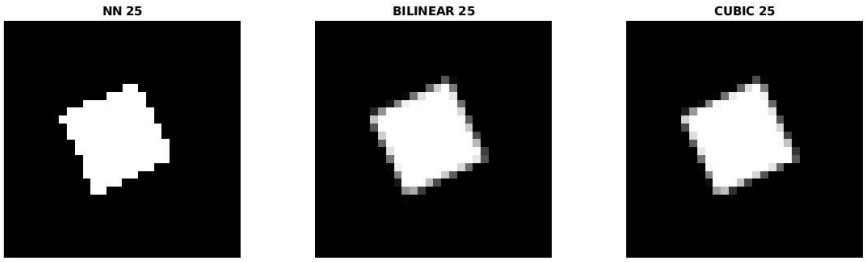
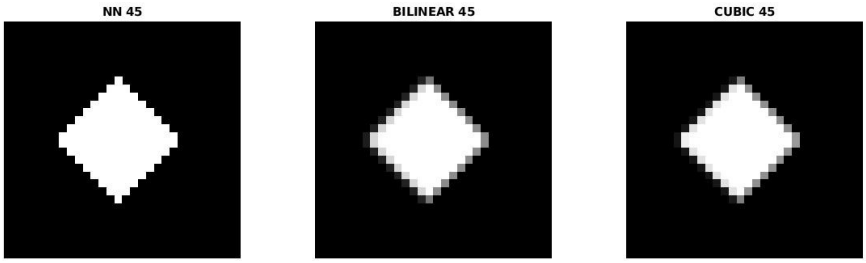
HW1: HONGRU LI 95318168

1 (a-c) The result is below:

Since nearest neighbor use only the closest pixel in rotated image for translated image, there are clear artifact(zagged line) and the image is still binary after rotation. Bilinear (bicubic) use weighted average of 2(4) points and thus there are gray points in rotated image, and the edge looks much better.



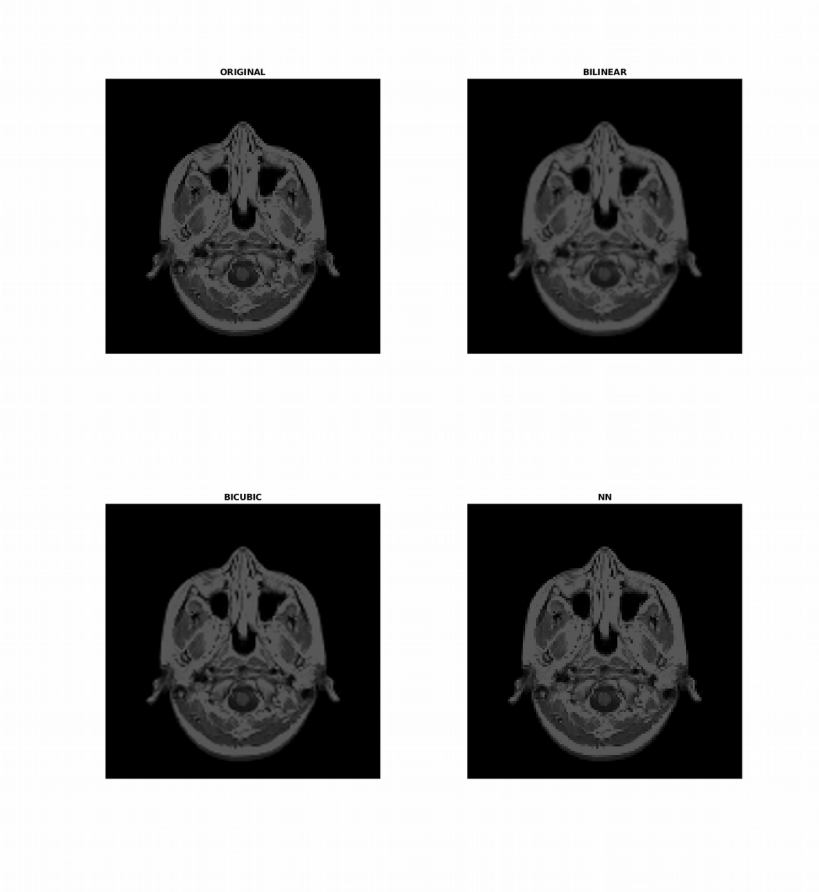


Code:

```
%==========================================
A=zeros(30,30);
imshow(A)
for i=10:20
    for j=10:20
        A(i,j)=1;
    end
end
%(b)
figure(1)
NN_25 = imrotate(A,25,'nearest','crop');
```

```
NN_45 = imrotate(A,45,'nearest','crop');
subplot(2,3,1); imshow(NN_25) ; title('NN 25') ;
subplot(2,3,4); imshow(NN_45) ; title('NN 45') ;
Bilinear_25 = imrotate(A,25,'bilinear','crop');
Bilinear_45 = imrotate(A,45,'bilinear','crop');
subplot(2,3,2); imshow(Bilinear_25) ; title('BILINEAR 25') ;
subplot(2,3,5); imshow(Bilinear_45) ; title('BILINEAR 45') ;
CUBIC_25 = imrotate(A,25,'bicubic','crop');
CUBIC_45 = imrotate(A,45,'bicubic','crop');
subplot(2,3,3); imshow(CUBIC_25) ; title('CUBIC 25') ;
subplot(2,3,6); imshow(CUBIC_45) ; title('CUBIC 45') ;
%=====================================
```

1 (d) The result is below



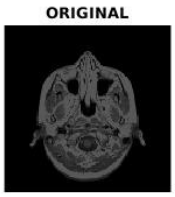
There are zagged edge if the interpolation method is nearest neighbor (And MATLAB Default method of zoom and display image is nearest neighbor). The edge made by bicubic or bilinear interpolation is smooth.

Code:

```
%=============================================
figure(2)
MRI=imread('mri.tif');
```

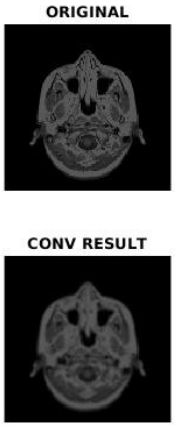
```
subplot(2,2,1);imshow(MRI);title('ORIGINAL');
MRI_BILINEAR=imresize(MRI,2,'bilinear');
subplot(2,2,2);imshow(MRI_BILINEAR);title('BILINEAR');
MRI_CUBIC=imresize(MRI,2,'bicubic');
subplot(2,2,3);imshow(MRI_CUBIC);title('BICUBIC');
MRI_NN=imresize(MRI,2,'nearest');
subplot(2,2,4);imshow(MRI_NN);title('NN');
%=======================================
```

2 (a, b) h3 is sobel operator, it is highpass



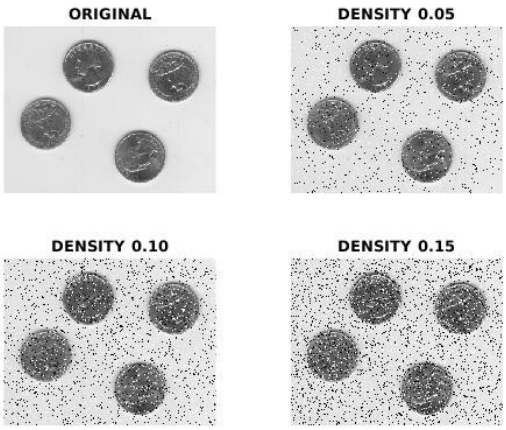

```
%=============================================
h1 = [0 0 0;1 2 1;0 0 0];
h2 = [0 1 0;0 0 0;0 -1 0];
h3 = conv2(h1, h2, 'same');
%(b) this is sobel operator, it dectect lines and thus is highpass
%(c)
MRI = imread('mri.tif');
MRI_CONV = imfilter(MRI, h3);
figure(1);
subplot(2,1,1); imshow(MRI); title('ORIGINAL');
subplot(2,1,2); imshow(MRI_CONV); title('CONV RESULT');
%=======================================
```

2(d) this is now low pass filter:

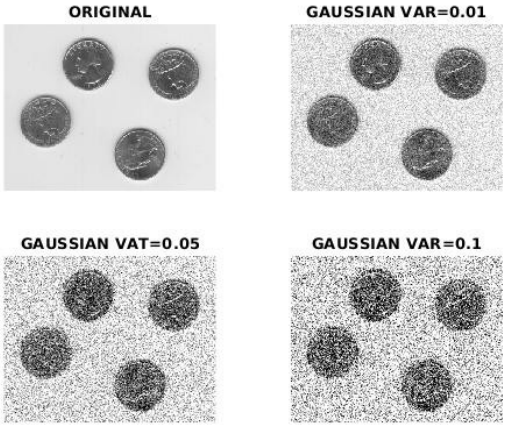



```
%=====================================
h1 = [0 0 0;1 2 1;0 0 0];
h2 = [0 1 0;0 2 0;0 1 0];
h3 = conv2(h1, h2, 'same');
h3 = h3./sum(h3(:));
MRI_CONV = imfilter(MRI, h3);
figure(2);
subplot(2,1,1); imshow(MRI); title('ORIGINAL');
subplot(2,1,2); imshow(MRI_CONV); title('CONV RESULT');
%=====================================
```

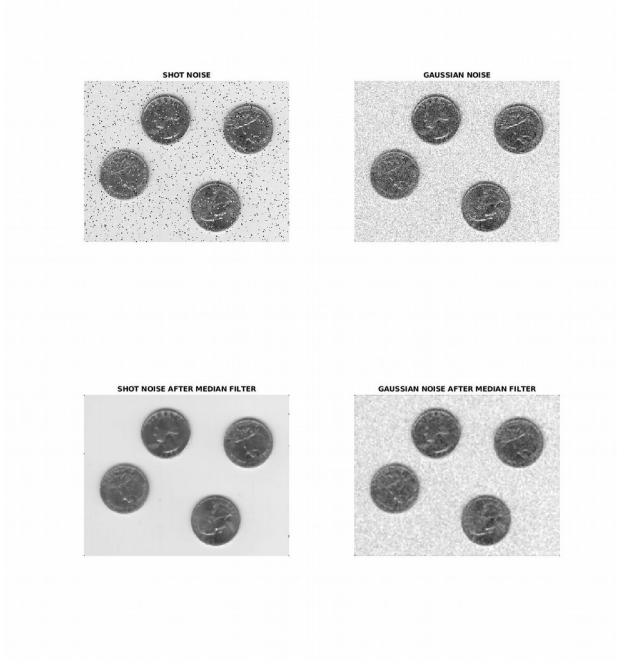
3. (a) The salt noise has no variance but density, the higher density means more points are affected



3 (b)



3 (c)

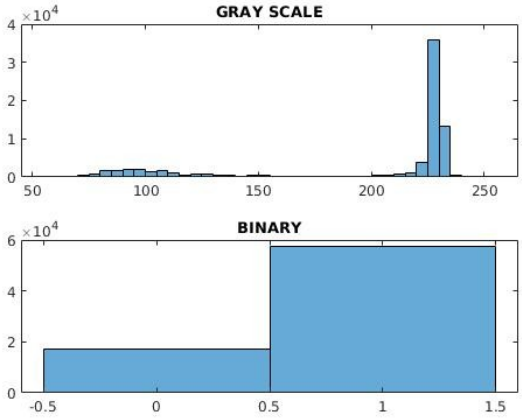


Gaussian noise makes every points noisy while salt noise make random points white or black. The median filter works well for salt noise but can not handle gaussian noise.

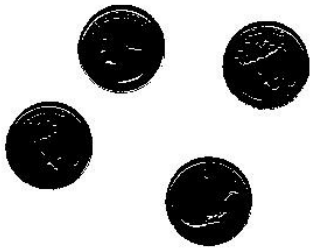
```
%=====================================
E = imread('eight.tif');
%(a)
figure(1);
subplot(2,2,1);imshow(E);title('ORIGINAL');
subplot(2,2,2);imshow(imnoise(E, 'salt & pepper',
0.05));title('DENSITY 0.05');
subplot(2,2,3);imshow(imnoise(E, 'salt & pepper',
0.10));title('DENSITY 0.10');
subplot(2,2,4);imshow(imnoise(E, 'salt & pepper',
0.15));title('DENSITY 0.15');
%(b)
```

```
figure(2);
subplot(2,2,1);imshow(E);title('ORIGINAL');
subplot(2,2,2);imshow(imnoise(E, 'gaussian', 0,
0.01));title('GAUSSIAN VAR=0.01');
subplot(2,2,3);imshow(imnoise(E, 'gaussian', 0,
0.05));title('GAUSSIAN VAT=0.05');
subplot(2,2,4);imshow(imnoise(E, 'gaussian', 0,
0.1));title('GAUSSIAN VAR=0.1');
%(c)
figure(3);
E_shot = imnoise(E, 'salt & pepper', 0.05);
E_gaussian = imnoise(E, 'gaussian', 0, 0.01);
subplot(2,2,1);imshow(E_shot);title('SHOT NOISE');
subplot(2,2,2);imshow(E_gaussian);title('GAUSSIAN NOISE');
subplot(2,2,3);imshow(medfilt2(E_shot));title('SHOT NOISE
AFTER MEDIAN FILTER');
subplot(2,2,4);imshow(medfilt2(E_gaussian));title('GAUSSIAN
NOISE AFTER MEDIAN FILTER');
%==================================
```

3 (d) The binary image has only two color while the gray scale has 256 different levels:

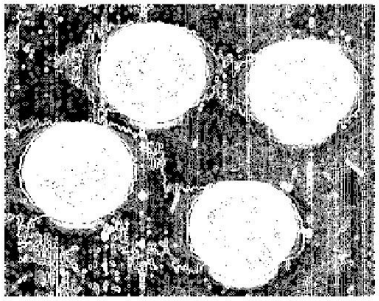


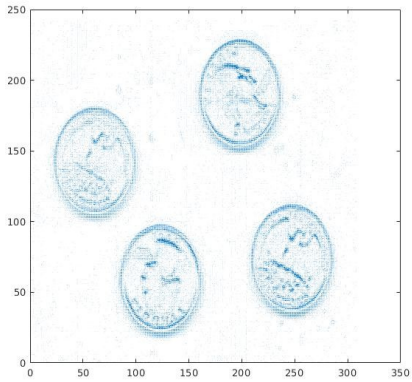
3 (e) Yes it is possible, see below:



```
%=======================================s
E = imread('eight.tif');
ref = double(imbinarize(E));
figure(1)
subplot(2,1,1);histogram(E);title('GRAY SCALE');
subplot(2,1,2);histogram(ref);title('BINARY');
figure(2)
E_binary = imhistmatch(E,ref);
imshow(E_binary);
%=======================================
```

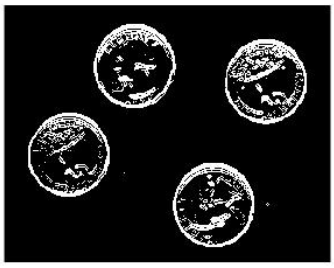
3 (i ,j) magnitude of gradient image and gradient vectors:





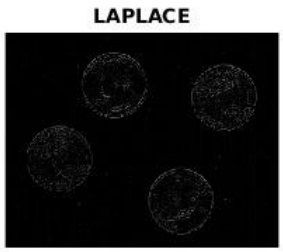

```
%=======================================
E = imread('eight.tif');
E = imgaussfilt(E);
[dy, dx] = gradient(double(E));
M = sqrt(dx.^2+dy.^2);
figure(1);
subplot(1,2,1);imshow(M);
subplot(1,2,2);quiver(dy,dx);
%=======================================
```

3 (k)



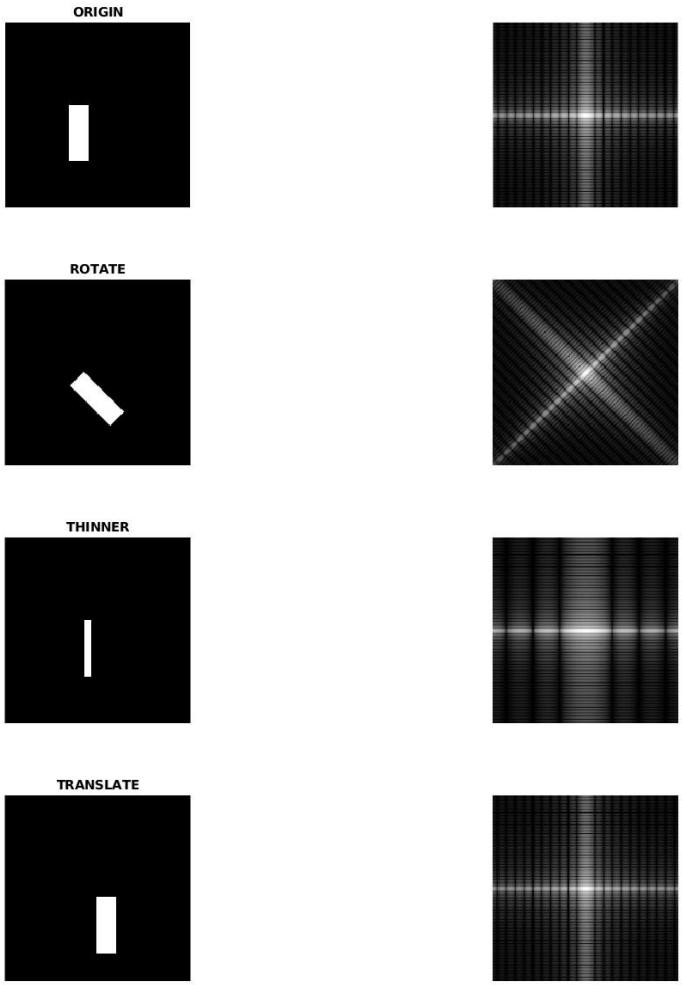
```
%=======================================
figure(2);
imshow(imbinarize(M,15));
%=======================================
```

3 (l)




```
%=======================================
figure(3);
laplace = [0 1 0; 1 -4 1; 0 1 0];
subplot(2,1,1);imshow(imfilter(E, laplace));title('LAPLACE');
gx = [-1 -2 -1; 0 0 0; 1 2 1]
gy = [-1 0 1; -2 0 2; -1 0 1]
Ex = imfilter(E, gx);
Ey = imfilter(E, gy);
subplot(2,1,2);imshow((Ex+Ey)./2);title('SOBEL');
%=======================================
```

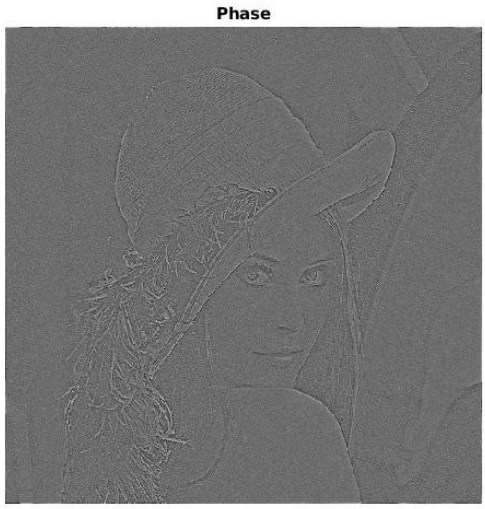
4 (a, b) When image get rotated, the fourier trasform also get rotated. When image get translated, the fourier stays the same. When image at spatial domain get thinner, fourier get “expanded”




```
%=========================================
%(a)
I = zeros(200,200);
for i=90:150
    for j=70:90
        I(i,j) = 1;
    end
end
F = fftshift(fft2(I));
F = abs(F);
```

```
F = log(1+F);
subplot(4,2,1);imshow(I);title('ORIGIN');
subplot(4,2,2);imshow(F,[]);
%rotate
I = zeros(200,200);
for i=90:150
    for j=70:90
        I(i,j) = 1;
    end
end
I = imrotate(I, 45, 'bilinear', 'crop');
F = fftshift(fft2(I));
F = abs(F);
F = log(1+F);
subplot(4,2,3);imshow(I);title('ROTATE');
subplot(4,2,4);imshow(F,[]);
%thinner
I = zeros(200,200);
for i=90:150
    for j=87:93
        I(i,j) = 1;
    end
end
F = fftshift(fft2(I));
F = abs(F);
F = log(1+F);
subplot(4,2,5);imshow(I);title('THINNER');
subplot(4,2,6);imshow(F,[]);
%translated
I = zeros(200,200);
for i=110:170
    for j=100:120
        I(i,j) = 1;
    end
end
F = fftshift(fft2(I));
F = abs(F);
F = log(1+F);
subplot(4,2,7);imshow(I);title('TRANSLATE');
subplot(4,2,8);imshow(F,[]);
%=====================================
```

4 (c, d)




```
%================================================
A = imread('lena512.bmp');
F = abs(fft2(A));
C = ifft2(F);
subplot(2,1,1);imshow(C,[]);title('Magnitude');
F = fft2(A);
F = F./abs(F);
C = ifft2(F);
subplot(2,1,2);imshow(C,[]);title("Phase");
```

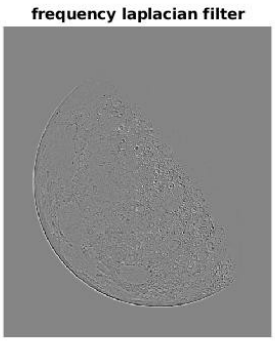
5 I calculate the energy percentage of all possible radius from 0 to width/2 and generate a look up table(An array called radius_power in my code). I check the radius for the given percentage in the array.
The radius is: remove 0.5% r=63;remov 5.4%, r=7;remove 8%, r=5


```
%==================a5.m================
A = imread('lena512.bmp');
F = fftshift(fft2(A));
```

```
F_abs = abs(F);
figure(1);
subplot(4,2,1);imshow(log(1+F_abs), []);title('FFT');
subplot(4,2,2);imshow(ifft2(ifftshift(F)), []);
radius_power = get_radius_power(F_abs);
[M,N]=size(F);
;%0.5%
r1 = get_radius(radius_power, 0.005);
for i=1:M
    for j=1:N
        if r1^2 < ((i-M/2)^2 + (j-N/2)^2)
            F(i, j)=0;
        end
    end
end
subplot(4,2,3);imshow(log(1+abs(F)), []);title('REMOVE 0.5%');
subplot(4,2,4);imshow(ifft2(ifftshift(F)), []);
%5.4%
r2 = get_radius(radius_power, 0.054);
for i=1:M
    for j=1:N
        if r2^2 < ((i-M/2)^2 + (j-N/2)^2)
            F(i, j)=0;
        end
    end
end
subplot(4,2,5);imshow(log(1+abs(F)), []);title('REMOVE 5.4%');
subplot(4,2,6);imshow(ifft2(ifftshift(F)), []);
%8%
r3 = get_radius(radius_power, 0.08);
for i=1:M
    for j=1:N
        if r3^2 < ((i-M/2)^2 + (j-N/2)^2)
            F(i, j)=0;
        end
    end
end
subplot(4,2,7);imshow(log(1+abs(F)), []);title('REMOVE 8%');
subplot(4,2,8);imshow(ifft2(ifftshift(F)), []);
%===============get_radius.m================
function radius = get_radius(radius_power, ratio)
    [M, N] = size(radius_power);
    for i = 1:N
        if radius_power(1, i)>1-ratio
```

```
            radius = i;
            break;
        end
    end
end
%=============get_radius_power.m==============
function radius_power = get_radius_power(FFTArray)
[M, N] = size(FFTArray);
radius_power = zeros(1, N/2);
total_power = 0;
for i = 1:M
    for j = 1:N
        total_power = total_power + FFTArray(i,j)^2;
    end
end
for r=1:N/2
    temp_power=0;
    for i=1:M
        for j = 1:N
            if r^2 > ((i-M/2)^2 + (j-N/2)^2)
                temp_power = temp_power + FFTArray(i,j)^2;
            end
        end
    end
    radius_power(1, r) = temp_power/total_power;
end
end
%=======================================
```

6 both filter can extract edges. But I think since frequency domain filter change the value at F(0,0) to 0 so the reconstruct image in spatial domain looks different. The frequency domain laplacian means the frequency is amplified by its radius. Thus low frequency is erased and high frequency get amplified.




```
%=======================================
A = imread('blurry_moon.tif');
subplot(3,1,1);imshow(A);
F = fftshift(fft2(A));
[M, N] = size(F);
Laplace = zeros(M, N);
for i=1:M
    for j=1:N
        Laplace(i,j) = -4*pi^2*((i-M/2)^2 + (j-N/2)^2);
```

```
    end
end
F = F.*Laplace;
A_rec = ifft2(ifftshift(F));
subplot(3,1,2);imshow(A_rec,[]);title('frequency laplacian filter')
kernel = [0 1 0; 1 -4 1; 0 1 0];
subplot(3,1,3);imshow(imfilter(A, kernel),[]);title('spatial filter')
%=====================================
```