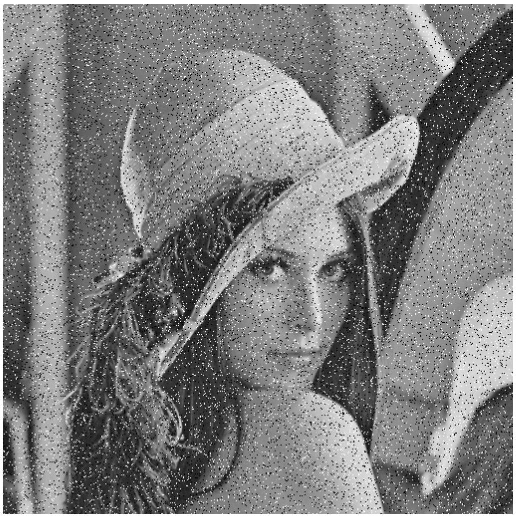
Gaussian Noise with amplitude 10



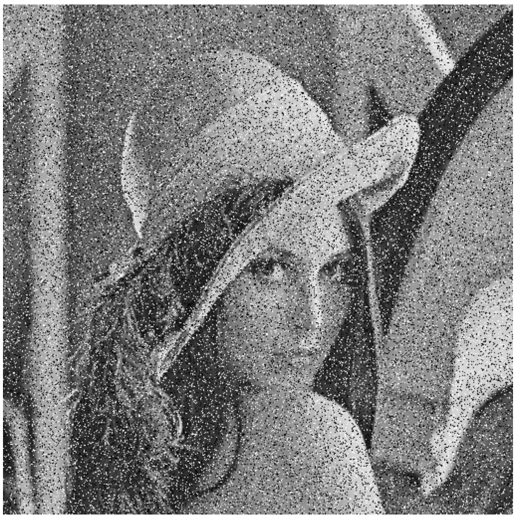
Gaussian Noise with amplitude 30



Salt-And-Pepper with threshold 0.05

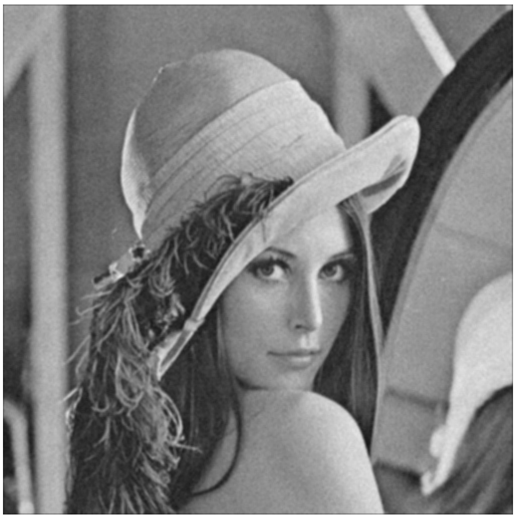


Salt-And-Pepper with threshold 0.1

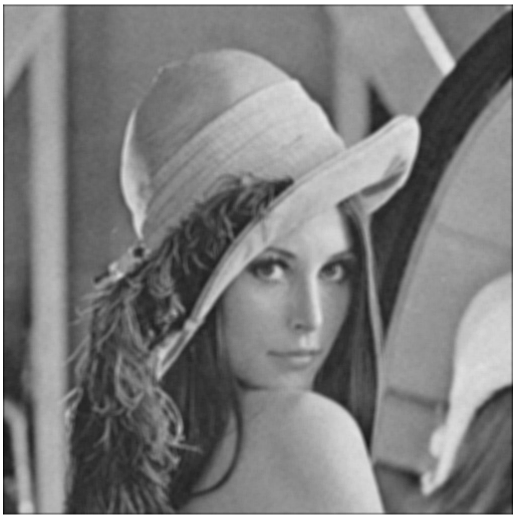


Gaussian Noise with amplitude 10

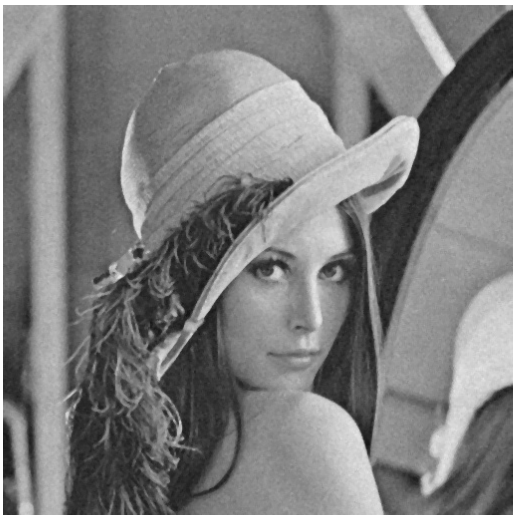
3x3 Box filter



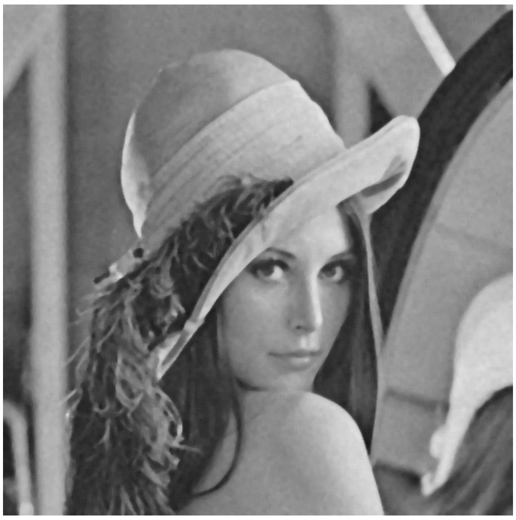
5x5 Box filter



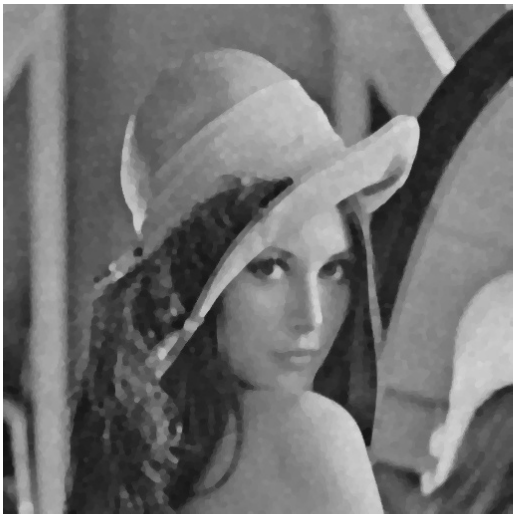
3x3 Median filter



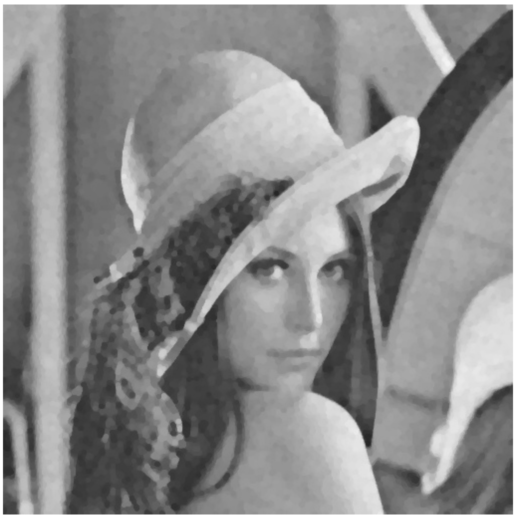
5x5 Median filter



Opening then Closing

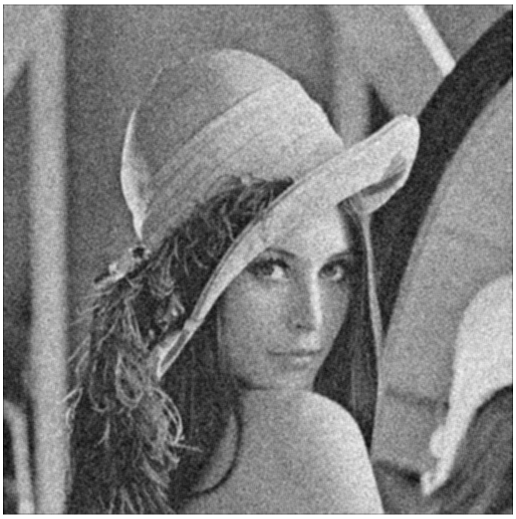


Closing then Opening

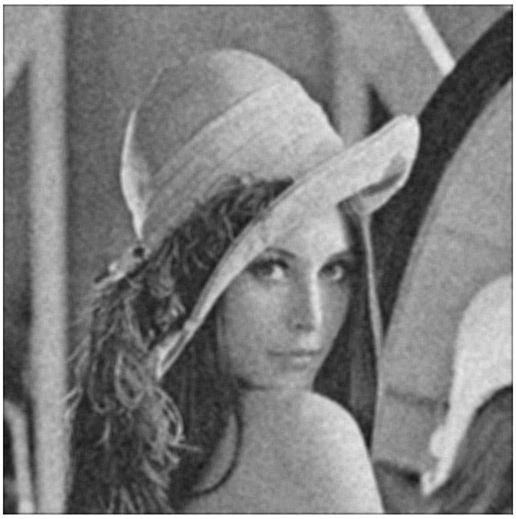


Gaussian Noise with amplitude 30

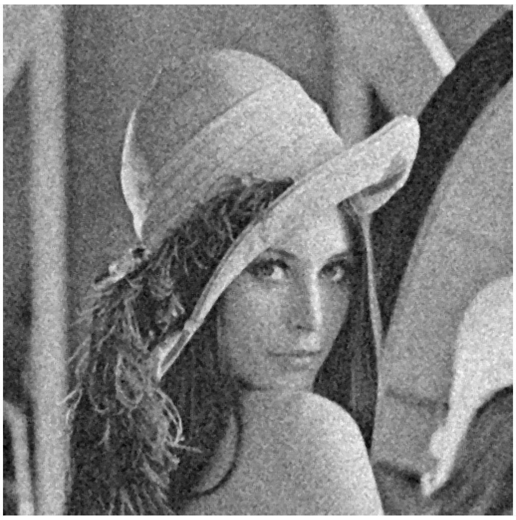
3x3 Box filter



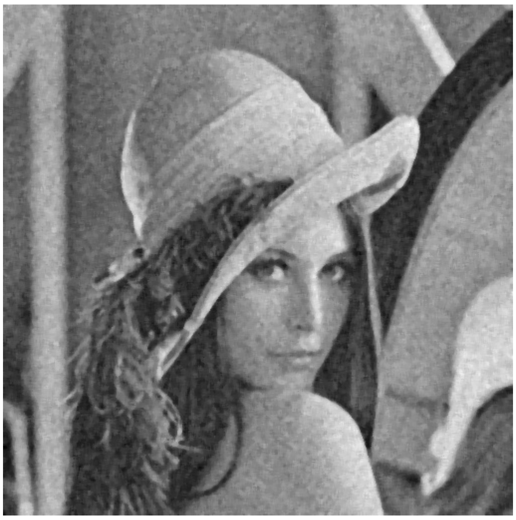
5x5 Box filter



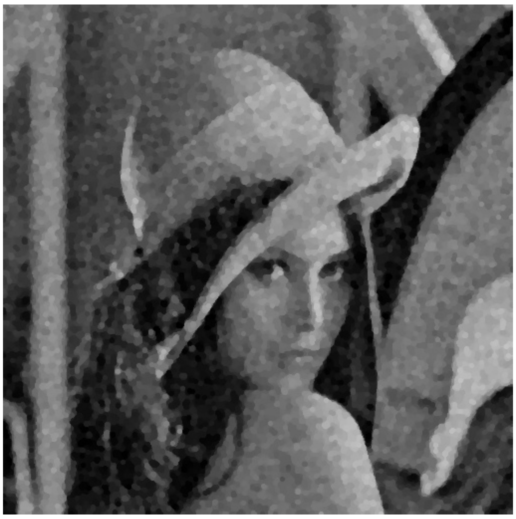
3x3 Median filter



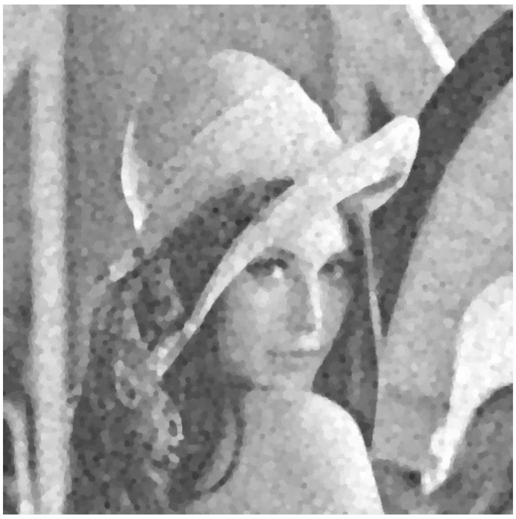
5x5 Median filter



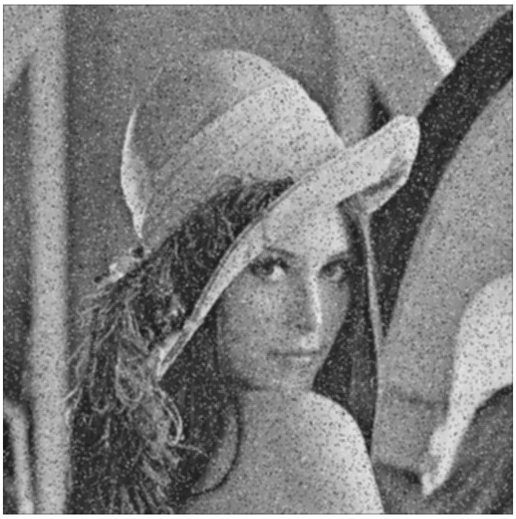
Opening then Closing

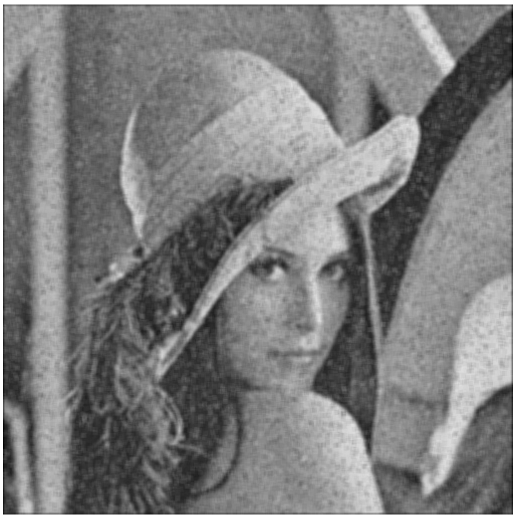


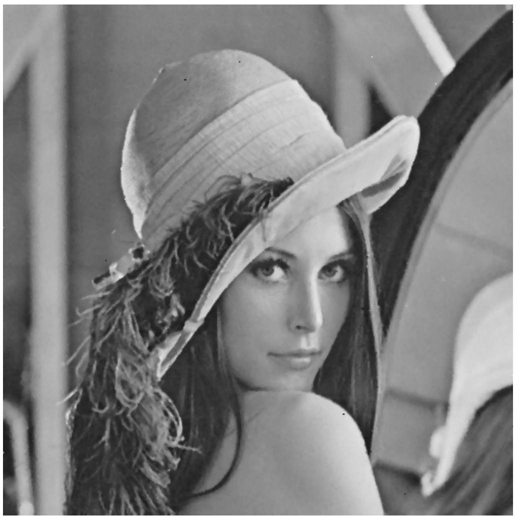
Closing then Opening

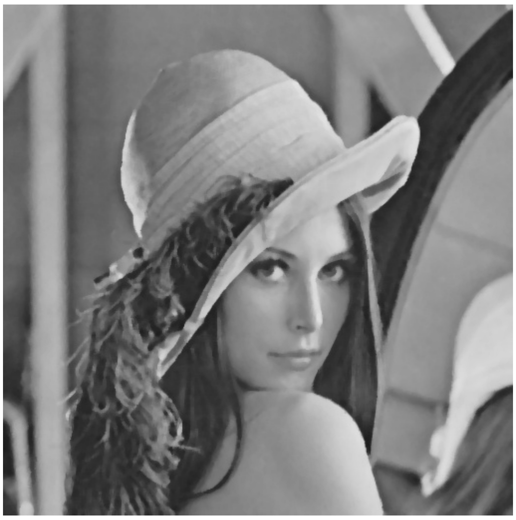


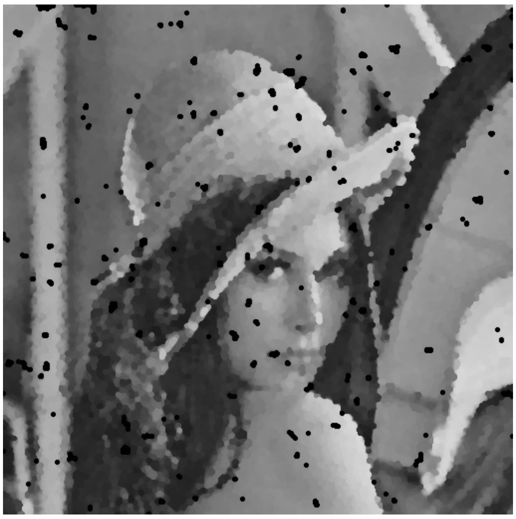
Salt-And-Pepper with threshold 0.05

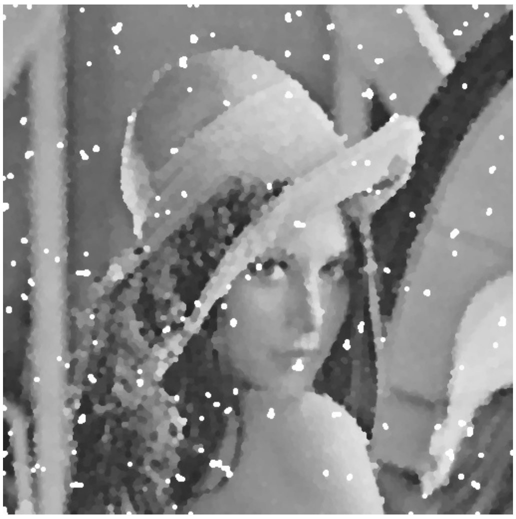

3x3 Box filter


5x5 Box filter


3x3 Median filter

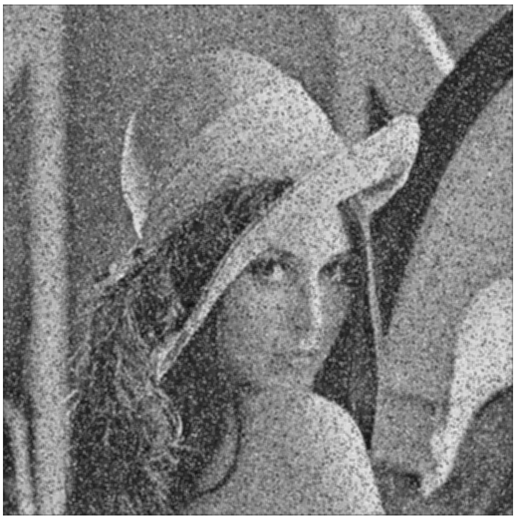

5x5 Median filter


Opening then Closing

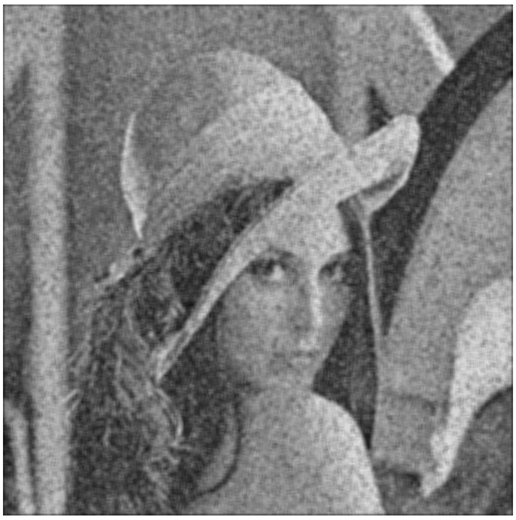

Closing then Opening

Salt-And-Pepper with threshold 0.1

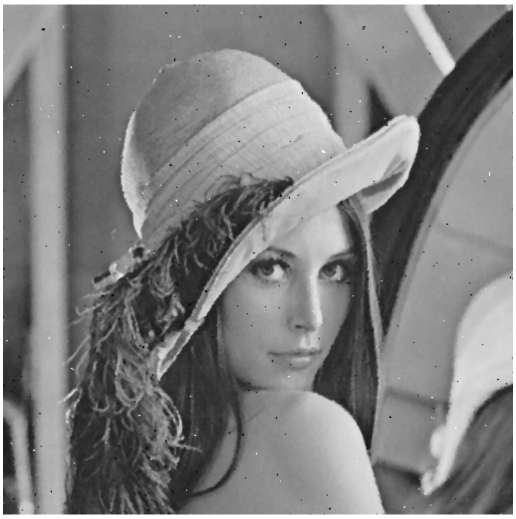
3x3 Box filter



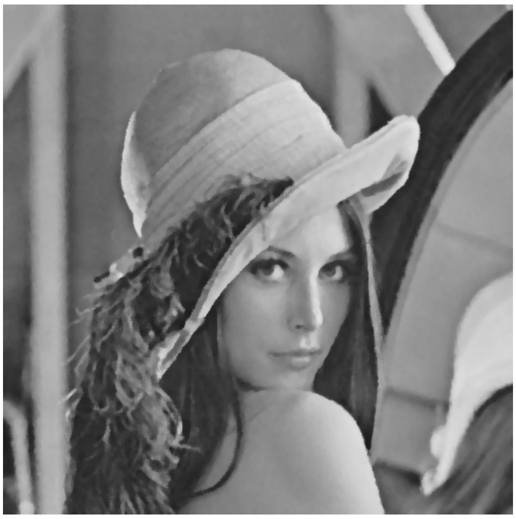
5x5 Box filter



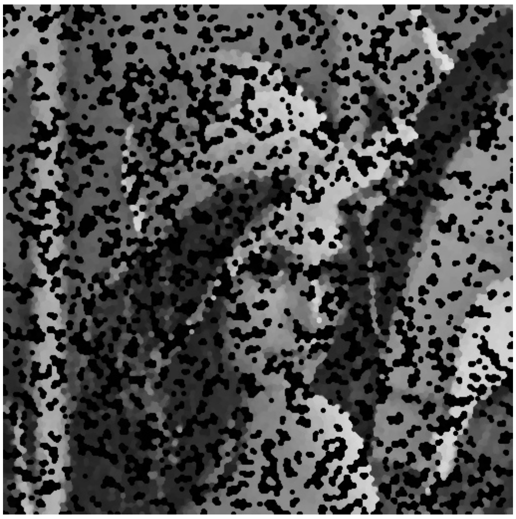
3x3 Median filter



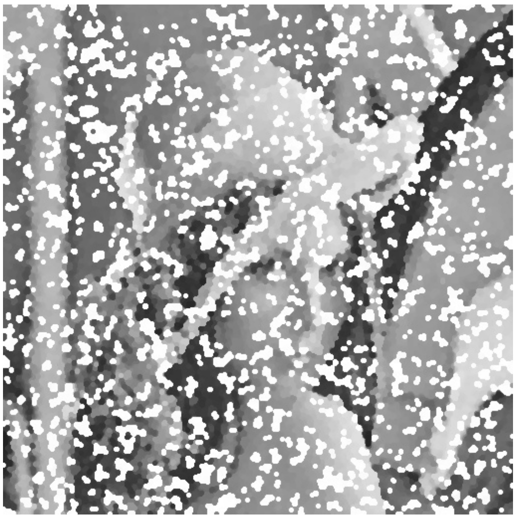
5x5 Median filter



Opening then Closing



Closing then Opening



SNR

根據定義算出SNR。

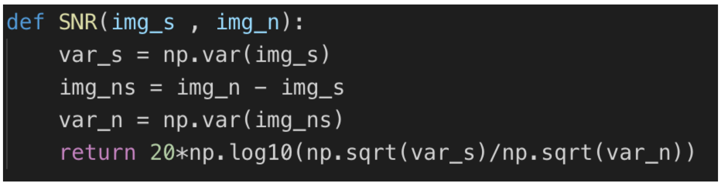
$$VS = \frac{\sum_{\forall n} (I(i,j) - \mu)^2}{\|n\|}$$

$$\mu = \frac{\sum_{\forall n} I(i,j)}{\|n\|}$$

$$VN = \frac{\sum (I_N(i,j) - I(i,j) - \mu_N)^2}{\|n\|}$$

$$\mu_N = \frac{\sum_{\forall n} (I_N(i,j) - I(i,j))}{\|n\|}$$

$$SNR = 20 \log_{10} \frac{\sqrt{VS}}{\sqrt{VN}}$$



```
def SNR(img_s , img_n):
    var_s = np.var(img_s)
    img_ns = img_n - img_s
    var_n = np.var(img_ns)
    return 20*np.log10(np.sqrt(var_s)/np.sqrt(var_n))
```

SNR	Gaussian Noise with amplitude 10	Gaussian Noise with amplitude 30	Salt-and-pepper with threshold 0.05	Salt-and-pepper with threshold 0.1
Box filter 3x3	12.15	4.37	0.99	-2.24
Box filter 5x5	10.36	3.52	0.14	-3.35
Median filter 3x3	12.83	4.40	1.11	-1.76
Median filter 5x5	11.60	3.76	0.95	-1.92
Opening then closing	20.97	15.85	2.68	4.28
Closing then opening	18.26	14.85	1.96	3.49