Weifeng Jin

UID: 904429456
MATH 155 HW #8
Date: 03/09/2017

Problem 1

Suppose the clear true image is f, the image with noise if $g = f + noise$, and we want to output image $\hat{f}$.
(a) Suppose we use S_{xy}, a $m \times n$mask. The formula for the geometric mean filter is:

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

Notice that for each (x, y), as long as there is one $g(s, t) = 0$ in the mask, $\hat{f}(x, y)$ would be 0. Thus, when applying the geometric filter to these edge pixels, these pixels would generate 0 as their output value since the values outside the edge are 0.Thus, in the output image, the edges are eliminated and the image would be less blurred.
(b) The formula for the arithmetic mean filter is:

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

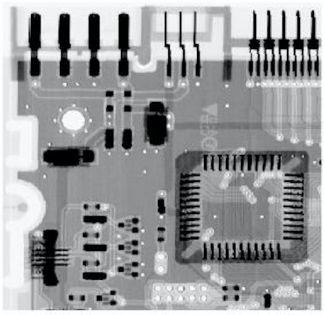
In the edge profile, these pixels would not be eliminated in the output since the summation would not be 0 as long as there is one $g(s, t) \neq 0$ in the mask. Also, when dealing with the white pixels outside the edge, adding the intensity of the edge values to the summation would make the white pixels be black and blur. Thus, the black components in the left image (arithmetic mean filter) are thicker.

Problem 2

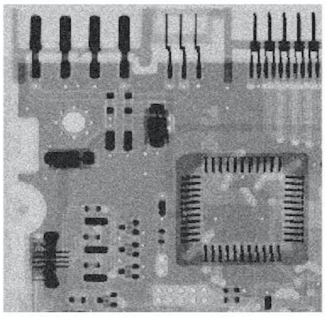
```
img = imread ('Fig5.07(b).jpg');
img = mat2gray(img,[0 255]);
[m,n] = size(img);
for i = 2:1:m-1
    for j = 2:1:n-1
        sum = 0;
        for p = i-1:1:i+1
            for q = j-1:1:j+1
```

```
                sum = sum + img(p,q);
            end
        end
        f(i,j) = sum/9;
    end
end
for i = 2:1:m-1
    for j = 2:1:n-1
        f2(i,j) = (img(i-1,j-1)*img(i-1,j)*img(i-1,j+1)*...
            img(i,j-1)*img(i,j)*img(i,j+1)*...
            img(i+1,j-1)*img(i+1,j)*img(i+1,j+1))^(1/9);
    end
end
ori = imread ('Fig5.07(a).jpg');
ori = mat2gray(ori,[0 255]);
SUM_1 = 0;
SUM_1_2 = 0;
SUM_2 = 0;
SUM_2_2 = 0;
SUM_ori = 0;
SUM_ori_2 = 0;
for i = 1:1:m
    for j = 1:1:n
        SUM_1 = SUM_1 + f(i,j)^2;
        SUM_1_2 = SUM_1_2 + (ori(i,j)-f(i,j))^2;

        SUM_ori = SUM_ori + img(i,j)^2;
        SUM_ori_2 = SUM_ori_2 + (ori(i,j)-img(i,j))^2;

        SUM_2 = SUM_2 + f2(i,j)^2;
        SUM_2_2 = SUM_2_2 + (ori(i,j)-f2(i,j))^2;
    end
end
SNR1 = 10 * log10(SUM_1/SUM_1_2)
SNR2 = 10 * log10(SUM_2/SUM_2_2)
SNR3 = 10 * log10(SUM_ori/SUM_ori_2)
subplot(2,2,1);
imshow(ori,[ ]);
xlabel('a).Original Image');
subplot(2,2,2);
imshow(img,[ ]);
xlabel({'b). Image with noise','SNR=14.1841'});
```

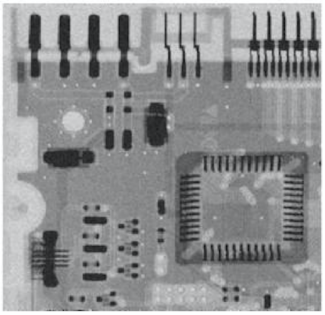
```
subplot(2,2,3);
imshow(f, [0 1]);
xlabel({'b).arithmetic mean filter','SNR=14.0472'});
subplot(2,2,4);
imshow(f, [0 1]);
xlabel({'c).geometric mean filter','SNR=14.0508'});
```



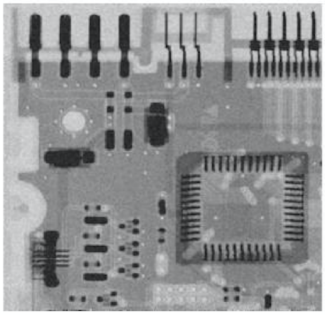
a).Original Image



b). Image with noise
SNR=14.1841



b).arithmetic mean filter
SNR=14.0472



c).geometric mean filter
SNR=14.0508

The arithmetic mean filter and geometric mean filter have relatively great denoising effect in this image visually. They have almost the same SNR. However, for some mysterious reason, the SNR for the noised image is even higher.

Problem 3

The formula of contraharmonic mean filter is:

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

(a)
When dealing with pepper noise, we want the intensity of the pixel with the pepper noise changes to a similar value of the surrounding pixels. The intensity of a pepper noise is significantly lower than the surrounding values, suppose we have a 3×3 mask as follows:

$$\begin{bmatrix} 100 & 100 & 100 \\ 100 & 0 & 100 \\ 100 & 100 & 100 \end{bmatrix}$$

The center point is the pepper noise we want to eliminate. Using $Q = 0.5$, we get $\hat{f}(x, y) = 98.78$, $Q = 1.0$, we get $\hat{f}(x, y) = 98.88$,$Q = 2$, we get $\hat{f}(x, y) = 99.99$,$Q = 5$, we get $\hat{f}(x, y) = 100$. As we can seen, when using a positive Q, $\hat{f}(x, y)$ would have a satisfactory estimation. Thus, the contraharmonic filter is effective in eliminating pepper noise when Q is positive.

(b)
The intensity of a salt noise is significantly higher than the surrounding values, suppose we have 3×3 mask as follows:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 100 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The center point is the salt noise we want to eliminate. Using $Q = -0.5$, we get $\hat{f}(x, y) = 2.22$. Using $Q = -2$, we get $\hat{f}(x, y) = 1.011$. Using $Q = -5$, we get $\hat{f}(x, y) = 1.00$. As we can seen, when using a negative Q, $\hat{f}(x, y)$ would have a satisfactory estimation. Thus, the contraharmonic filter is effective in eliminating salt noise when Q is negative.

(c)
When applying the wrong polarity, such as applying a negative Q to a pepper noise, the result would not be good. For example, if we apply $Q = -2$ to the matrix in (a), one of denominators would be 0 and we would not get any valid results. If we apply $Q = 2$ to the matrix in (b),$\hat{f}(x, y)$ would be 99.996. All in all, when applying the wrong polarity to the noise, the result output would be close to the noise's intensity rather than the surrounding areas' intensity.

(d)
When $Q = -1$, the contraharmonic mean filter would become harmonic mean filter:

$$\hat{f}(x, y) = \frac{\sum\limits_{(s,t) \in S_{xy}} g(s, t)^0}{\sum\limits_{(s,t) \in S_{xy}} g(s, t)^{-1}} = \frac{mn}{\sum\limits_{(s,t) \in S_{xy}} \frac{1}{g(s,t)}}$$

(e)

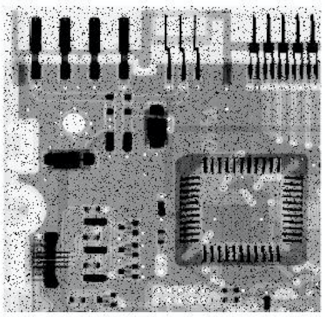
For constant gray levels, suppose all the pixels in the mask have the intensity k.$\hat{f}(x,y)$ would be k.

$$\hat{f}(x,y) = \frac{\sum\limits_{(s,t)\in S_{xy}} k^{Q+1}}{\sum\limits_{(s,t)\in S_{xy}} K^{Q}} = \frac{mnk^{Q+1}}{mnk^{Q}} = k$$

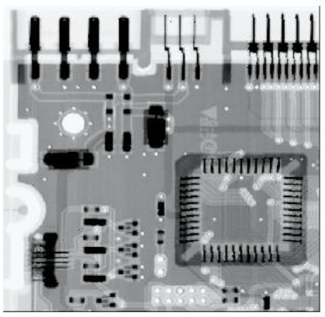
Problem 4

```
I = imread ('Fig5.08(a).jpg');
I2 = imread ('Fig5.08(b).jpg');
I = mat2gray(I,[0 255]);
I2 = mat2gray(I2, [0 255]);
I_output = contrah(I,1.5);
I2_output = contrah(I2,-1.5);
figure();
subplot(2,2,1);
imshow(I,[ ]);
xlabel('a).Original Image');
subplot(2,2,2);
imshow(I_output, [0 1]);
xlabel({'b).contraharmonic filter','Q=1.5'});
subplot(2,2,3);
imshow(I2, [ ]);
xlabel('b).Original Image');
subplot(2,2,4);
imshow(I2_output, [0 1]);
xlabel({'b).contraharmonic filter','Q=-1.5'});
function f = contrah(img,Q)
    [m,n] = size(img);
    sum_1 = 0;
    sum_1_1 = 0;
    for i = 1:1:m-1
        for j = 1:1:n-1
            for p = -1:1:1
                for q = -1:1:1
                    if (i+p>0 && i+p < m-1 && j+q>0 && j+q < n-1 && ...
                        1+p>0 && 1+p < m-1 && 1+q>0 && 1+q<n-1)
                        pix1 = (img(i+p,j+q)).^(Q+1);
                        pix2 = (img(i+p,j+q)).^Q;
                        sum_1 = sum_1 + pix1;
```

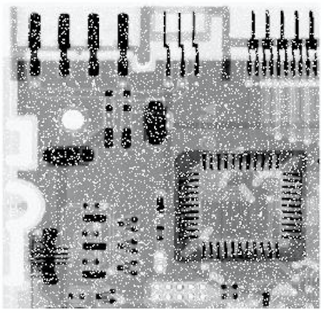
```
                    sum_1_1 = sum_1_1 + pix2;
                end
            end
        end
        f(i,j) = (sum_1/sum_1_1);
        sum_1 = 0;
        sum_1_1 = 0;
    end
end
end
```



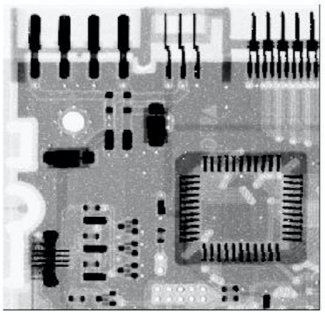
a).Original Image



b).contraharmonic filter
Q=1.5



b).Original Image



b).contraharmonic filter
Q=-1.5

Problem 5

(a)

$$f * g(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x-u, y-v)g(u,v)dudv = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x-u, y-v)f(u,v)dudv$$

(b)

From the definition of convolution and partial derivatives, we have:

$$\frac{\partial^2(f*g)}{\partial x^2} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \frac{\partial^2 g(x-u, y-v)}{\partial x^2} f dudv = \frac{\partial^2 g}{\partial x^2} * f \tag{1}$$

$$\frac{\partial^2(f*g)}{\partial y^2} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \frac{\partial^2 g(x-u, y-v)}{\partial y^2} f dudv = \frac{\partial^2 g}{\partial y^2} * f \tag{2}$$

Since we know that $\nabla^2(f*g) = \frac{\partial^2(f*g)}{\partial x^2} + \frac{\partial^2(f*g)}{\partial y^2}$, we have:

$$\begin{aligned}\nabla^2(f*g) &= \frac{\partial^2 g}{\partial x^2} * f + \frac{\partial^2 g}{\partial y^2} * f = (\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}) * f \\ &= (\nabla^2 g) * f = f * (\nabla^2 g)\end{aligned}$$