COMPUTER VISION 1

Homework 8

姓名　　：蘇宛琳

系所　　：電信所碩一

學號　　：R05942060

指導教授：傅楸善老師

Computer Vision Report – Homework 8

R05942060 蘇宛琳

Question :

Write the following programs

1. Generate additive white Gaussian noise

2. Generate salt-and-pepper noise

3. Run box filter *(3X3, 5X5)* on all noisy images

4. Run median filter *(3X3, 5X5)* on all noisy images

5. Run opening followed by closing or closing followed by opening

* Generate additive white Gaussian noise Concept *

White 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

Step1.

利用『X =randn(512,512)』函式產生一個高斯分布，透過亂數產生器(存為矩陣，大小為512*512對應參數為平均值為0，標準差為1)

Step2.

N=amp*X，如上面的公式，將高斯分布乘上一個amplitude(這裡指定為amp=10,30)

Step3

image1(i,j)=image1(i,j)+N(i,j)，最後將輸入的影像加上noise，並且依照每個像素點位置去慢慢拼回成原來的 lena 圖像。

Source code

```
clear;
close;
image = imread('lena.bmp');
figure;
imshow(image);
[height,width] = size(image);

amp = 30;
X = randn(512,512); %產生一個高斯分布在亂數產生器(儲存成矩陣，大小為 512*512 對應參數為平均值為 0，標準差為 1)
N = amp * X;
```

```
for i = 1 : height
    for j = 1 : width
        image(i,j) = image(i,j)+N(i,j);
    end
end
figure; imshow(image);
imwrite(image,'gau_noise2.bmp');
```

Result

* amplitude = 10

Original image

Add with Gaussian noise

Gaussian noise image

* amplitude = 30

Original image

Add with Gaussian noise

Gaussian noise image

* Generate salt-and-pepper noise Concept *

Salt-and-pepper noise

$$I(nim,i,j)=0 \ \ if\ uniform(0,1)<0.05$$
$$I(nim,i,j)=255 \ \ if\ uniform(0,1)>1-0.05$$
$$I(nim,i,j)=I(im,i,j) \ \ otherwise$$
$$uniform(0,1):random\ variable\ uniformly\ distributed\ over\ [0,1]$$
$$try\ both\ 0.05\ and\ 0.1$$

Step1.

利用『X =rand(512,512)』函式產生一個均勻分布於 0 到 1 之間的的隨機亂數，(儲存成矩陣，大小為 512*512)。

Step2.

如上面的公式，再設置一個 threshold=0.05，以及 1-threshold，當作門檻值。再將均勻分布的亂數藉由門檻值定義成三種判別式。製造出類似白點和黑點的雜訊。

Step3.

image1(i,j)=image1(i,j)+N(i,j)，最後將輸入的影像加上noise，並且依照每個像素點位置去慢慢拼回成原來的 lena 圖像。

Source code

```
image=imread('lena.bmp');
figure; imshow(image);
[height,width]=size(image);
X=rand(512,512);

p=0.05;
pp=1-p;
for i=1:height
    for j=1:width
        if  X(i,j)<p
            image(i,j)=0;
        else if X(i,j)>pp
```

```
                image(i,j)=255;
        else
        image(i,j)=image(i,j);
            end
        end
    end
end
figure; imshow(image);
imwrite(image,'sap_noise1.bmp');
```

Result

* threshold = 0.05

Add with Salt and pepper noise

Original image　　Salt and pepper noise image

* threshold = 0.1

Add with Salt and pepper noise

Original image　　Salt and pepper noise image

* Run Box filter (3X3) on all noisy images *

Box filter

$$3X3\ \text{boxfilter} : B = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Step1.

利用『filtered_img = zeros(height,width)』函式產生一個和輸入影像大小相等的為零矩陣(用途為 ： 最後將 3*3 box filter 權重分配至原圖像後最終值可以存回此矩陣=output image)。

Step2.

如上面的範例操作，利用 4 個 for 迴圈 sum=sum+image(k,l)，filtered_img(i+1,j+1)=sum/9，由於 3*3box filter 矩陣值都為1，因此直接將image累加，最後除上9，表示此矩陣的所有像素總和(做normalization)

Step3.

filtered_img(i+1,j+1) = mean2(image(i:i+2,j:j+2))，最後將輸入的影像透過 filter 去除掉 noise,並且依照每個像素點位置去慢慢拼回成原來的 lena 圖像。

Source code

```
function [filtered_img]=boxthree(image)


image=imread('sap_noise2.bmp');
figure;
imshow(image);
```

```
[height,width] = size(image);

filtered_img = zeros(height,width);

for i = 1:height-2

    for j = 1:width-2

        sum =0;

        for k = i:i+2

            for l = j:j+2

                sum=sum+image(k,l);

                filtered_img(i+1,j+1)=sum/9;

            end

         end

         filtered_img(i+1,j+1) = mean2(image(i:i+2,j:j+2));

     end

end

filtered_img=uint8(filtered_img);

figure;

imshow(filtered_img);

imwrite(filtered_img,'boxthree_sap_noise2.bmp');

end
```

* amplitude = 10

Original image

Add with Gaussian noise

Gaussian noise image

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Multiply a 3X3 box filter to remove noise

After 3*3 box filter remove noise

* amplitude = 30

Original image

Add with Gaussian noise

Gaussian noise image

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Multiply a 3X3 box filter to remove noise

After 3*3 box filter remove noise

* threshold = 0.05

Original image

Add with Salt and pepper noise

Salt and pepper noise image

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Multiply a 3X3 box filter to remove noise

After 3*3 box filter remove noise

* threshold = 0.1

Original image

Add with Salt and pepper noise

Salt and pepper noise image

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Multiply a 3X3 box filter to remove noise

After 3*3 box filter remove noise

* Run Box filter (5X5) on all noisy images *

Box filter

$$\text{5X5 boxfilter : B} = \frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

Step1.

利用『filtered_img = zeros(height,width)』函式產生一個和輸入影像大小相等的為零矩陣(用途為 ： 最後將 5*5 box filter 權重分配至原圖像後最終值可以存回此矩陣=output image)。

Step2.

如上面的範例操作，利用 4 個 for 迴圈 sum=sum+image(k,l)，filtered_img(i+1,j+1)=sum/25，由於 5*5box filter 矩陣值都為1，因此直接將image累加，最後除上25，表示此矩陣的所有像素總和(做normalization)

Step3.

filtered_img(i+1,j+1) = mean2(image(i:i+4,j:j+4))，最後將輸入的影像透過 filter 去除掉 noise,並且依照每個像素點位置去慢慢拼回成原來的 lena 圖像。

Source code

```
function [filtered_img]=boxfive(image)


image=imread('sap_noise2.bmp');
```

```
figure;
imshow(image);
[height,width] = size(image);
filtered_img = zeros(height,width);
for i = 1:height-4
    for j = 1:width-4
        sum =0;
        for k = i:i+4
            for l = j:j+4
                sum=sum+image(k,l);
                filtered_img(i+1,j+1)=sum/25;
            end
        end
        filtered_img(i+1,j+1) = mean2(image(i:i+4,j:j+4));
    end
end
filtered_img=uint8(filtered_img);
figure;
imshow(filtered_img);
imwrite(filtered_img,'boxfive_sap_noise2.bmp');
end
```

* amplitude = 10

Original image

Add with Gaussian noise

Gaussian noise image

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Multiply a 5X5 box filter to remove noise

After 5*5 box filter remove noise

* amplitude = 30

Add with Gaussian noise

Original image

Gaussian noise image

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Multiply a 5X5 box filter to remove noise

After 5*5 box filter remove noise

* threshold = 0.05

Original image

Add with Salt and pepper noise

Salt and pepper noise image

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Multiply a 5X5 box filter to remove noise

After 5*5 box filter remove noise

* threshold = 0.1

Add with Salt and pepper noise

Original image

Salt and pepper noise image

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Multiply a 5X5 box filter to remove noise

After 5*5 box filter remove noise

* Run Median filter (3X3) on all noisy images *

Median filter

The Median filter is used to remove noise from an image by replacing pixels with the middle pixel value selected from a certain window size.

Step1.

3X3 Median filter，重新設計一個全為零的矩陣，大小則為原來輸入影像的大小(M,N)的各邊長加 2→(M+2,N+2)，modifyimage=zeros(size(image)+2)。

Step2.

從原來輸入影像中的每點像素質 COPY 到新的矩陣，並且設計一個 window3*3(當作判斷鄰近像素質中位數的求取)。

Step3.

將每一個 window 對應到影像中後，那一圈的 9 個像素質大小進行排序，利用到『sort』函式進行，並且求出中間值=med，再將其儲存取代到 window 的中心 window(5)的位置。

Step4.

重複步驟 3 一直到整張影像都經過 median filter 後，將影像每個點像素儲存回去到新的影像矩陣 B，之後記得轉換回 0-255 強度像素，使用『unit8』。

Source code

```
image1=imread('sap_noise2.bmp');
figure,imshow(image1);
%PAD THE MATRIX WITH ZEROS ON ALL SIDES
modifyimage1=zeros(size(image1)+2);
B=zeros(size(image1));
%COPY THE ORIGINAL IMAGE MATRIX TO THE PADDED MATRIX
for x=1:size(image1,1)
    for y=1:size(image1,2)
        modifyimage1(x+1,y+1)=image1(x,y);
    end
end
%LET THE WINDOW BE AN ARRAY
%STORE THE 3-by-3 NEIGHBOUR VALUES IN THE ARRAY
%SORT AND FIND THE MIDDLE ELEMENT
for i= 1:size(modifyimage1,1)-2
    for j=1:size(modifyimage1,2)-2
        window=zeros(9,1);
        inc=1;
        for x=1:3
            for y=1:3
                window(inc)=modifyimage1(i+x-1,j+y-1);
                inc=inc+1;
            end
        end
        med=sort(window);
        %PLACE THE MEDIAN ELEMENT IN THE OUTPUT MATRIX
        B(i,j)=med(5);
    end
end
%CONVERT THE OUTPUT MATRIX TO 0-255 RANGE IMAGE TYPE
B=uint8(B);
figure,imshow(B);
imwrite(B,'medianthree_sap_noise2.bmp');
```

* amplitude = 10

Original image

Add with Gaussian noise

Gaussian noise image

$$\begin{bmatrix} 12 & 77 & 17 \\ 6 & 54 & 33 \\ 80 & 27 & 3 \end{bmatrix}$$

Multiply a 3X3 Median filter to remove

After 3*3 Median filter remove noise

* amplitude = 30

Original image

Add with Gaussian noise

Gaussian noise image

$$\begin{bmatrix} 12 & 77 & 17 \\ 6 & 54 & 33 \\ 80 & 27 & 3 \end{bmatrix}$$

Multiply a 3X3 Median filter to remove

After 3*3 Median filter remove noise

* threshold = 0.05

Original image

Add with

Salt and

pepper

noise

Salt and pepper noise image

$$\begin{bmatrix} 12 & 77 & 17 \\ 6 & 54 & 33 \\ 80 & 27 & 3 \end{bmatrix}$$

Multiply a 3X3

Median filter

to remove

After 3*3 Median filter remove noise

* threshold = 0.1

Original image

Add with Salt and pepper noise

Salt and pepper noise image

$$\begin{bmatrix} 12 & 77 & 17 \\ 6 & 54 & 33 \\ 80 & 27 & 3 \end{bmatrix}$$

Multiply a 3X3 Median filter to remove

After 3*3 Median filter remove noise

* Run Median filter (5X5) on all noisy images *

Median filter

The Median filter is used to remove noise from an image by replacing pixels with the middle pixel value selected from a certain window size.

Step1.

5X5 Median filter，重新設計一個全為零的矩陣，大小則為原來輸入影像的大小(M,N)的各邊長加 4→(M+4,N+4)，modifyimage=zeros(size(image)+4)。

Step2.

從原來輸入影像中的每點像素質 COPY 到新的矩陣，並且設計一個 window5*5(當作判斷鄰近像素質中位數的求取)。

Step3.

將每一個 window 對應到影像中後，那一圈的 25 個像素質大小進行排序，利用到『sort』函式進行，並且求出中間值=med，再將其儲存取代到 window 的中心 window(13)的位置。

Step4.

重複步驟 3 一直到整張影像都經過 median filter 後，將影像每個點像素儲存回去到新的影像矩陣 B，之後記得轉換回 0-255 強度像素，使用『unit8』。

Source code

```
image1=imread('sap_noise2.bmp');
figure,imshow(image1);
%PAD THE MATRIX WITH ZEROS ON ALL SIDES
modifyimage1=zeros(size(image1)+4);
B=zeros(size(image1));
%COPY THE ORIGINAL IMAGE MATRIX TO THE PADDED MATRIX
for x=1:size(image1,1)
    for y=1:size(image1,2)
        modifyimage1(x+1,y+1)=image1(x,y);
    end
end
%LET THE WINDOW BE AN ARRAY
%STORE THE 3-by-3 NEIGHBOUR VALUES IN THE ARRAY
%SORT AND FIND THE MIDDLE ELEMENT
for i= 1:size(modifyimage1,1)-4
    for j=1:size(modifyimage1,2)-4
        window=zeros(25,1);
        inc=1;
        for x=1:5
            for y=1:5 window(inc)=modifyimage1(i+x-1,j+y-1);
                inc=inc+1;
            end
        end
        med=sort(window);
        %PLACE THE MEDIAN ELEMENT IN THE OUTPUT MATRIX
        B(i,j)=med(13);
     end
end
%CONVERT THE OUTPUT MATRIX TO 0-255 RANGE IMAGE TYPE
B=uint8(B);
figure,imshow(B);
imwrite(B,'medianfive_sap_noise2.bmp');
```

* amplitude = 10

Original image

Add with Gaussian noise

Gaussian noise image

$$\begin{bmatrix} 12 & 52 & 3 & 1 & 30 \\ 1 & 13 & 27 & 59 & 30 \\ 11 & 22 & 77 & 5 & 11 \\ 32 & 98 & 72 & 6 & 14 \\ 44 & 1 & 78 & 12 & 17 \end{bmatrix}$$

Multiply a 5X5 Median filter to remove noise

After 5*5 Median filter remove noise

* amplitude = 30

Original image

Add with Gaussian noise

Gaussian noise image

$$\begin{bmatrix} 12 & 52 & 3 & 1 & 30 \\ 1 & 13 & 27 & 59 & 30 \\ 11 & 22 & 77 & 5 & 11 \\ 32 & 98 & 72 & 6 & 14 \\ 44 & 1 & 78 & 12 & 17 \end{bmatrix}$$

Multiply a 5X5 Median filter to remove noise

After 5*5 Median filter remove noise

* threshold = 0.05

Original image

Add with Salt and pepper noise

Salt and pepper noise image

$$\begin{bmatrix} 12 & 52 & 3 & 1 & 30 \\ 1 & 13 & 27 & 59 & 30 \\ 11 & 22 & 77 & 5 & 11 \\ 32 & 98 & 72 & 6 & 14 \\ 44 & 1 & 78 & 12 & 17 \end{bmatrix}$$

Multiply a 5X5 Median filter to remove noise

After 5*5 Median filter remove noise

* threshold = 0.1

Add with Salt and pepper noise

Original image

Salt and pepper noise image

$$\begin{bmatrix} 12 & 52 & 3 & 1 & 30 \\ 1 & 13 & 27 & 59 & 30 \\ 11 & 22 & 77 & 5 & 11 \\ 32 & 98 & 72 & 6 & 14 \\ 44 & 1 & 78 & 12 & 17 \end{bmatrix}$$

Multiply a 5X5 Median filter to remove noise

After 5*5 Median filter remove noise

*** Run opening followed by closing or closing followed by opening***

opening followed by closing

先進行 opening 再進行 closing

closing followed by opening

先進行 closing 再進行 opening

1. opening

$$f_{opening} = f \circ K$$

$$(f \ominus K) \oplus K$$

先求 Erosion 再求 Dilation

2. closing

$$f_{closing} = f \bullet K$$

$$(f \oplus K) \ominus K$$

先求 Dilation 再求 Erosion

Using octagonal kernel K = $\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

Source code (Main code)

```
%% programs which do gray scale morphological opening, closing,

transform on a binary image

clc;

clear all;

close all;

%input=imread('sap_noise1.bmp');

I=imread('gau_noise1.bmp');



K=[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];

Kc=[3,3];



img_open=GrayImageOpening(I,K,Kc,1);

outputOC=GrayImageClosing(img_open,K,Kc,1);

figure;

imshow(outputOC);

imwrite(outputOC,'OC_gau_noise1.bmp');



img_close=GrayImageClosing(I,K,Kc,1);

outputCO=GrayImageOpening(img_close,K,Kc,1);

figure;

imshow(outputCO);

imwrite(outputCO,'CO_gau_noise1.bmp');
```

Source code (Gray Image Dilation Function)

```
function output=GrayImageDilation(input,K,Kc,showImg)

if ~exist('showImg')
    showImg=0;
end
[im_height,im_width]=size(input);
% Find the coordinate where '1' exists
[Krow,Kcol]=find(K);
Krow=Krow-Kc(1); % Relative position which kernel center is at (0,0)
Kcol=Kcol-Kc(2);
% Minimum and maximum distance between the origin and the borders of
K
% To avoid asymmetrical kernel, we use 4 parameters.
min_h=-min(Krow);
max_h=max(Krow);
min_w=-min(Kcol);
max_w=max(Kcol);

im_tmp=zeros(im_height+min_h+max_h, im_width+min_w+max_w);
im_tmp(min_h+1:min_h+im_height, min_w+1:min_w+im_width)=input;
output=zeros(im_height, im_width);

% Process for dilation
for c=1:im_width
    for  r=1:im_height
        x=c+Krow+min_h;
        y=r+Kcol+min_w;
        abc=(x-1)*size(im_tmp,1)+y;
        output(r,c)=max(im_tmp(abc));
    end
end
```

Source code (Gray Image Erosion Function)

```
function output=GrayImageErosion(input,K,Kc,showImg)
if ~exist('showImg')
    showImg=0;
end
[im_height,im_width]=size(input);
% Find the coordinate where '1' exists
[Krow,Kcol]=find(K);
Krow=Krow-Kc(1); % Relative position which kernel center is at (0,0)
Kcol=Kcol-Kc(2);

min_h=-min(Krow);
max_h=max(Krow);
min_w=-min(Kcol);
max_w=max(Kcol);

im_tmp=255*ones(im_height+min_h+max_h, im_width+min_w+max_w);
im_tmp(min_h+1:min_h+im_height, min_w+1:min_w+im_width)=input;
output=zeros(im_height, im_width);

% Process for dilation
for c=1:im_width
    for r=1:im_height
        x=c+Krow+min_h;
        y=r+Kcol+min_w;
        abc=(x-1)*size(im_tmp,1)+y;
        output(r,c)=min(im_tmp(abc)); lean ??? ????????
    end
end
output=uint8(output);
```

Source code (Gray Image Opening Function)

```
function output=GrayImageOpening(input,K,Kc,showImg)


if ~exist('showImg')
    showImg=0;
end


% 先 Erosion 在 Dilation
img_ero=GrayImageErosion(input,K,Kc);
output=GrayImageDilation(img_ero,K,Kc);


if showImg~=0
   figure;
   imshow(output);
   title('Opening');
   imwrite(output,'Opening.bmp')
end
end
```

Source code (Gray Image Closing Function)

```
function output=GrayImageClosing(input,K,Kc,showImg)


if ~exist('showImg')
    showImg=0;
end


% ? Dilation ? Erosion
img_dil=GrayImageDilation(input,K,Kc);
output=GrayImageErosion(img_dil,K,Kc);


if showImg~=0
   figure;
   imshow(output);
   title('Closing');
   imwrite(output,'Closing.bmp')
end
end
```

* amplitude = 10

Original image

Add with

Gaussian

noise

Gaussian noise image

先 Opening

再 Closing

After opening first and then closing

* amplitude = 30

Original image

Add with

Gaussian

noise

Gaussian noise image

先 Opening

再 Closing

After opening first and then closing

* threshold = 0.05

Original image

Add with

Salt and

pepper

noise

Salt and pepper noise image

先 Opening

再 Closing

After opening first and then closing

* threshold = 0.1

Original image

Add with

Salt and

pepper

noise

Salt and pepper noise image

先 Opening

再 Closing

After opening first and then closing

* amplitude = 10

Original image

Add with

Gaussian

noise

Gaussian noise image

先 Closing

再 Opening

After closing first and then opening

* amplitude = 30

Original image

Add with

Gaussian

noise

Gaussian noise image

先 Closing

再 Opening

After closing first and then opening

* threshold = 0.05

Original image

Add with

Salt and

pepper

noise

Salt and pepper noise image

先 Closing

再 Opening

After closing first and then opening

* threshold = 0.1

Original image

Add with

Salt and

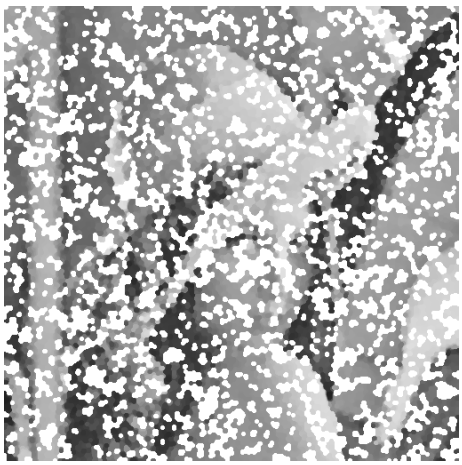
pepper

noise

Salt and pepper noise image

先 Closing

再 Opening

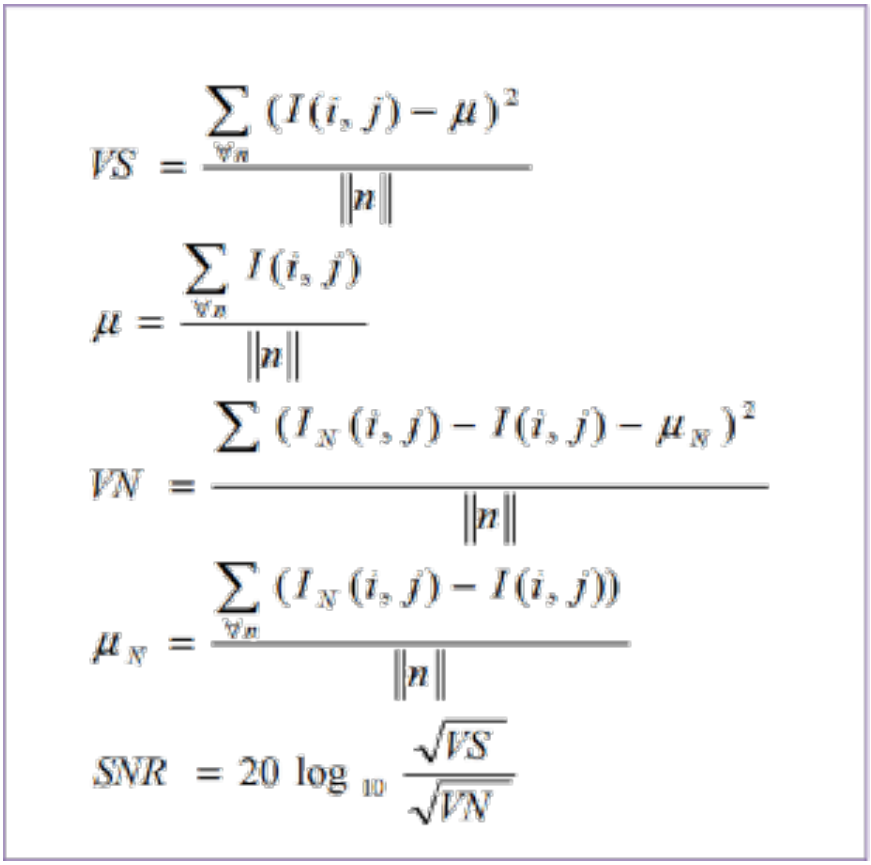


After closing first and then opening

*** calculate the signal-to-ratio (SNR) for each instance ***

signal-to-ratio (SNR)

by formula :



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

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

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

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

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

※先計算原輸入影像無雜訊Lena圖像的像素期望值和變異數，再計算含有雜訊的前面五題的輸出結果圖之像素期望值和變異數，最後再將兩者變異數帶入SNR公式取log。

Source code (signal-to-ratio (SNR) Function)

```
function outputsnr=snr(image_noise,image_ori)

image_noise=imread('OC_sap_noise1.bmp');
image_ori=imread('lena.bmp');
[height1,width1]=size(image_ori);
mean=0;
meanN=0;
vs=0;
vn=0;

for i=1:height1;
    for j=1:width1;
        mean=double(mean+image_ori(i,j));
        meanN=double(meanN+(image_noise(i,j)-image_ori(i,j)));
    end
end

mean=mean./(height1.*width1);
meanN=meanN./(height1.*width1);

for i=1:height1;
    for j=1:width1;
        tem1=double(image_ori(i,j)-mean);
        vs=vs+(tem1.*tem1);
        tem2=double(image_noise(i,j)-image_ori(i,j)-meanN);
        vn=vn+(tem2.*tem2);
    end
end
vs=vs./(height1*width1);
vn=vn./(height1*width1);

outputsnr=20.*log10(sqrt((vs./vn)));
end
```

Comparison

*signal-to-ratio (SNR)

$$SNR = 20\log_{10}\left(\frac{A_{signal}}{A_{noise}}\right)$$

1. Without filter (only noise)

2. 3x3 box filter

3. 5x5 box filter

4. 3x3 median filter

5. 5x5 median filter

6. Opening followed by closing

7. Closing followed by opening

	1	2	3	4	5	6	7
GAU 10	13.57	16.37	13.59	17.54	15.81	13.25	13.62
GAU 30	4.18	12.13	12.36	11.00	12.58	11.16	11.18
SAP 0.05	0.88	9.20	10.53	18.89	15.93	5.26	5.19
SAP 0.1	-2.08	6.26	8.22	14.41	14.15	-2.12	-2.57

*MSE

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$$

1. Without filter (only noise)

2. 3x3 box filter

3. 5x5 box filter

4. 3x3 median filter

5. 5x5 median filter

6. Opening followed by closing

7. Closing followed by opening

	1	2	3	4	5	6	7
GAU 10	100.62	52.89	100.47	40.35	60.11	251.65	227.82
GAU 30	874.35	140.04	133.22	181.91	126.56	1359.51	1341.12
SAP 0.05	1869.58	274.79	202.30	29.56	58.45	820.47	833.62
SAP 0.1	3703.90	541.31	344.82	82.92	88.21	6069.71	6905.31