Project title: Image noise reduction through the applications of linear and non-linear filters

Project number: 03

Course number: EECE 883

Student name: Kazi Aminul Islam

Date due: 11/11/2016

Date handed in:11/11/2016

Abstract:

In this project, linear and non-linear filtering is performed to different blurred image. Corresponding performance is estimated comparing their output image. First, target image is blurred using periodic noise. Then, image is de-blurred using optimal notch filtering using different size of neighborhood. Varying neighborhood size optimal notch filtering performance is measured. Again, target image is distorted using motion blur and Gaussian additive noise. Then this noise is removed using Weiner filtering and constrained least square filtering (CLSF) technique. Their performance is improved varying parameters. Then out of focus blurring point spread function is applied to the image. Then image de-blurring performance is estimated using Weiner filtering and CLSF.

Technical Discussion:

Periodic noise filtering:

A periodic noise pattern equation is

$$r(x,y) = A\sin[2\pi u_0(x+B_x)/M + 2\pi v_0(y+B_y)/N] \quad (1)$$

Where A is the amplitude, u_0 and v_0 determines the sinusoidal frequencies with respect to x and y axis, respectively, B_x and B_y are phase displacement with respect to origin.

Optimum Notch filtering: The Fourier transform of the interference noise pattern ($N(u,v)$) is,

$$N(u,v) = H_{NP}(u,v)G(u,v) \quad (2)$$

Where $H_{NP}(u,v)$ and $G(u,v)$ are the notch filter and Fourier transform of corrupted image respectively. Corresponding spatial domain $\eta(u,v)$ is obtained as following

$$\eta(u,v) = F^{-1}\{H_{NP}(u,v)G(u,v)\} \quad (3)$$

The weighting or modulation function $w(x,y)$ can be obtained as

$$w(x,y) = \frac{\overline{g(x,y)\eta(x,y)} - \bar{g}(x,y)\bar{\eta}(x,y)}{\eta^2(x,y) - \bar{\eta}^2(x,y)} \quad (4)$$

Estimation of the original image $\hat{f}(x,y)$ can be found using following equation

$$\hat{f}(x,y) = g(x,y) - w(x,y)\eta(x,y) \quad (5)$$

Blur and Additive noise filtering:

Blurring filter degradation function H(u,v) can be given as,

$$H(u,v) = \frac{T}{\pi(ua+vb)}\sin[\pi(ua+vb)]e^{-j\pi(ua+vb)} \quad (6)$$

Where a is the displaced distance, T is time and b is motion variance.

Minimum mean square error (Weiner) filtering can be performed using following equation,

$$\hat{F}(u,v) = [\frac{1}{H(u,v)} * \frac{|H(u,v)|^2}{|H(u,v)|^2 + k}]G(u,v) \quad (7)$$

K is a specific constant added to the equation.

Frequency domain solution for constant least square filtering is following

$$\hat{F}(u,v) = [\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma|P(u,v)|^2}]G(u,v) \quad (8)$$

Where γ is a parameter which is adjusted to find a better filtering performance and P(u,v) is the Fourier transform of following function,

$$P(x,y)=\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{9}$$

This is a Laplacian operator function.

Out of focus-blurring point spread function as,

$$h(x,y)=\begin{cases} 1/\pi r^2 when, x^2+y^2 \le r^2 \\ 0, elsewhere \end{cases} \tag{10}$$

Where r is the radius of the circle of confusion.

Discussion of esults:

Figure 1.1 Input Image

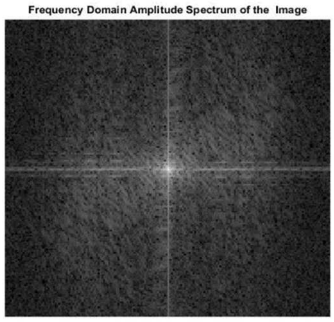

Figure 1.2: frequency spectrum of input image

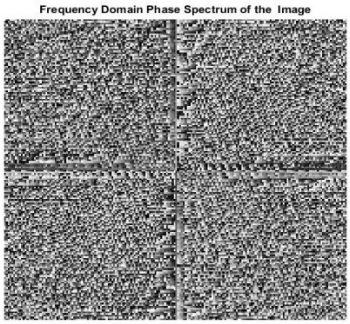

Figure 1.3: Phase Spectrum of input image

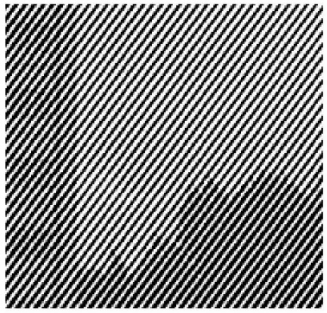

Figure 1.4: Noisy image

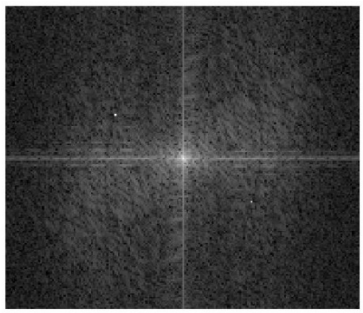

Figure 1.5: Frequency Spectrum of Noisy Image

Figure 1.6: De-noised Image

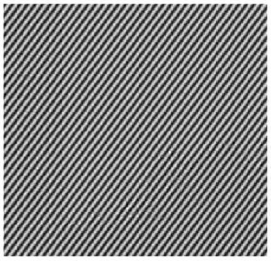

figure 1.7: Noise pattern

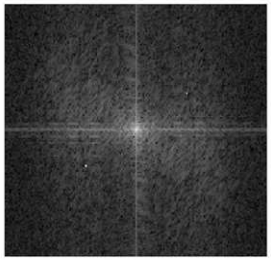

figure 1.8: Noise Fourier spectrum

First, the image was blurred using periodic noise in Figure1.3. From the frequency spectrum, we may find the noise spectrum Figure1.4. Then, Butterworth notch reject filter is applied to the image. From the noise spectrum the noise location was determined. From Butterworth Notch reject filter and noisy image, noise pattern is determined. For different neighborhood size, weight portion of the noise is determined. Weight is calculated based on average value of the neighborhood noise pattern and noisy image. For different neighborhood size the weight is calculated then at the end de-noised image is determined in Figure 1.6.

figure 2 1:original image

figure 2 2 Motion blurred image

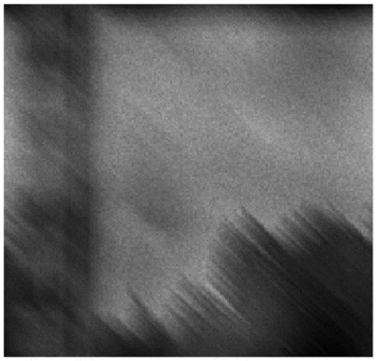

figure 2 3: Motion blur and additive blur image

figure 2 4: image restored using Weiner filtering

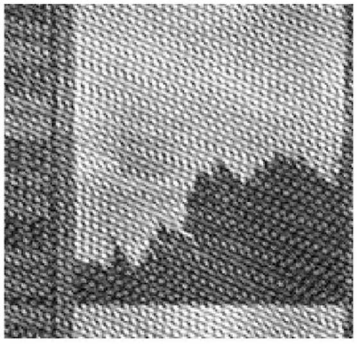

figure 2 5: Image restored using constrained least square filtering

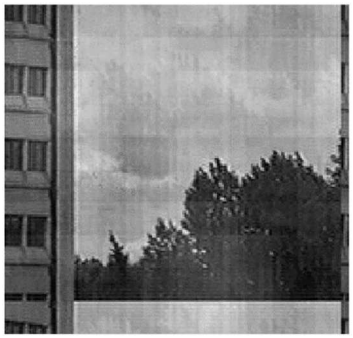

figure 2 6: Image restored using constrained least square filtering with different co-efficient then figure 2.5

First, the image is blurred in Figure 2.2 using motion blur from given equation. The similar a, b and T value from text book is used to apply motion blur in the image. Gaussian additive noise is applied to the image in Figure 2.3. Then, image noise is minimized with applying Weiner filtering. Here, k value is varied to find better de-blurred image. Constrained least squares filtering algorithm is applied to find better de-

blurred image according to specified equation. Gamma value is varied to find the optimum result. Here, Newton-Raphson algorithm is used to find the optimum Gamma value.

figure 3. 1:original Image

figure 3. 2: Out of Focus blurred image

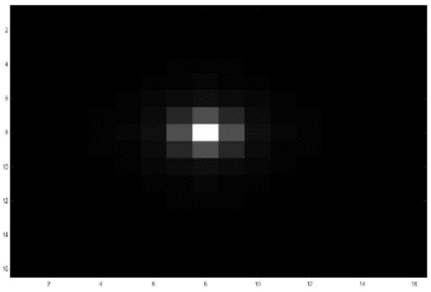

figure 3. 3: Blurred kernel

figure 3. 4: Gaussian additive noise

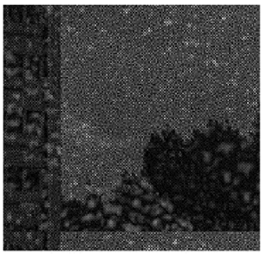

figure 3. 5: Corresponding Weiner filtering output

figure 3. 6: Constrained Least Square filtering

figure 3. 7: lessen Gaussian Additive Noise

figure 3. 8: Corresponding Weiner filtering output

figure 3. 9: Constrained Least Square filtering

Here, image is de-blurred using out of focus blurring function point spread function. Then additional Gaussian additive noise is added to image in figure 3.4. To measure de-noising performance of the image with constrained least square filtering, first Weiner filtering is performed to the image in figures 3.5 and 3.8. Then image is de-noised using constrained least square filtering (CLSF) in figure 3.6 and 3.9. It is found that CLSF performance is better than Weiner filter.

In conclusion, it is found that constrained least square filtering technique performance is better than Weiner filter. In the optimum notch filtering text book image is compared with the project image. Similar set of noise pattern and frequency spectrum is found for similar operation.

Appendix:

Methods	PSNR	ISNR
Notch Filtering	23.8347	20.8244
Weiner	12.1728	0.0912
CLSF	12.2839	-0.7297
PSF-Weiner	13.2808	22.2612
PSF-CLSF	19.5057	28.4884

Optimum notch filter:

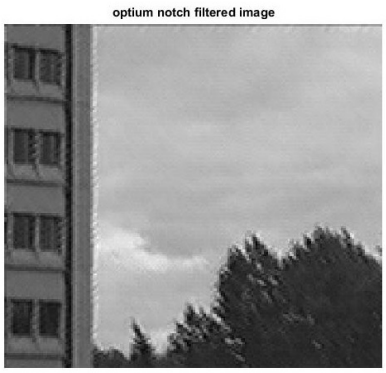

Figure 3: Optimum notch filtered image for Neighborhood 4

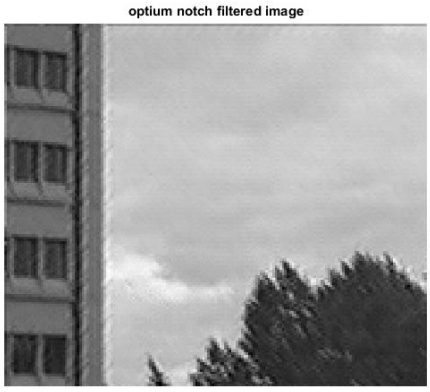

Figure 4: Optimum notch filtered image for Neighborhood 5

Weiner filter Image:

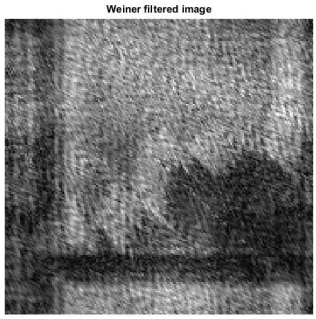

Figure 5: Weiner filtered image varying constant k

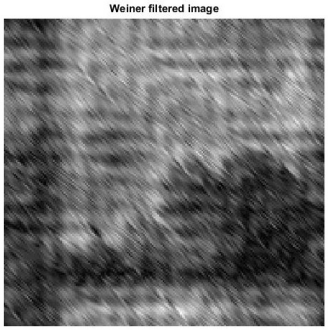

Figure 6: Weiner filtered image varying constant k

Appendix-Matlab Code:

optimum_notch_filtering.m

```
clear all;
close all;
I=imread('SKY.PGM');

I=im2double(I);
figure();imshow(I);
[M, N]=size(I);
F=fft2(double(I));
figure;imshow(log(1+abs(fftshift(F))),[]); title('Frequency Domain Amplitude
Spectrum');
figure();imshow(angle(fftshift(F)),[]); title('Frequency Domain Phase
Spectrum ');

B2=double(I);

A=1;
u=25;
v=35;
bx=29;
by=37;

for i=1:M,
  for j=1:N,

      p2(i,j)=(A*sin((2*pi*(u./M)*(i+bx))+(2*pi*(v/N)*(j+by))));

  end
end
g=p2+double(I);
G=fft2(g);

figure();imshow((g));
figure;imshow(log(1+abs(fftshift(G))),[]);

%notch
[p,q]=size(I);
M=p;
N=q;

x=0:(p-1);
y=0:(q-1);
xin=find(x>p/2);
x(xin)=x(xin)-p;
yin=find(y>q/2);
y(yin)=y(yin)-q;
[v,u]=meshgrid(y,x);

%notch filtering
order = 2;
centers = [62 58];
centers2 = [121 117];%centers of the highpass filters.
```

```
for i = 1: p
   for j = 1:q
        D0(i,j)=sqrt(((u(i,j)-M/2).^2)+(((v(i,j)-N/2).^2)));
        Dk(i,j)=sqrt(((u(i,j)-M/2-centers(1)).^2)+(((v(i,j)-N/2-
centers(1)).^2)));
        Dkm(i,j)=sqrt(((u(i,j)-M/2+centers2(1)).^2)+(((v(i,j)-
M/2+centers2(1)).^2)));


    end
end

Hnr=1./((1+((D0./Dk).^(2*order))).*(1+((D0./Dkm).^(2*order))));
Hnp=Hnr;
%figure();imshow(log(1+abs(fftshift(Hnr))),[]);

I_noise_FT=fft2(g);
I_nfil=(Hnp).*I_noise_FT;
n_xy=real(ifft2(double(I_nfil)));

figure();imshow(n_xy,[]);
figure();figure;imshow(log(1+abs(fftshift(fft2(n_xy)))),[])

%weights

%{
I=input image
B=noiseyimage
np=filter for corresponding image
%}

I=I;
B=g;
np=n_xy;
blockSize=2;
[numRow, numCol] = size(I);

numRow_flr1=floor(numRow/(blockSize*blockSize));
numCol_flr1=floor(numCol/(blockSize*blockSize));
numRow_flr=numRow_flr1*blockSize;
numCol_flr=numCol_flr1*blockSize;

I_resize=I(1:(numRow_flr*blockSize),1:(numCol_flr*blockSize));
B_resize=B(1:(numRow_flr*blockSize),1:(numCol_flr*blockSize));
np_resize=np(1:(numRow_flr*blockSize),1:(numCol_flr*blockSize));

a=blockSize*ones(1,numRow_flr);
b=blockSize*ones(1,numCol_flr);
B_blckImgsize=mat2cell(B_resize,a,b);
np_blckImgsize=mat2cell(np_resize,a,b);
clear w_b;
for i=1:1:numRow_flr
    for j=1:1:numCol_flr
```

```
        s_g=0;
        s_n=0;
        gn=0;
        sum2=0;

s_g=double(reshape(cell2mat(B_blckImgsize(i,j)),blockSize*blockSize,1));

s_n=double(reshape(cell2mat(np_blckImgsize(i,j)),blockSize*blockSize,1));
        gn=double(double(s_g).*s_n);
        np2=double(s_n.*s_n);

        g1=double(mean(s_g));
        ns=double(mean(s_n));
        ns2=double(mean(np2));
        gns=double(mean(gn));
        w=double(gns-(g1*ns))/(ns2-(ns^2));

        sum2(1:blockSize,1:blockSize)=w;
        w_b(i,j)=mat2cell(sum2,blockSize,blockSize);
    end
end

w2=cell2mat(w_b);

I2=double((B_resize)-(w2.*np_resize));
I2=I2;
figure();imshow(I2,[]);title('optium notch filtered image')

I3=I(1:172,1:180);
%1=g(1:172,1:180);
[peaksnr, snr] = psnr(I3, I2)

Idiff1=I3-g1;
idiff2=I3-I2;
mean1=mean2(Idiff1.*Idiff1);
mean2=mean2(idiff2.*idiff2);
isnr=10*log(mean1/mean2)/log(10)
```

weiner_filtering_final.m

```
clear all;
close all;
I=imread('SKY.PGM');

%I=im2double(I);
figure();imshow(I);
[M, N]=size(I);
F=fft2(double(I));

T=1;
a=0.1;
b=0.1;

[p,q]=size(I);
```

```
x=1:(p);
y=1:(q);
xin=find(x>p/2);
x(xin)=x(xin)-p;
yin=find(y>q/2);
y(yin)=y(yin)-q;
[v,u]=meshgrid(y,x);
for n = 1: p
   for m = 1:q
       H(n,m)=(T/(pi*(u(n,m)*a+v(n,m)*b))).*(sin(pi*(u(n,m)*a+v(n,m)*b)))...
           *exp(-(1j*pi*(u(n,m)*a+v(n,m)*b)));
    end
end

F=fft2(I);
H(isnan(H)) =0;
G=F.*(H);
g=real(ifft2(double(G)));
figure();imshow(uint8(g));
%gaussian noise
mean1=0;   % value ofmean
gstd=abs(g*0.05); % standard deviation 5%
nse= mean1 + gstd.*randn(size(g));
out_noise=g+nse;
figure();imshow(uint8(out_noise));
G2=fft2(out_noise);
I_nfil=(H).*G2;
%figure();imshow(log(1+abs(fftshift(I_nfil))),[]);
n_xy=real(ifft2(double(I_nfil)));
mean_n_xy=mean(reshape(n_xy,p*q,1));
std_n=std2(n_xy);
n2=M*N*((mean_n_xy^2)+(std_n^2));
H2=abs(H).^2;
[m,n]=size(I); % image sizes
mn=m*n;
tmp=[1:m]'*ones(1,n)+ones(m,1)*[1:n];
H2=H.^2;
k1=.1;
%k1=n2;
Hhs=((conj(H)).*G2)./(H2+k1);
hhs2=abs(ifft2(Hhs));

%figure();imshow(uint8(hhs2));

gn2=uint8(hhs2);
gn3=zeros(174,182);
gn3(1:174,1:14)=gn2(1:174,169:182);
gn3(1:174,15:182)=gn2(1:174,1:168);

figure();imshow(uint8(gn3));title('Weiner filtered image')

%imshow(im2double(gn3),[])

Gn=fft2(gn3);
R=G2-(H.*Gn);
r=abs(ifft2(R));
```

```
r2=std2(r.^2);
[peaksnr, snr] = psnr(I, uint8(gn3))
Idiff1=I-uint8(out_noise);
idiff2=I-uint8(gn3);
mean1=mean2(Idiff1.*Idiff1);
mean2=mean2(idiff2.*idiff2);
isnr=10*log(mean1/mean2)/log(10)
```

CLSF_final.m

```
clear all;
close all;
I=imread('SKY.PGM');

%I=im2double(I);
figure();imshow(I);
[M, N]=size(I);
F=fft2(double(I));

T=1;
a=0.1;
b=0.1;

[p_cons,q]=size(I);
x=1:(p_cons);
y=1:(q);
xn1=find(x>p_cons/2);
x(xn1)=x(xn1)-p_cons;
yn1=find(y>q/2);
y(yn1)=y(yn1)-q;
[v,u]=meshgrid(y,x);


for N = 1: p_cons
   for M = 1:q
       H(N,M)=(T/(pi*(u(N,M)*a+v(N,M)*b))).*(sin(pi*(u(N,M)*a+v(N,M)*b)))...
           *exp(-(1j*pi*(u(N,M)*a+v(N,M)*b)));
    end
end
%F=fft2(I);
H(isnan(H)) =0;
G=F.*(H);
g=real(ifft2(double(G)));
figure();imshow(uint8(g));

%gaussian noise

mean1=0;   % value ofmean
gd=abs(g*0.0005); %
nse= mean1 + gd.*randn(size(g));
out_noise=g+nse;
figure();imshow(uint8(out_noise));
G2=fft2(out_noise);
I_nfil=(H).*G2;
%figure();imshow(log(1+abs(fftshift(I_nfil))),[]);
```

```
n_xy=real(ifft2(double(I_nfil)));
H2=abs(H).^2;
%constrained linear filtering
 [M,N]=size(I);
MN=M*N;
tamp_data=[1:M]'*ones(1,N)+ones(M,1)*[1:N];
H2=H.^2;
%gamma=.000009;
gmm=.000000009;

p_cons=[0 -1 0; -1 4 -1; 0 -1 0];
f_cons=fft2(p_cons.*tamp_data(1:size(p_cons,1),1:size(p_cons,2)),174,182);
Hcons=conj(H).*G./(H2+gmm*abs(f_cons).^2);
hcons=abs(ifft2(Hcons));
%figure();imshow(uint8(hcons));
gn2=uint8(hcons);
gn3=zeros(174,182);
gn3(1:174,1:14)=gn2(1:174,169:182);
gn3(1:174,15:182)=gn2(1:174,1:168);
figure();imshow(uint8(gn3));title('CLSF filtered image')
%figure();imshow(gn3,[]);
%iteraion
Rr=abs(G-H.*Hcons);
r=real(ifft2(Rr));
r2=(sum(sum(abs(r).^2))-r(1,1)^2)/MN;
%iteration2
mean_n_xy=mean2(n_xy);
std_n=std2(n_xy);
n2=M*N*((mean_n_xy.^2)+(std_n.^2))
Gn=fft2(gn3);
R=G2-(H.*Gn);
r=abs(ifft2(R));
r2=std2(r.^2)
[peaksnr, snr] = psnr(I,uint8(gn3))
Idiff1=double(I)-double(out_noise);
idiff2=double(I)-double(uint8(gn3));
mean1=mean2(Idiff1.*Idiff1);
mean2=mean2(idiff2.*idiff2);
isnr=10*log(mean1/mean2)/log(10)
```

psf_weiner_final.m

```
close all
clear all
I=im2double(imread('SKY.PGM'));
[M,N]=size(I);
figure();imshow(I,[]);title('Original Image');

%Kernel

pdfsize = 16;
pkrnl = zeros(pdfsize);

mid = pdfsize/2;
[U, V] = meshgrid(1:pdfsize);
pr=sqrt((U-mid).^2 + (V-mid).^2);
```

```
pkrnl = (1./(pi.*pr.^2));
pkrnl(find(pkrnl == inf)) = 1;
figure, imagesc(pkrnl);colormap('gray');
krnl_img = zeros(M,N);
krnl_img(1:pdfsize, 1:pdfsize) = pkrnl;
F=fft2(double(I));
F_krnl = fft2(krnl_img);
F_krnl(find(F_krnl == 0)) = 1e-6;
F_blur = F.*F_krnl;
blur_img = double(ifft2(F_blur));
figure, imshow(blur_img,[]);title('PSF blurred')

%gaussian noise
mean1=0;
g=blur_img;
gstd=abs(g*0.005);
nse= mean1 + gstd.*randn(size(g));
out_noise=g+nse;
figure();imshow(out_noise,[]);
G2=fft2(out_noise);
%weiner filtering
H=F_krnl;
G=F_blur;
H2=H.^2;
k1=.1;
Hhcls=((conj(H)).*G2)./(H2+k1);
hcls2=abs(ifft2(Hhcls));
%figure();imshow(hcls2,[]);
gn2=(hcls2);
gn3=zeros(174,182);
gn3(1:174,1:14)=gn2(1:174,169:182);
gn3(1:174,15:182)=gn2(1:174,1:168);
figure();imshow(gn3,[]);title('Weiner filtered image')
[peaksnr, snr] = psnr(I,gn3)

Idiff1=double(I)-double(out_noise);
idiff2=double(I)-double(gn3);
mean1=mean2(Idiff1.*Idiff1);
mean2=mean2(idiff2.*idiff2);
isnr=10*log(mean1/mean2)/log(10)
```

psf_constrained_final.m

```
close all
clear all
I=im2double(imread('SKY.PGM'));
[M,N]=size(I);
figure();imshow(I,[]);title('Original Image');
%Kernel
pdfsize = 16;
pkrnl = zeros(pdfsize);
mid = pdfsize/2;
[U, V] = meshgrid(1:pdfsize);
pr=sqrt((U-mid).^2 + (V-mid).^2);
pkrnl = (1./(pi.*pr.^2));
pkrnl(find(pkrnl == inf)) = 1;
```

```
figure, imagesc(pkrnl);colormap('gray');
krnl_img = zeros(M,N);
krnl_img(1:pdfsize, 1:pdfsize) = pkrnl;
F=fft2(double(I));
F_krnl = fft2(krnl_img);
F_krnl(find(F_krnl == 0)) = 1e-6;
F_blur = F.*F_krnl;
blur_img = double(ifft2(F_blur));
figure, imshow(blur_img,[]);title('PSF blurred')
%gaussian noise

mean1=0;   % value ofmean
g=blur_img;
gstd=abs(g*0.02); % standard deviation 5%
nse= mean1 + gstd.*randn(size(g));
out_noise=g+nse;
figure();imshow(out_noise,[]);
G2=fft2(out_noise);
%constrained linear filtering
H=F_krnl;
G=F_blur;
[M,N]=size(I);
MN=M*N;
tmp=[1:M]'*ones(1,N)+ones(M,1)*[1:N];
H2=H.^2;
gamma=.000000009;
p_cons=[0 -1 0; -1 4 -1; 0 -1 0];
consP=fft2(p_cons.*tmp(1:size(p_cons,1),1:size(p_cons,2)),174,182);
Hcons=(H).*G2./(H2+gamma*abs(consP).^2);
hcons=abs(ifft2(Hcons));
figure();imshow(hcons,[]);title('CLSF filtered image')
gn3=hcons;
Rr=abs(G-H.*Hcons);
r=real(ifft2(Rr));
r2=(sum(sum(abs(r).^2))-r(1,1)^2)/MN;
%iteration2
n_xy=out_noise;
mean_n_xy=mean2(n_xy);
std_n=std2(n_xy);
n2=M*N*((mean_n_xy.^2)+(std_n.^2))
Gn=fft2(gn3);
R=G2-(H.*Gn);
r=abs(ifft2(R));
r2=std2(r.^2)
[peaksnr, snr] = psnr(I,gn3)
 Idiff1=double(I)-double(out_noise);
idiff2=double(I)-double(gn3);
mean1=mean2(Idiff1.*Idiff1);
mean2=mean2(idiff2.*idiff2);
isnr=10*log(mean1/mean2)/log(10)
```