# Assignment2 Report

Zepeng Chen

July 4, 2021

## Contents



# 1 Part I:code for manually add noise

```
%-------------------------------------------------
%Part I
%Manually add noise and do fft and ifft accordingly
%-------------------------------------------------

F=imread('moonlanding.png'); % read the gs image
im_size=size(F);% Obtain the size of the image
P=2*im_size(1);
Q=2*im_size(2); % Obtaining padding parameters as 2*image size
FS=fft2(double(F),P,Q);
%caculate the max value of specturm
                  %--------------
                  %QUESTION1
                  %--------------
FSmax=max(FS(:));
fprintf("FSmax is:%d\n", FSmax);
Fim=fftshift(FS);%move spectrum to center of image
Fim_log=log(1+abs(Fim));%compute the magnitude, log to compress the data range
%so that can be shown properly.
%FT range around from 0 to 16, so normalize it to[0,255] for 'imshow' and
%'imwrite'
Fim_log = uint8(mat2gray(Fim_log) * 255);
                  %--------------
                  %QUESTION2
                  %--------------
figure(1)
%show the spectrum of original
imshow(Fim_log,[])
imwrite(Fim_log,'ori_spec.png')
title('original spectrum')

%caculate the coordinates of noise points
%center point coordinate
x_o=P/2+1; y_o=Q/2+1;
%east noise point
%distance from noise point to center
d=100; offset=100/sqrt(2);
%east point
x_e=x_o; y_e=y_o+d;
%north east point
x_ne=round(x_o-offset); y_ne=round(y_o+offset);
%north point
x_n=x_o-d; y_n=y_o;
%north west point
x_nw=round(x_o-offset); y_nw=round(y_o-offset);
%west point
x_w=x_o; y_w=y_o-d;
```

```
%south west point
x_sw=round(x_o+offset); y_sw=round(y_o-offset);
%south point
x_s=x_o+d; y_s=y_o;
%south east point
x_se=round(x_o+offset); y_se=round(y_o+offset);
%south west point
%set neighbourhood of abouve points specified value
Fim_copy=Fim;
[Fim_copy(x_e-1,y_e+1),Fim_copy(x_e,y_e+1),Fim_copy(x_e+1,y_e+1),...
 Fim_copy(x_e-1,y_e),Fim_copy(x_e,y_e),Fim_copy(x_e+1,y_e),...
 Fim_copy(x_e-1,y_e-1),Fim_copy(x_e,y_e-1),Fim_copy(x_e+1,y_e-1)]=deal(FSmax/10);

[Fim_copy(x_se-1,y_se+1),Fim_copy(x_se,y_se+1),Fim_copy(x_se+1,y_se+1),...
 Fim_copy(x_se-1,y_se),Fim_copy(x_se,y_se),Fim_copy(x_se+1,y_se),...
 Fim_copy(x_se-1,y_se-1),Fim_copy(x_se,y_se-1),Fim_copy(x_se+1,y_se-1)]=deal(FSmax/10);

[Fim_copy(x_s-1,y_s+1),Fim_copy(x_s,y_s+1),Fim_copy(x_s+1,y_s+1),...
 Fim_copy(x_s-1,y_s),Fim_copy(x_s,y_s),Fim_copy(x_s+1,y_s),...
 Fim_copy(x_s-1,y_s-1),Fim_copy(x_s,y_s-1),Fim_copy(x_s+1,y_s-1)]=deal(FSmax/10);

[Fim_copy(x_sw-1,y_sw+1),Fim_copy(x_sw,y_sw+1),Fim_copy(x_sw+1,y_sw+1),...
 Fim_copy(x_sw-1,y_sw),Fim_copy(x_sw,y_sw),Fim_copy(x_sw+1,y_sw),...
 Fim_copy(x_sw-1,y_sw-1),Fim_copy(x_sw,y_sw-1),Fim_copy(x_sw+1,y_sw-1)]=deal(FSmax/10);

[Fim_copy(x_w-1,y_w+1),Fim_copy(x_w,y_w+1),Fim_copy(x_w+1,y_w+1),...
 Fim_copy(x_w-1,y_w),Fim_copy(x_w,y_w),Fim_copy(x_w+1,y_w),...
 Fim_copy(x_w-1,y_w-1),Fim_copy(x_w,y_w-1),Fim_copy(x_w+1,y_w-1)]=deal(FSmax/10);

[Fim_copy(x_nw-1,y_nw+1),Fim_copy(x_nw,y_nw+1),Fim_copy(x_nw+1,y_nw+1),...
 Fim_copy(x_nw-1,y_nw),Fim_copy(x_nw,y_nw),Fim_copy(x_nw+1,y_nw),...
 Fim_copy(x_nw-1,y_nw-1),Fim_copy(x_nw,y_nw-1),Fim_copy(x_nw+1,y_nw-1)]=deal(FSmax/10);

[Fim_copy(x_n-1,y_n+1),Fim_copy(x_n,y_n+1),Fim_copy(x_n+1,y_n+1),...
 Fim_copy(x_n-1,y_n),Fim_copy(x_n,y_n),Fim_copy(x_n+1,y_n),...
 Fim_copy(x_n-1,y_n-1),Fim_copy(x_n,y_n-1),Fim_copy(x_n+1,y_n-1)]=deal(FSmax/10);

[Fim_copy(x_ne-1,y_ne+1),Fim_copy(x_ne,y_ne+1),Fim_copy(x_ne+1,y_ne+1),...
 Fim_copy(x_ne-1,y_ne),Fim_copy(x_ne,y_ne),Fim_copy(x_ne+1,y_ne),...
 Fim_copy(x_ne-1,y_ne-1),Fim_copy(x_ne,y_ne-1),Fim_copy(x_ne+1,y_ne-1)]=deal(FSmax/10);
                %--------------
                %QUESTION3
                %--------------
figure(2)
Fim_copy_log=log(1+abs(Fim_copy));
Fim_copy_log=uint8(mat2gray(Fim_copy_log)*255);
imshow(Fim_copy_log,[])
imwrite(Fim_copy_log,'noise_added_spec.png');
title('noise added spectrum')
                %--------------
                %QUESTION4
                %--------------
%{
iFim=ifft2(Fim);
resize_iFim=iFim(1:im_size(1),1:im_size(2));
figure(3)
imshow(abs(resize_iFim),[]);
%}
iFim_copy=ifft2(Fim_copy);
iFim_copy=iFim_copy(1:im_size(1),1:im_size(2));
iFim_copy=uint8(abs(iFim_copy));
%iFim_copy_log=log(1+abs(iFim_copy));
%iFim_copy_log=uint8(mat2gray(iFim_copy_log)*255);
figure(3)
imshow(iFim_copy,[])
imwrite(iFim_copy,'noise_added_img.png');
title('noise added image')
                %--------------
                %QUESTION5
                %--------------
noise_Fim=fft2(iFim_copy);
```

```
noise_Fim_log=log(1+abs(noise_Fim));%use log to compress data range[0,16] around
noise_Fim_center=fftshift(noise_Fim_log);
noise_Fim_norm=uint8(mat2gray(noise_Fim_center)*255);%normalize date to [0,255]
figure(4)
imshow(noise_Fim_norm,[]);
imwrite(noise_Fim_norm,'noise_fft_spec.png')
title("noise added fft spectrum")
```

# 2 Part II:code for filters

```
%-------------------------------------------------
%Part II
%Construct three types of band-reject filter
%Ideal, Butterworth, Gaussian
%Cutoff frequency D0=100, width W=8
%For Butterworth order n=4
%-------------------------------------------------
F=imread('noise_added_img.png');
im_size=size(F); % Obtain the size of the image
P=2*im_size(1);Q=2*im_size(2); % Optaining padding parameters
FTIm=fft2(double(F),P,Q); % FT with padded size
D0 = 100; W=8;
Filter_ideal = band_reject_filters('ideal', P, Q, D0,W);
Filter_btw = band_reject_filters('btw', P, Q, D0,W);
Filter_gaussian = band_reject_filters('gaussian', P, Q, D0,W);
%center the filter
Ff_ideal=fftshift(Filter_ideal);
figure(1)
imshow(Ff_ideal,[]);
title('ideal')
imwrite(Ff_ideal,'ideal_.png');

Ff_btw=fftshift(Filter_btw);
figure(2)
imshow(Ff_btw,[]);
title('btw')
imwrite(Ff_btw,'btw_.png');

Ff_gaussian=fftshift(Filter_gaussian);
figure(3)
imshow(Ff_gaussian,[]);
title('gaussian')
imwrite(Ff_gaussian,'gaussian_.png');
                  %--------------
                  %QUESTION6
                  %--------------
%create the filters are of same size as image
%when apply the filter to the noise spectrum, will resize the filters as
%same as the spectrum size such that can do element-wise multiplication
%{
function H_out = band_reject_filters(filter_type, P, Q, D0,W)
m=P/2;n=Q/2;
D=zeros(P,Q);
for i=1:P
    for j=1:Q
        D(i,j)=sqrt((i-m)^2+(j-n)^2);
    end
end
switch filter_type
    case 'ideal'
        H2=double(D<=D0+W/2);
        H1=double(D<=D0-W/2);
        H_out=1-(H2-H1);
    case 'btw'
        H_out = 1./(1 + (D.*W./(D.^2-D0^2)).^(2*4));
    case 'gaussian'
        H_out = 1-exp(-((D.^2-D0^2)./(D.*W)).^2);
end
```

```
end
%}
                    %--------------
                    %QUESTION7
                    %--------------
Fim=fftshift(FTIm); % move the origin of the FT to the center
FTI=log(1+abs(Fim));
FTI_norm=double(mat2gray(FTI)*255);
figure(4)
imshow(FTI_norm,[]);
imwrite(uint8(FTI_norm),'noise_spec.png');
title('original spec');

ideal_filtered_spec=uint8(Ff_ideal.*FTI_norm);
figure(5)
imshow(ideal_filtered_spec,[]);
imwrite(ideal_filtered_spec,'ideal_filtered_spec.png');
title('ideal filtered spec');

btw_filtered_spec=uint8(Ff_btw.*FTI_norm);
figure(6)
imshow(btw_filtered_spec,[]);
imwrite(btw_filtered_spec,'btw_filtered_spec.png');
title('btw filtered spec');

gaussian_filtered_spec=uint8(Ff_gaussian.*FTI_norm);
figure(7)
imshow(gaussian_filtered_spec,[]);
imwrite(gaussian_filtered_spec,'gaussian_filtered_spec.png');
title('gaussian filtered spec');
                    %--------------
                    %QUESTION8
                    %--------------
ideal_image=uint8(abs(ifft2(Ff_ideal.*Fim))); % multiply the FT of
ideal_image=ideal_image(1:im_size(1), 1:im_size(2));
figure(8)
imshow(ideal_image,[]);
title('ideal image');
imwrite(ideal_image,'ideal_image.png');

btw_image=uint8(abs(ifft2(Ff_btw.*Fim))); % multiply the FT of
btw_image=btw_image(1:im_size(1), 1:im_size(2));
figure(9)
imshow(btw_image,[]);
title('btw image');
imwrite(btw_image,'btw_image.png');

gaussian_image=uint8(abs(ifft2(Ff_gaussian.*Fim))); % multiply the FT of
gaussian_image=gaussian_image(1:im_size(1), 1:im_size(2));
figure(10)
imshow(gaussian_image,[]);
title('gaussian image');
imwrite(gaussian_image,'gaussian_image.png');

function H_out = band_reject_filters(filter_type, P, Q, D0,W)
%  Developing frequency domain coordinates
u = 0:(P-1);
v = 0:(Q-1);

idx = find(u > P/2);
u(idx) = u(idx) - P;
idy = find(v > Q/2);
v(idy) = v(idy) - Q;
% Compute the meshgrid coordinates
[V, U] = meshgrid(v, u);
% Compute the istance matrix
D = sqrt(U.^2 + V.^2);

% Begin fiter computations.
switch filter_type
    case 'ideal'
```

```
        H2=double(D<=D0+W/2);
        H1=double(D<=D0-W/2);
        H_out=1-(H2-H1);
    case 'btw'
        H_out = 1./(1 + (D.*W./(D.^2-D0^2)).^(2*4));
    case 'gaussian'
        H_out = 1-exp(-((D.^2-D0^2)./(D.*W)).^2);
    otherwise
       error('Unknown filter type.')
end
end
```

# 3 Part III: questions answer

1. FSmax is:23229905
2. 

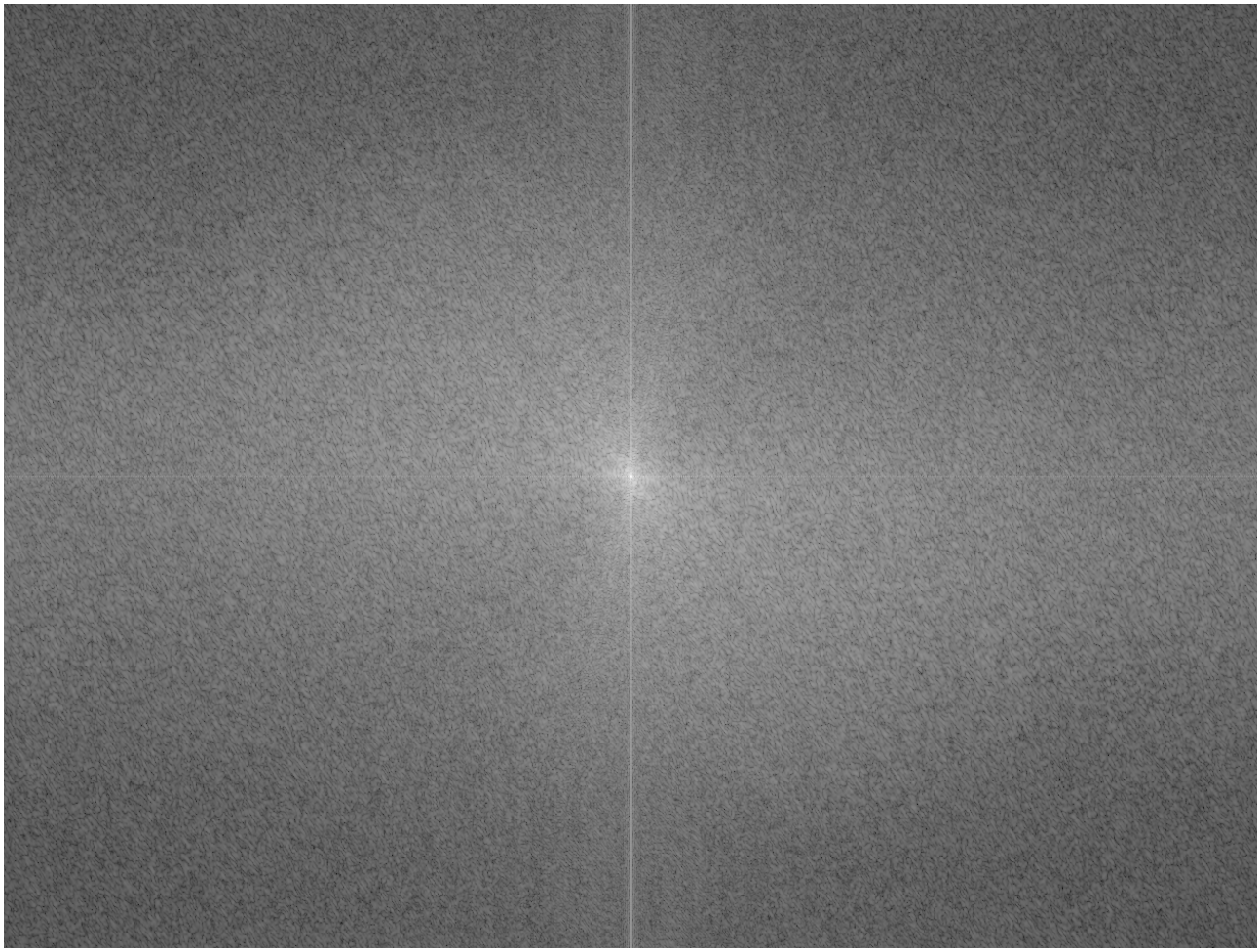


Figure 1: Original Spectrum.

3.

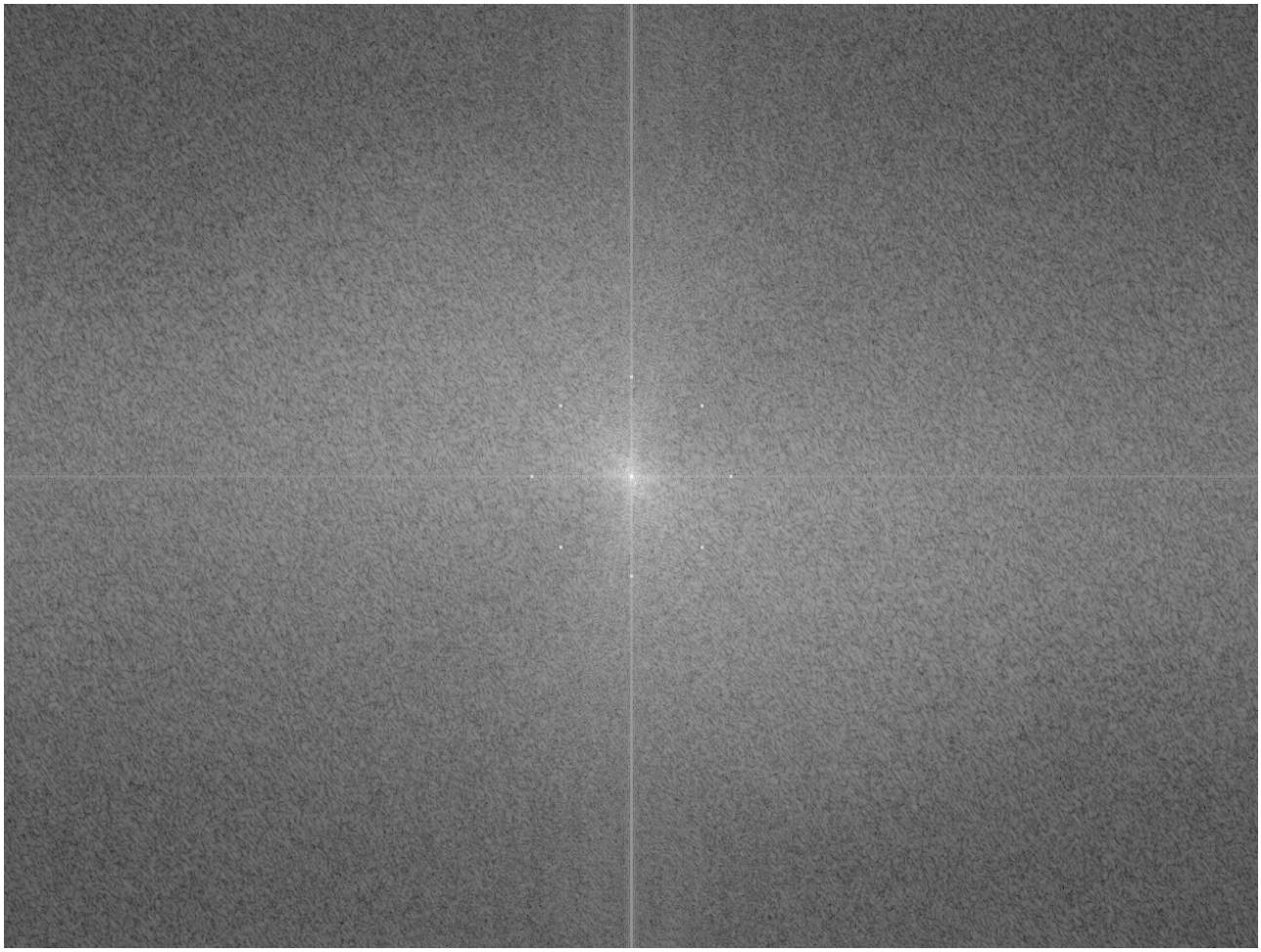
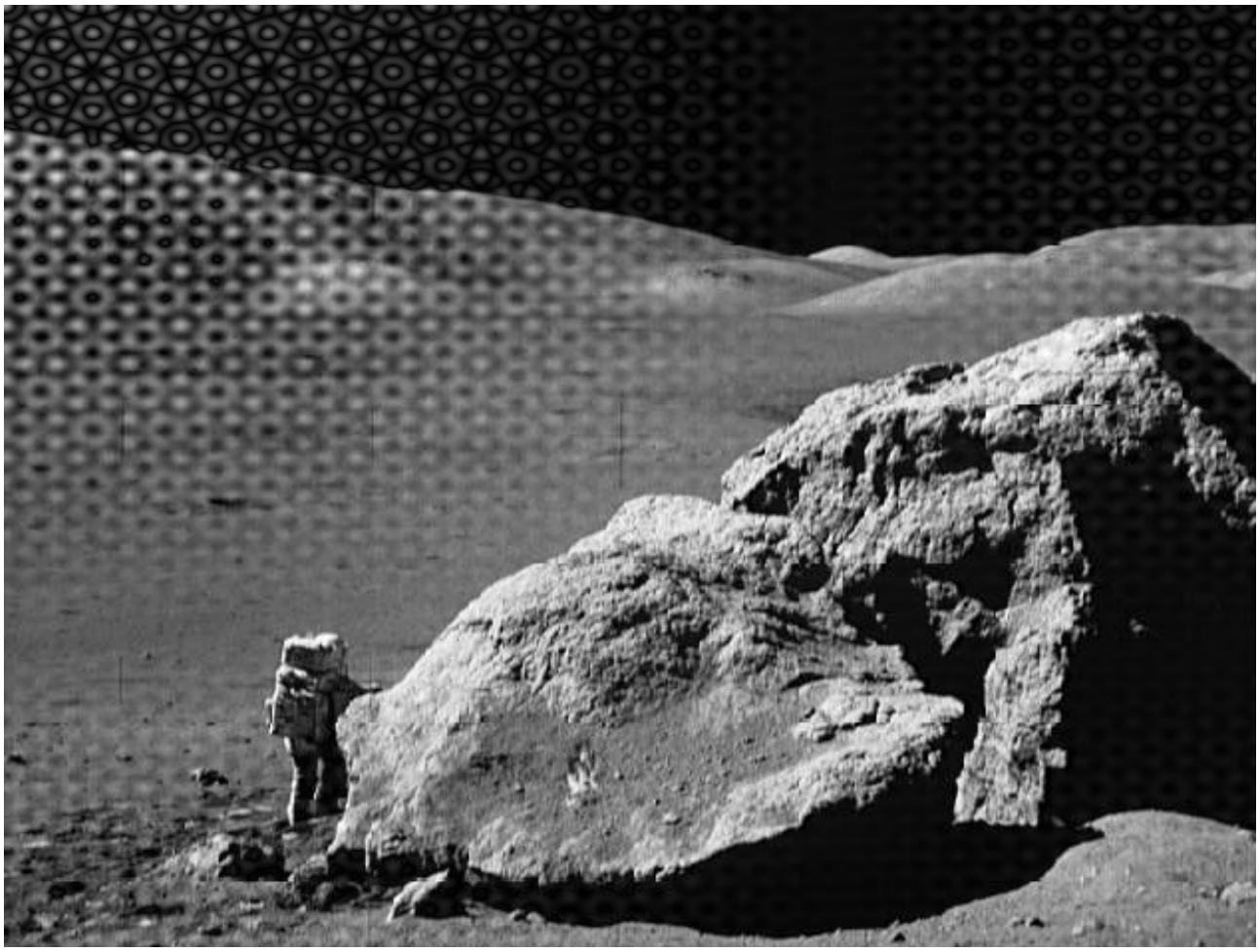


Figure 2: manual noise spectrum

4. We can found there are circles periodically repeating with dark and bright strip. Because we manually



add eight 3x3 square in spectrum. The square function's inverse fourier transformation is sinc function. So after we ifft the noise added spectrum we get that pattern noise.

5. From the image, we can see the spectrum transformed from the noise image does not look like what I created manually.

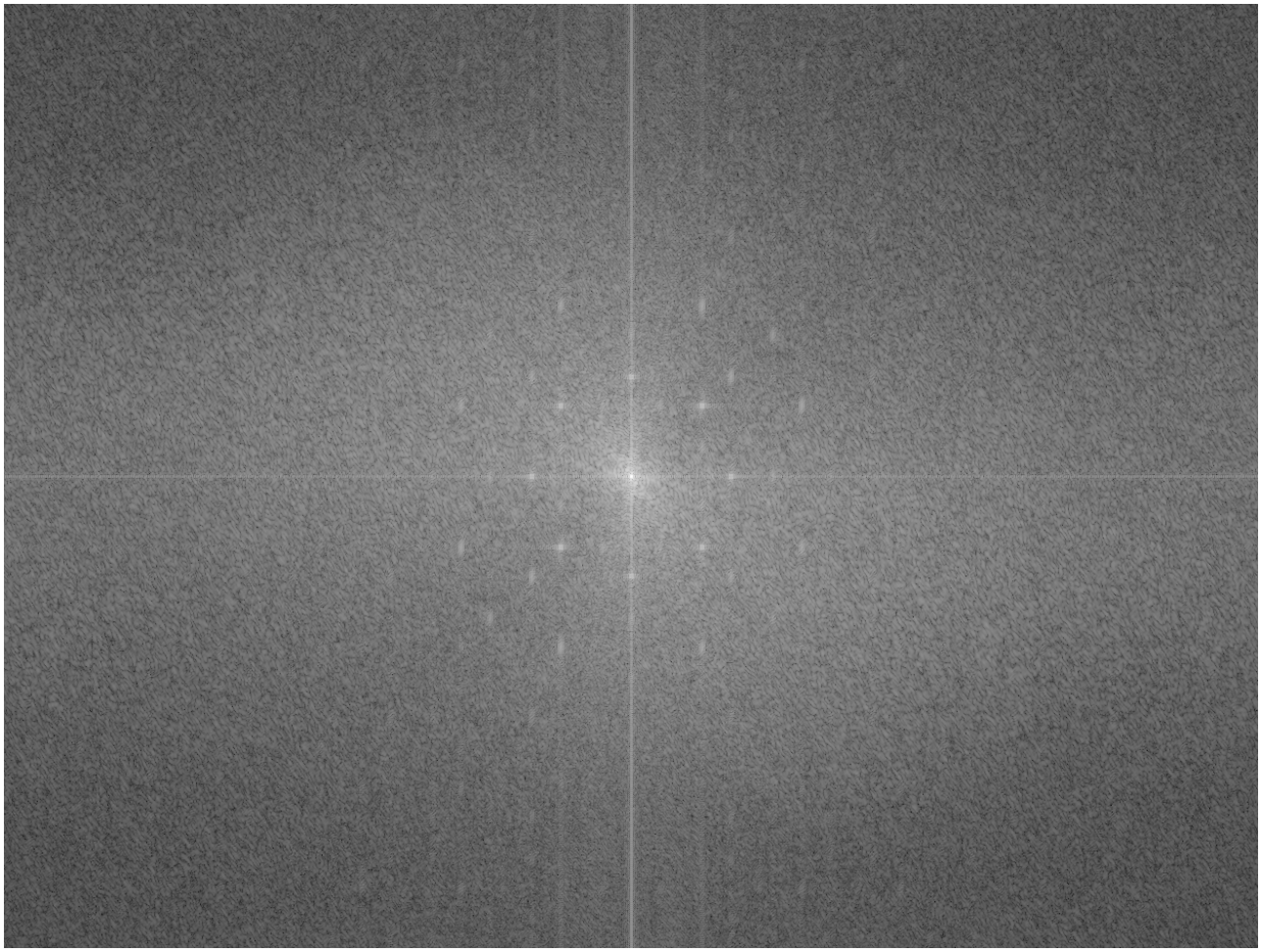


Figure 3: noise spectrum

6.

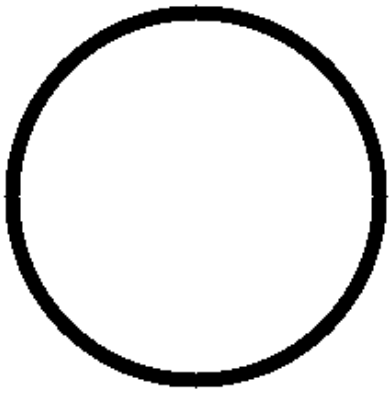


Figure 4: ideal filter.

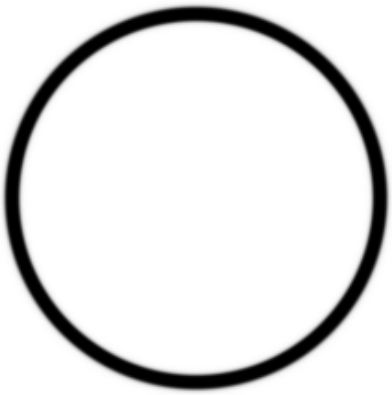


Figure 5: butterworth filter.

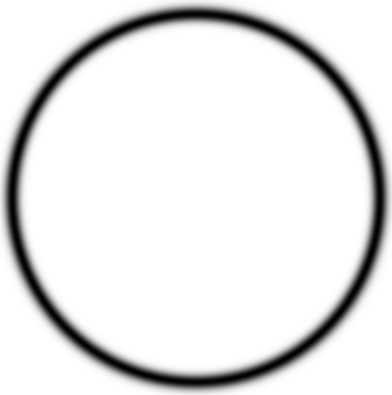
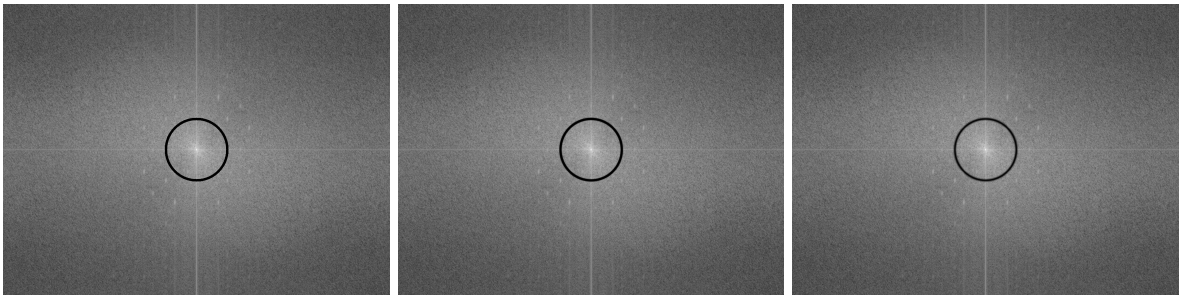


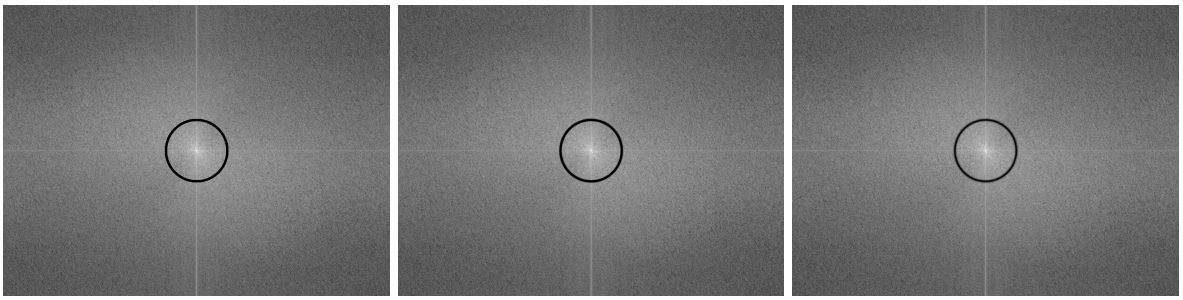
Figure 6: gaussian filter.

7.



(a) ideal filtered. (b) butterworth filtered. (c) gaussian filtered.

Figure 7: Filters applied to the spectrum transformed back from manually added noise.



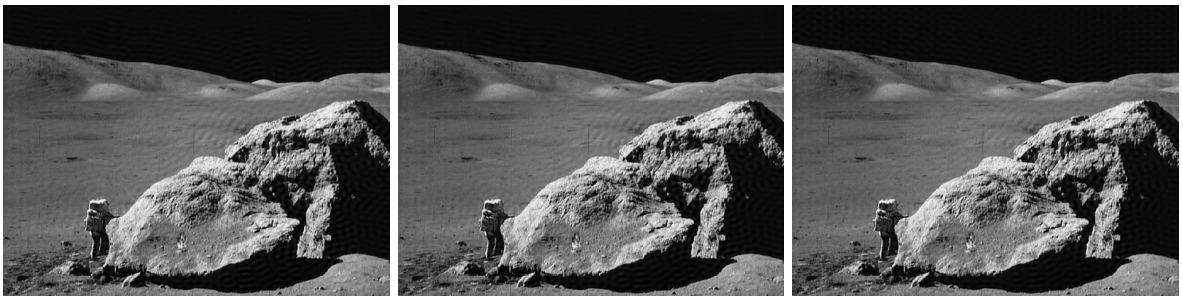
(a) ideal filtered. (b) butterworth filtered. (c) gaussian filtered.

Figure 8: Filters applied to the manually added noise spectrum directly.

8.

(a) ideal filtered image. (b) btw filtered image. (c) gaussian filtered image.

Figure 9: The resultant filtered images correspond to Figure 7

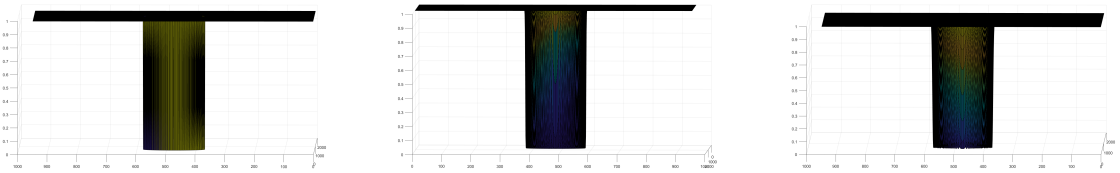


(a) ideal filtered image. (b) btw filtered image. (c) gaussian filtered image.

Figure 10: The resultant filtered images correspond to Figure 8

Compare the above images with (a) the noise added image, the filter does remove most noise noticeably. While compare with (b) the original image, there are some noise still existing in the top-left corner. If we apply the filter to manually added noise spectrum directly, we can remove it properly for the filters can fully cover the eight dots. We still can find some noise like ripples because when we apply the filters, not only were the noise frequency dots removed, but also the useful frequency.

9. In this case, there are no prominent differences between these three filters.From the cross-section image



(a) ideal. (b) butterworth. (c) gaussian.

Figure 11: cross-section of three filters.

of these three filters, it suggests the width of the band is too narrow to differentiate the these three filters and what's more the order of butterworth filter is 4 which makes the function very steep which almost approximates ideal filter. So comments on the ability to recover original image should depend on the noise pattern and parameters of filters. In other words, it is necessary to design specific filter against specific noise.