EE 475/575
DIGITIAL IMAGE PROCESSING
PROJECT 4
IMAGE RESTORATION

Name: Harsh Dubey

Date: 11/10/2019

Instructor: Dr. Dennis Helder

The purpose of this project is to practice the image restoration techniques learned in class by applying them to a noisy and blurred digital image. The image is obtainable on the D2L website. This is an image of a person holding a checkerboard pattern. It was obtained with a defocused digital camera. You are to restore this image using the Wiener filter. Use the following procedure.

1. Estimate the horizontal edge spread function (ESF) using the checkerboard. From the ESF generate the horizontal PSF. Show plots of both and list the steps you followed to produce these estimates.

   Steps:

   1. Checkerboard image was chosen to estimate the ESF. A sweep of pixel values at location x= [866:1302] and y= [863:1299] was performed. The checkerboard image was cropped horizontally and vertically and a total of 437 samples were taken to compute ESF in both directions.
   2. The mean of all row and column pixel values were obtained while performing the sweep. The values were stored in a one-dimensional array.
   3. Fermi equation was applied to fit the noisy the ESF, the fermi function was derived from pg. 19 of Dr. Dennis Helder's research paper "Generic Sensor Modeling using Pulse Method" at 2005. The negative sign on the "1" was changed to positive sign for reversed sweep of the checkerboard. Minus one was changed to Plus one to account for the reversed sweep of the checkerboard area.

   Results:

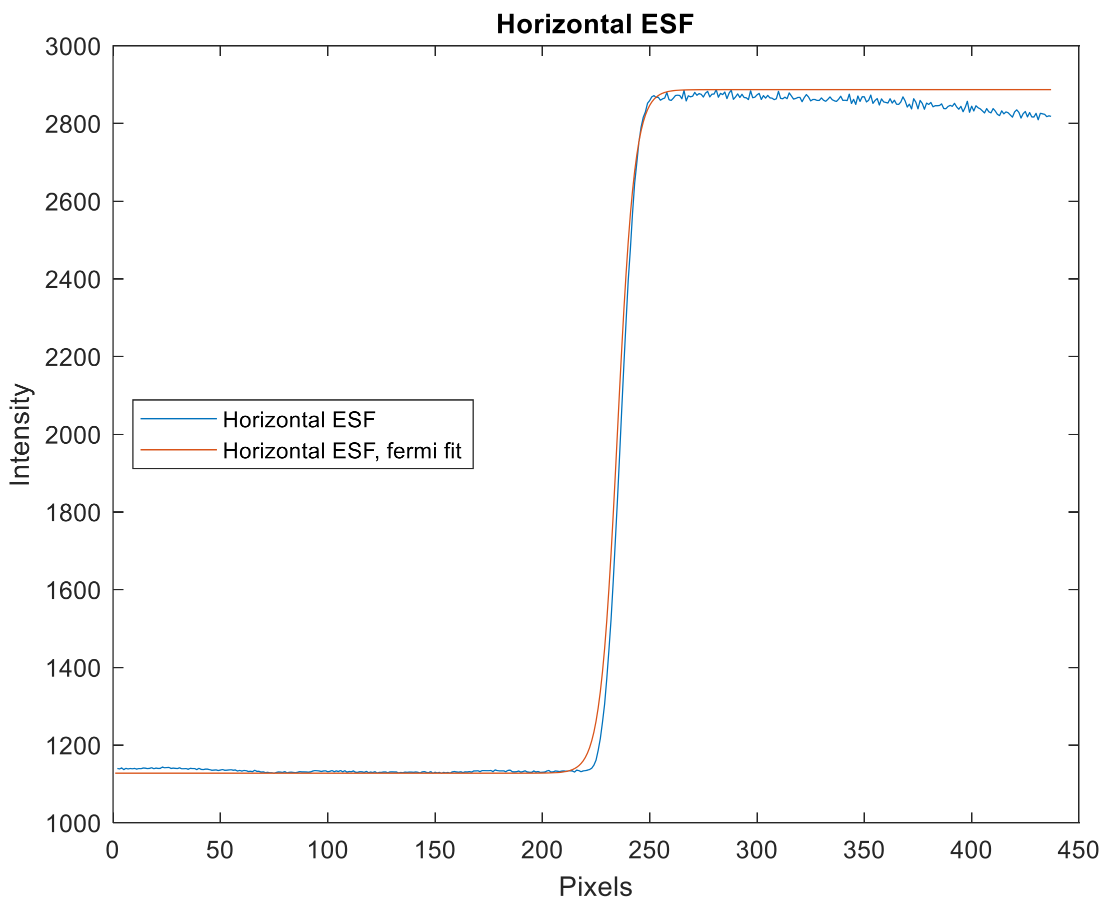

Figure 1: Horizontal edge spread function and Fermi fit

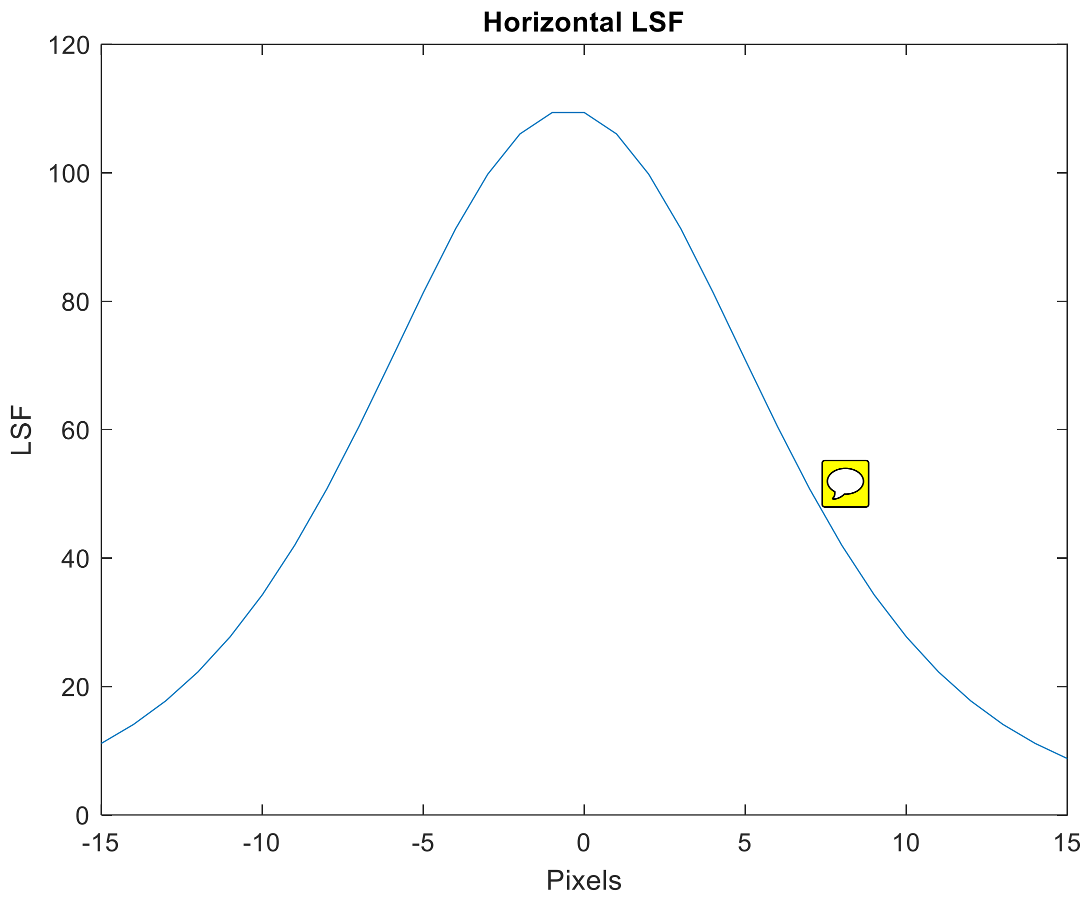

Figure 2: Horizontal line spread function

2. Repeat the first step to estimate the vertical ESF and PSF

   Steps:

   1. The ESF was differentiated using MATLAB diff() function to obtain LSF.

   Results:

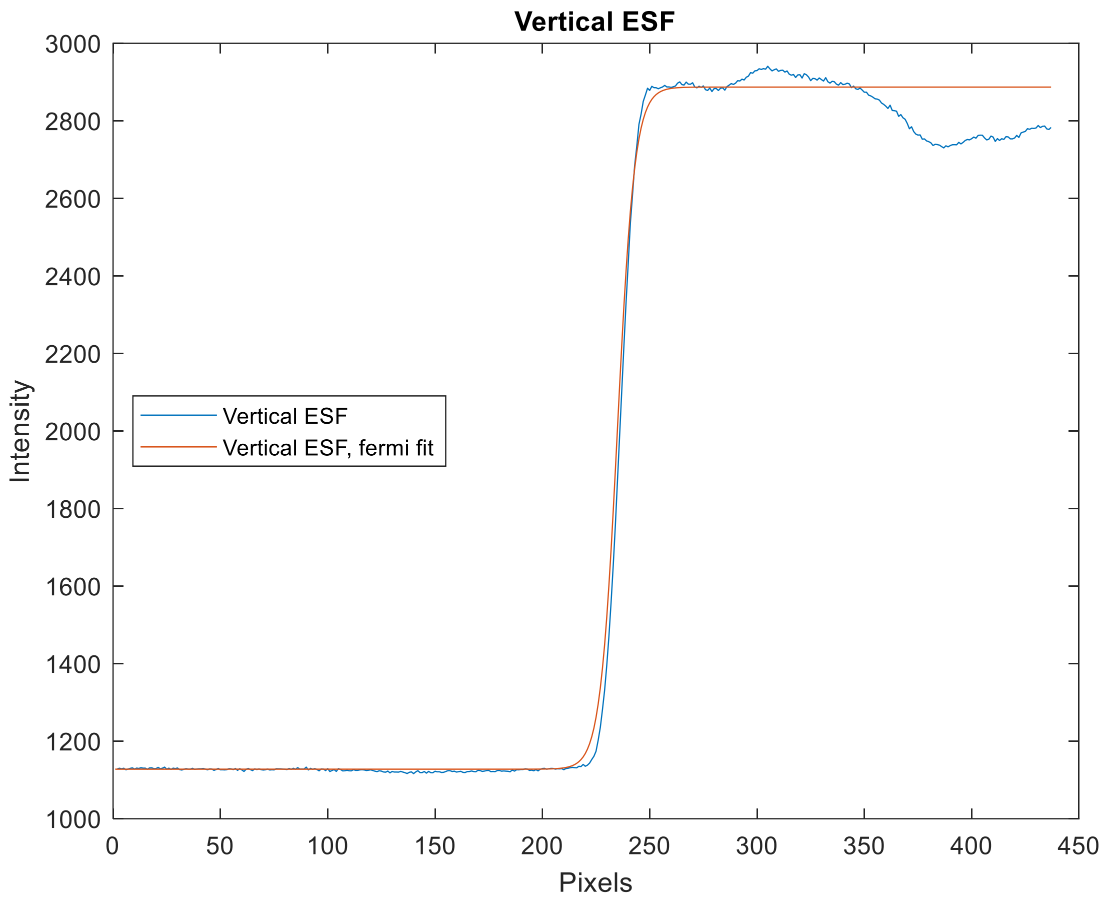

Figure 3: Vertical edge spread function and Fermi fit

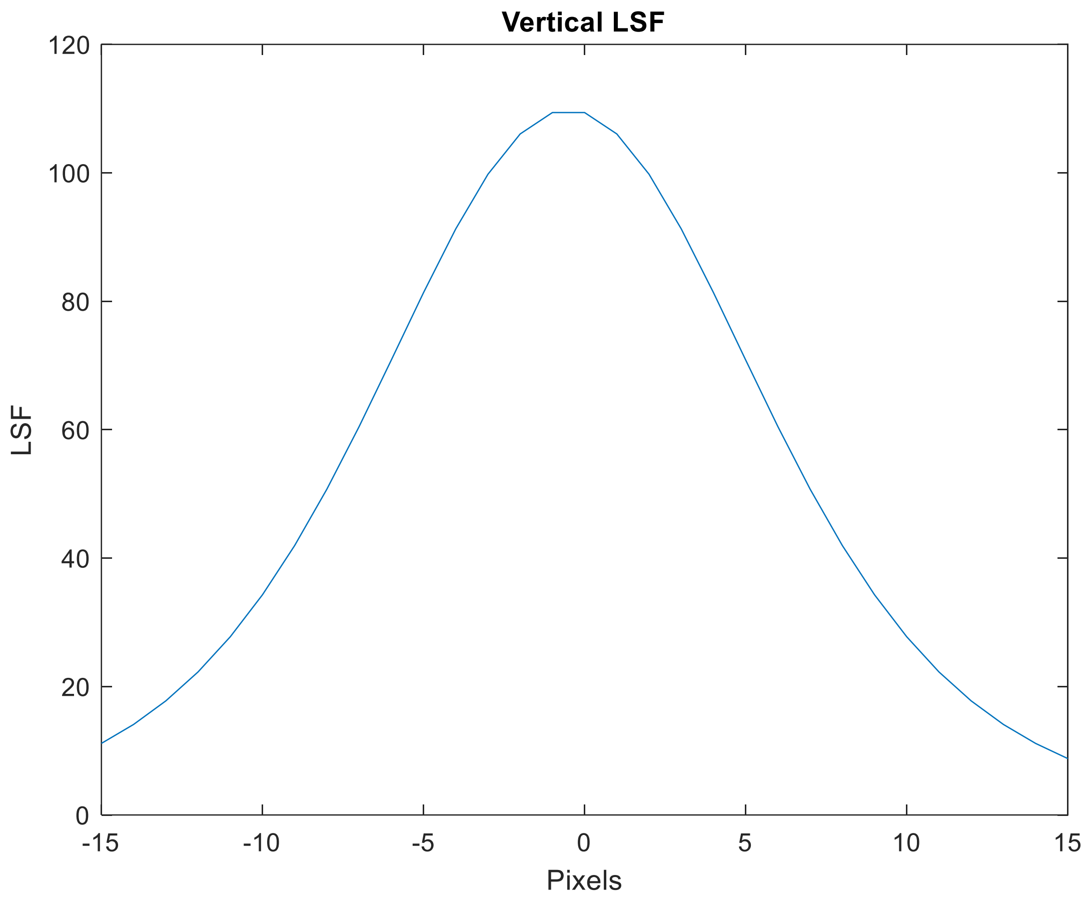

Figure 4: Vertical line spread function

3. Form the 2-D PSF using either separability or radial symmetry. Describe and justify your approach.

   Steps:
   1. On visual inspection the LSFs, they looked radially symmetric, with same width, same slope and same magnitude. Hence, a 2-D PSF was created from a radius vector and by circularly interpolating the LSF.
   2. Later, the 2-D PSF was normalized by dividing it by the sum of all matrix elements.

Result:

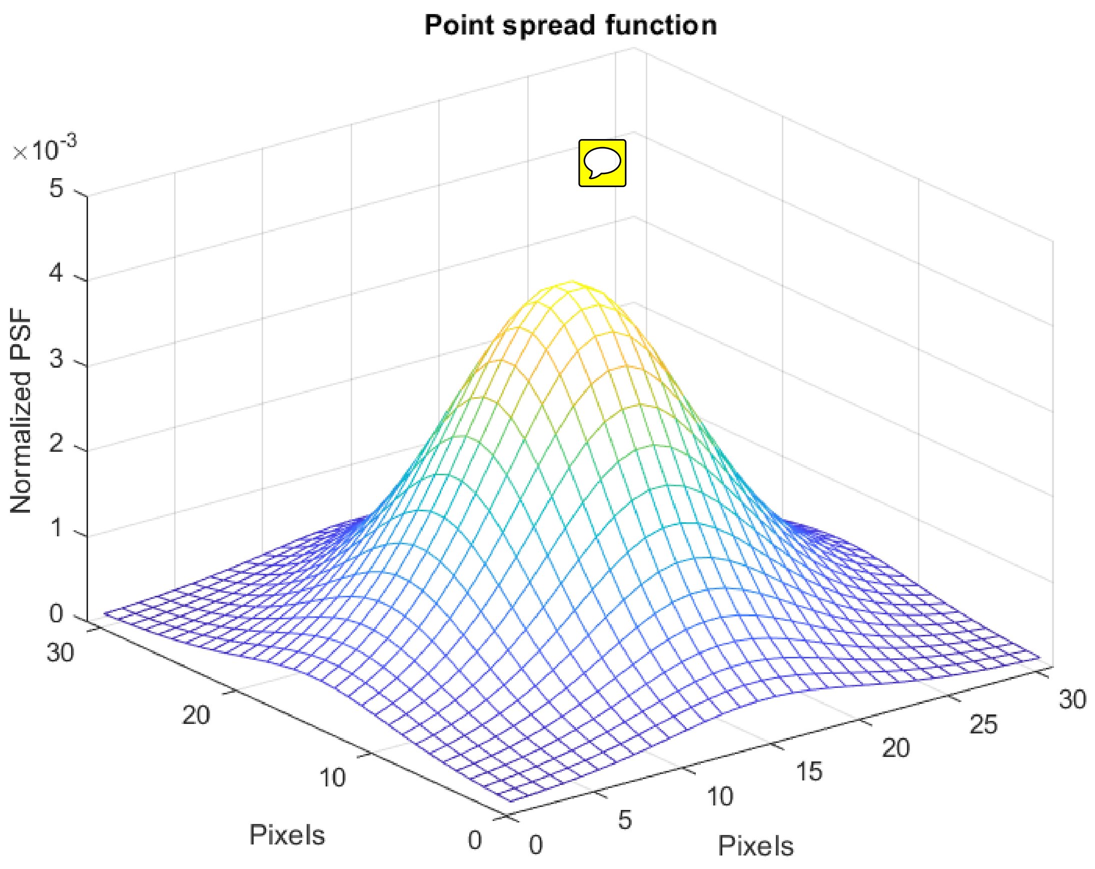

Figure 5: Point spread function

4. Estimate the noise in the image. Describe the assumptions you made and the process you followed. Be sure to list the pertinent parameters of your noise model.

Assumptions: Image and noise are uncorrelated. Noise is white noise with zero mean.

Steps:

1. Three smooth regions on the wall were selected, so that there is no sudden intensity change and noise estimation can be done without any error. Later, histogram was computed for all three regions and all three of them had a Gaussian shape.
2. The standard deviation of each histogram were computed by using std() command in MATLAB, they were also manually verified by examining the histograms. The average value of all three standard deviation was computed. Also, a large smooth area was selected to calculate the standard deviation using std2() command in MATLAB. Later, this value was used in other parts of the project.

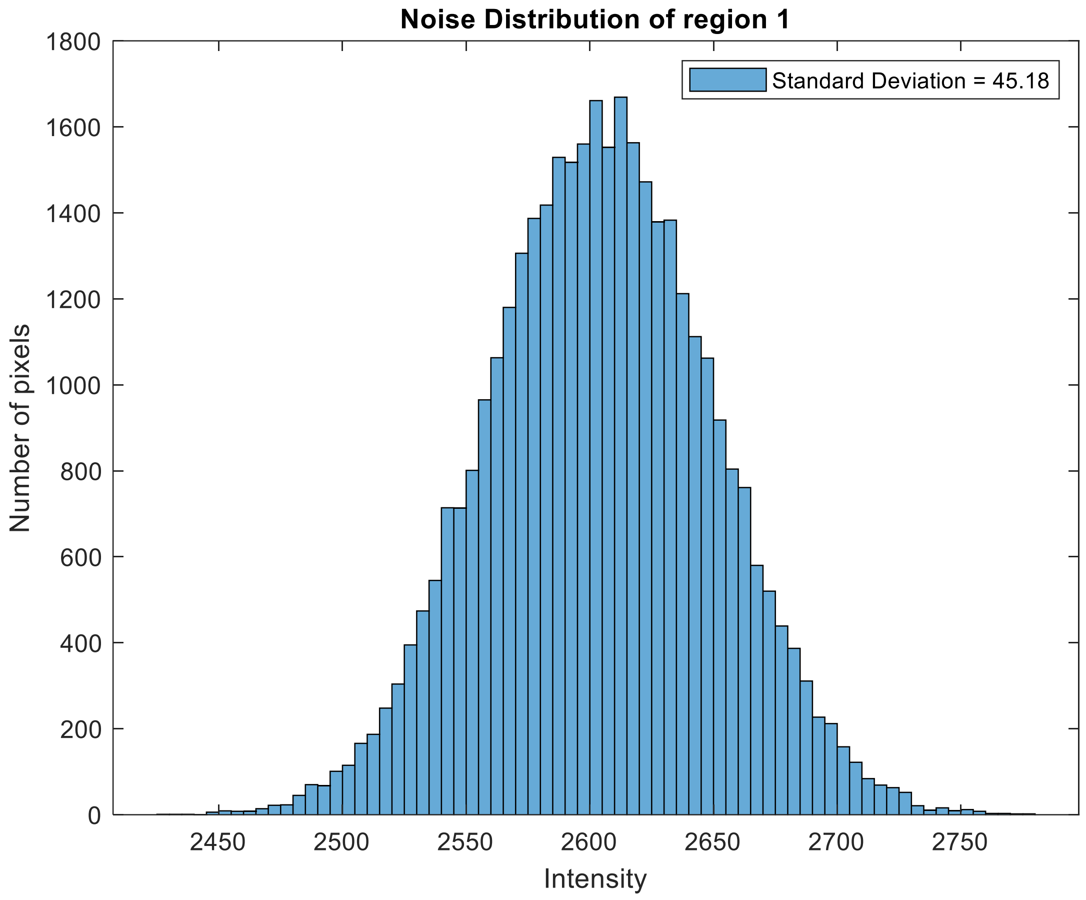

Figure 6: Histogram at region 1

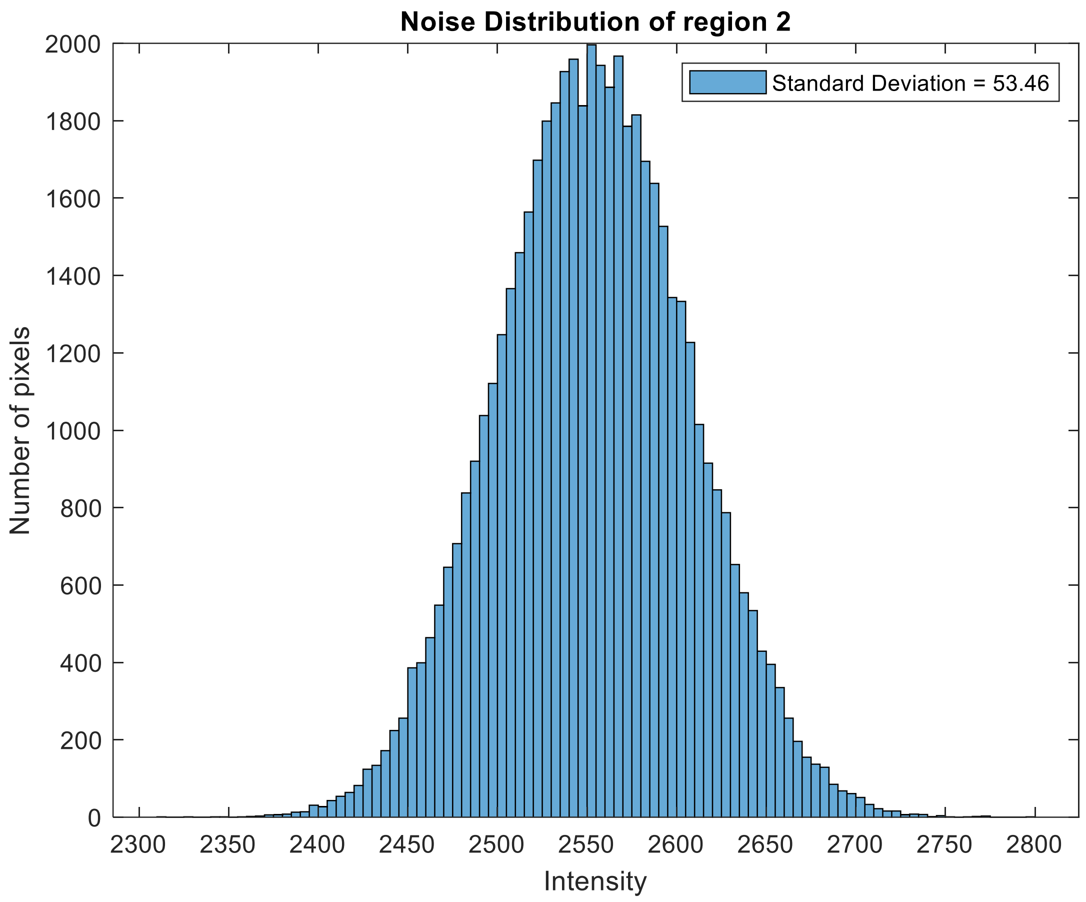

Figure 7: Histogram at region 2

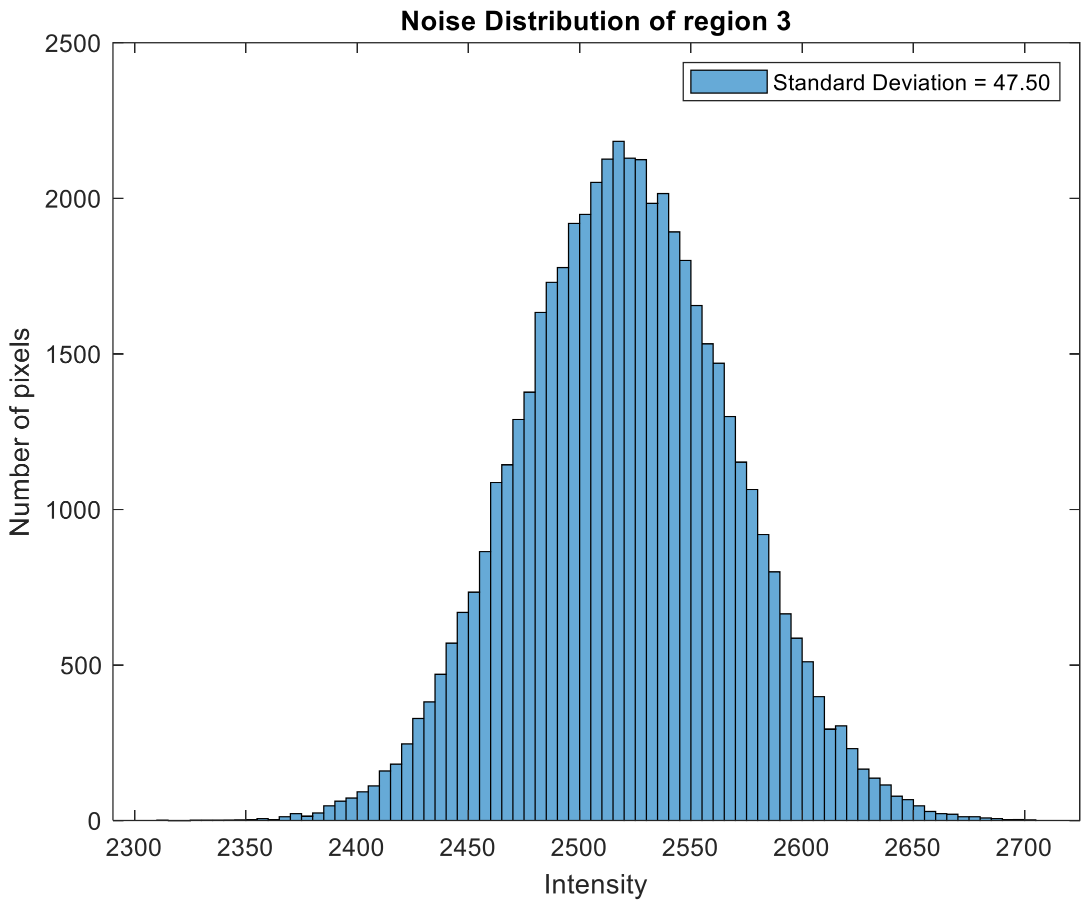

Figure 8: Histogram at region 3

Table 1: Parameters of each noise model

Region	Standard Deviation
1	45.18
2	53.46
3	47.50
Average STD	48.71
Noise STD calculated using std2 MATLAB function	49.49

5. Restore the image using the Wiener filter. Derive your Wiener filter based on the theory developed in class. Use the blur and noise models you derived from the image. Do not use Matlab's 'deconvwnr' function. List the steps that you followed in sufficient detail so that the instructor could replicate them. Show before and after restoration images on the same page. Show the degree of restoration that occurred by plotting horizontal and vertical edge profiles before and after restoration (before and after on the same plot; horizontal and vertical on different plots). Estimate the change in noise level by comparing noise estimates before and after restoration. Describe the improvement qualitatively as well.

Steps:

1. Fourier transform was computed for blurry Chris.
2. Then transfer function of Wiener filter was taken from pg. 353 of signal processing book Rafael C. Gonzalez, Richard E. Woods - Digital Image Processing (2008, Prentice Hall):

$$W(u,v)=\left[\frac{H^*(u,v)}{|H(u,v)|^2+S_\eta(u,v)/S_f(u,v)}\right]$$

   Where, H(u,v) is the fourier transform of the PSF, Sn(u,v) is power specturm of the noise, Sf(u,v) is power spectrum of the original object. NSR was computed by dividing the power pecturm of the noise by the mean of the blurred image. Conj() command in MATLAB was used to compute complex conjugate, and abs().^2 was used to calculate the magnitude.

3. After H(u,v) was derived, the blurring function was was multiplied with the fourier transform of blurred image. Later, inverse FFT was taken to obtain the spatial domain image.
4. Next, both horizontal and vertical ESFs before and after filtering were compared. Later, the noise standard deviation was computed for the filtered image and was compared to the standard deviation of the noise form the original image.

Results:

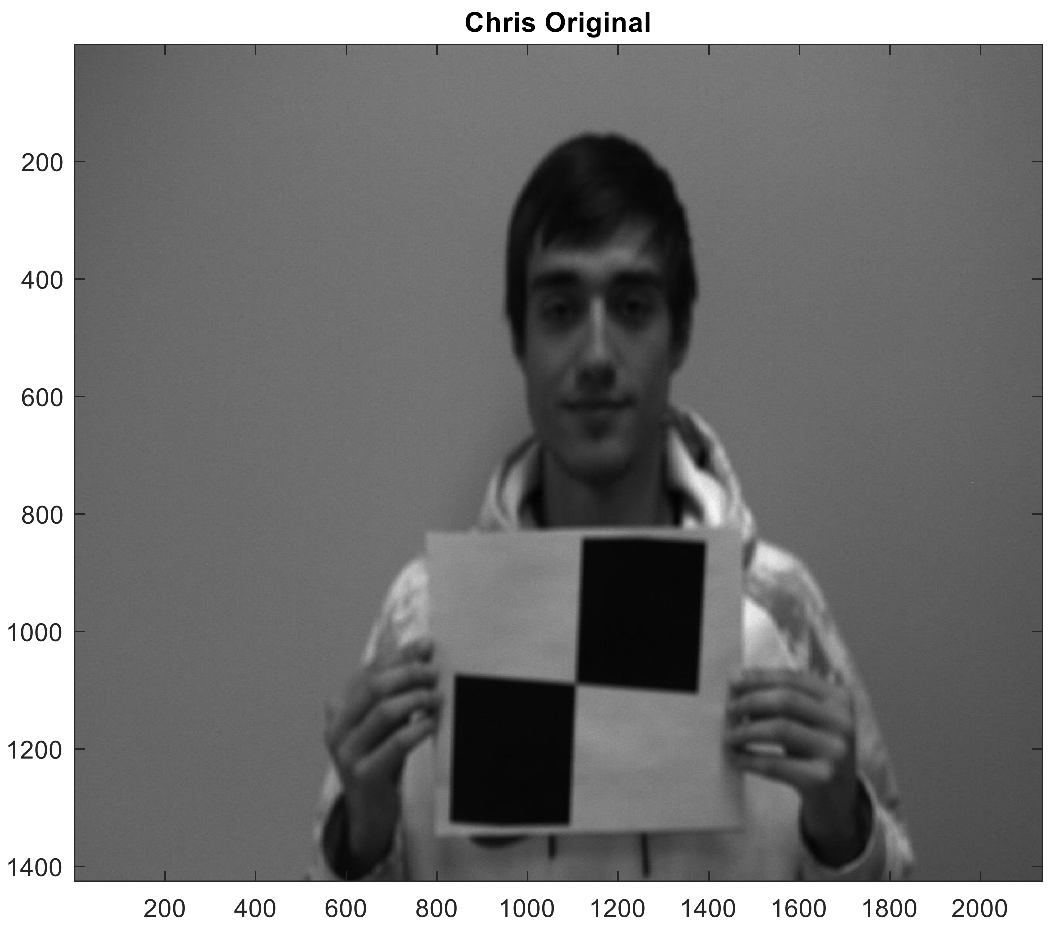

Figure 9: Chris Original

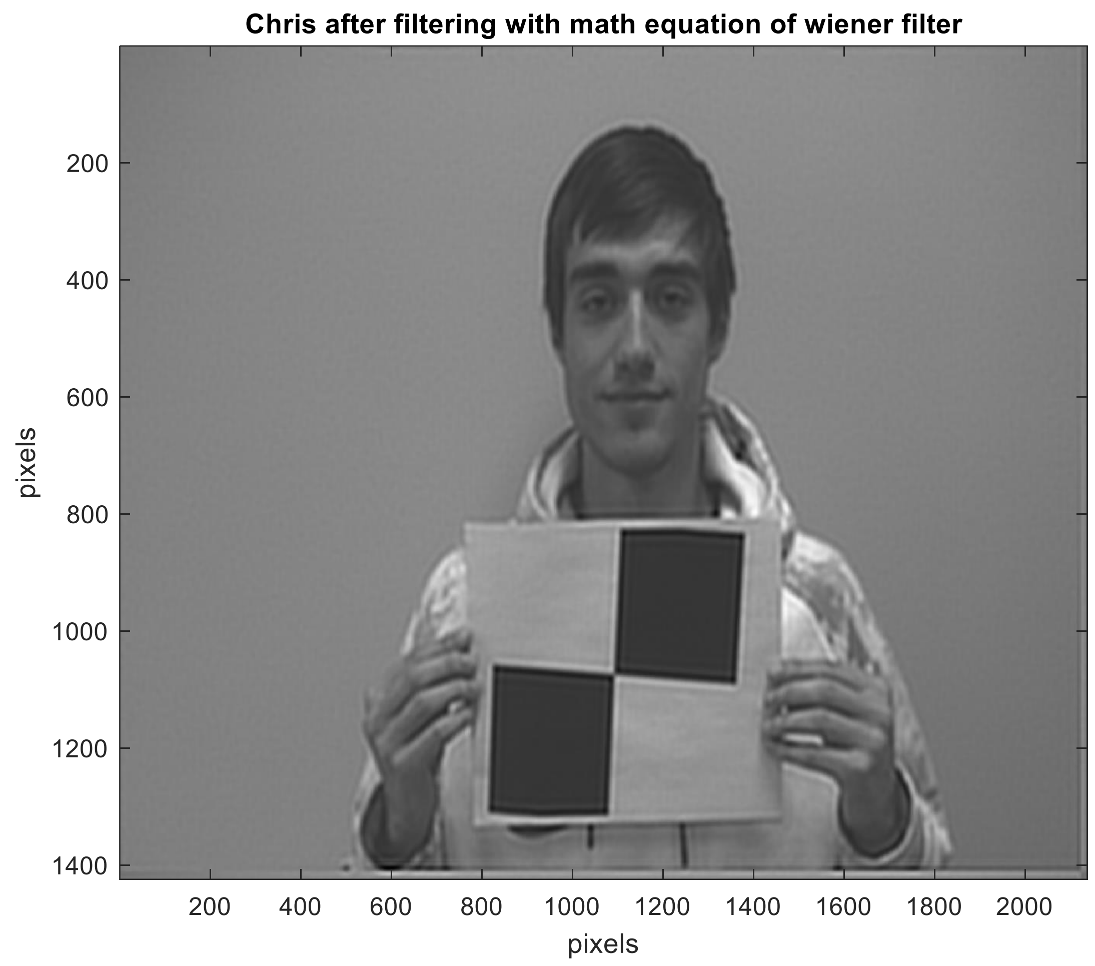

Figure 10: Chris Filtered

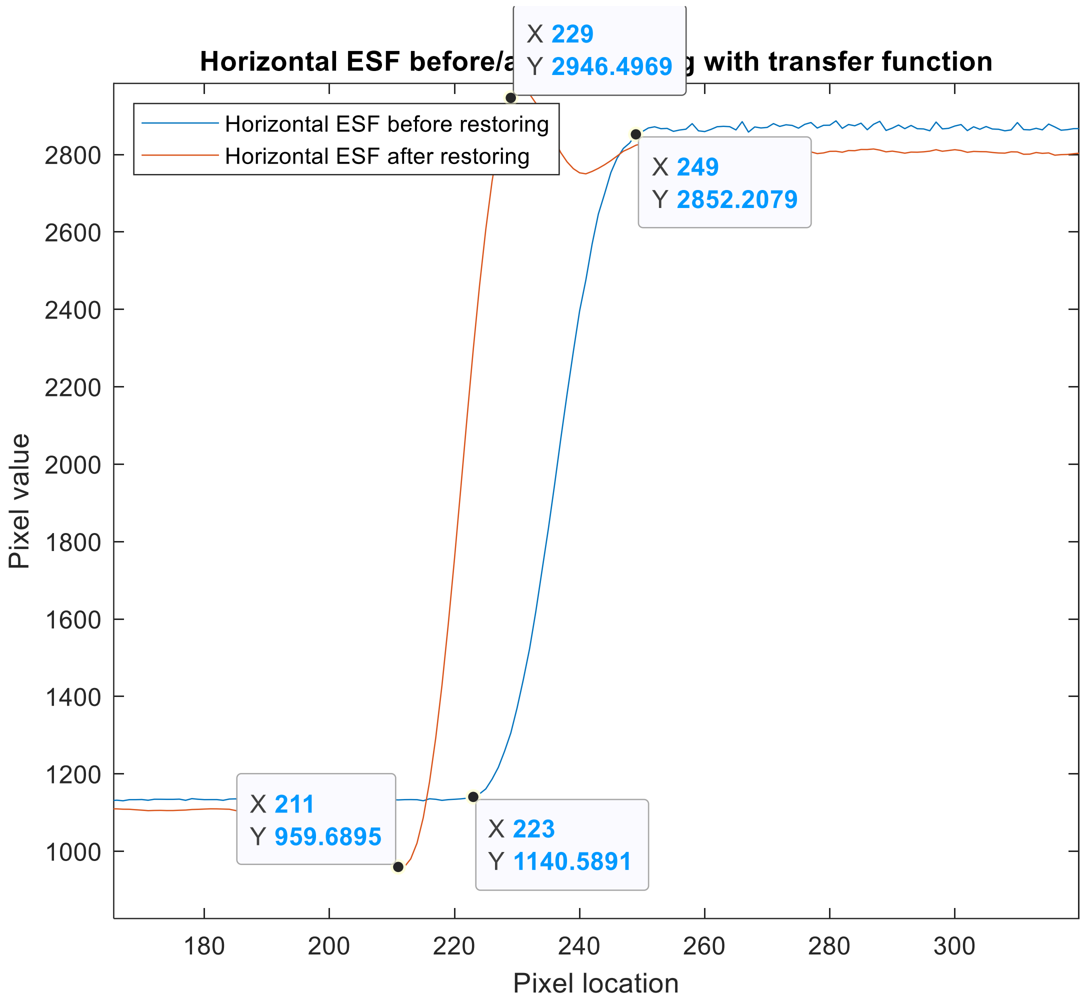

Figure 11: Horizontal profile, before and after filtering

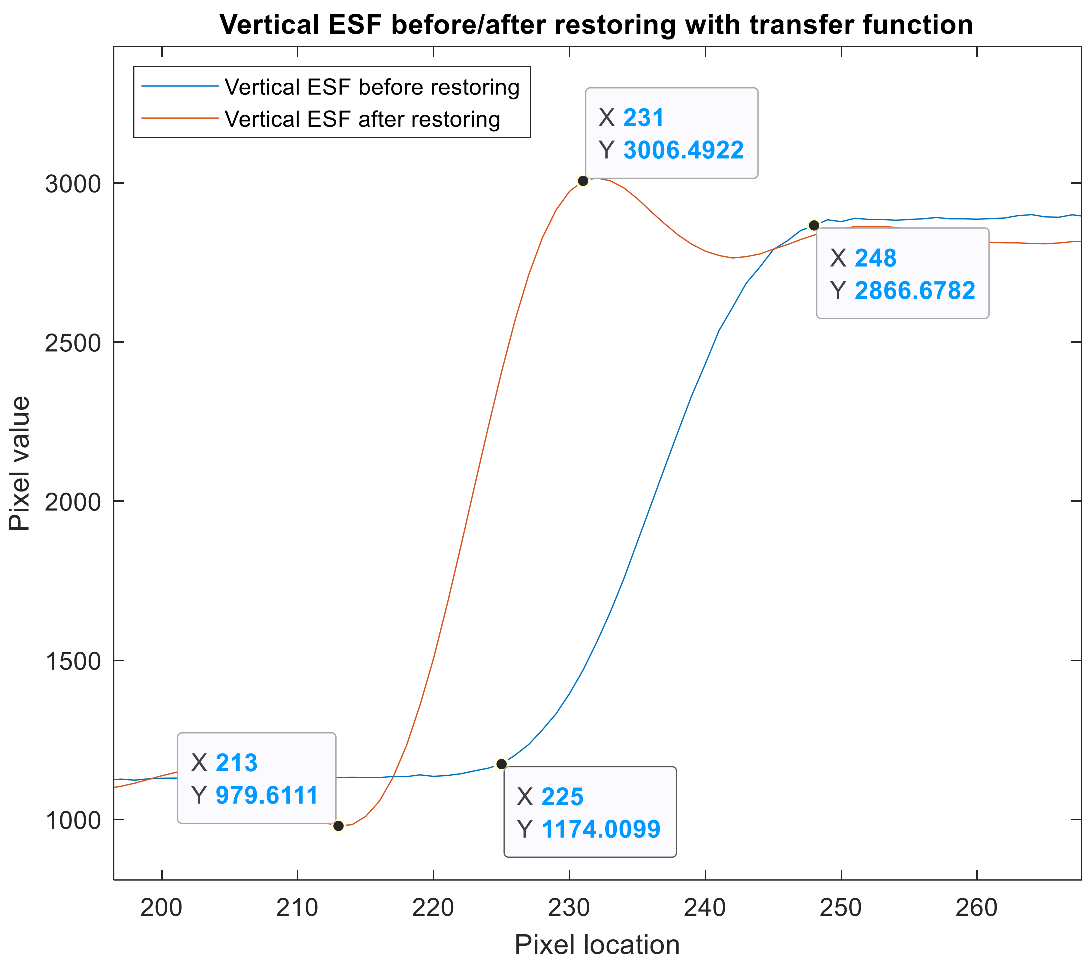

Figure 12: Vertical profile, before and after filtering

Table 2: Degree of restoration

	Slope before filtering	Slope after filtering
Horizontal Edge	65.83	110.38
Vertical Edge	73.59	112.60

Table 3: Noise comparison before and after filtering

Noise STD calculated using std2 MATLAB function	Standard Deviation
Blurred Chris	49.49
Filtered Chris	28.38

After looking at the before and after images, the filtered image looked sharper and brighter as compared to the original image. On comparing the ESFs, it was also noticed that the ESF of the filtered image had higher slope as compared to the original image. This was expected, since visually the filtered image looks sharper too. However, there was some ringing noticed in the filtered image. Which is also visible on the ESFs of the filtered image (overshoot and ripple).

Furthermore, on comparing the noise standard deviation of the filtered and original image, it was noticed that the filtered image has less noise as compared to the original image. Which was also expected.

6. Repeat the previous step using Matlab's 'deconvwnr' function.

Steps:

1. Fspecial was used to create the PSF in spatial domain by defining its spatial size and standard deviations.
2. A smooth region was selected on the blurred image and the NSR was calculated by dividing standard deviation of the noise with the mean value of the image.
3. Same procedure as step 5 was repeated.

   Result:

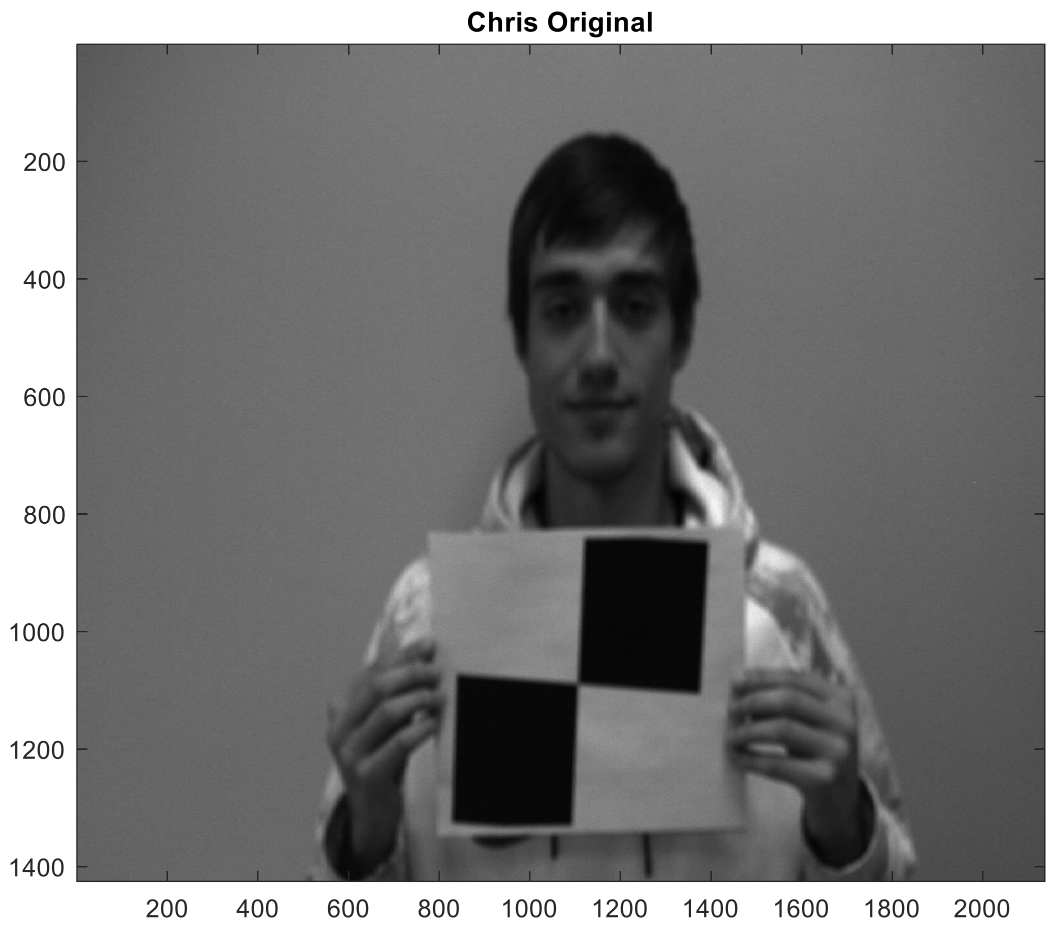

Figure 13: Chris Original

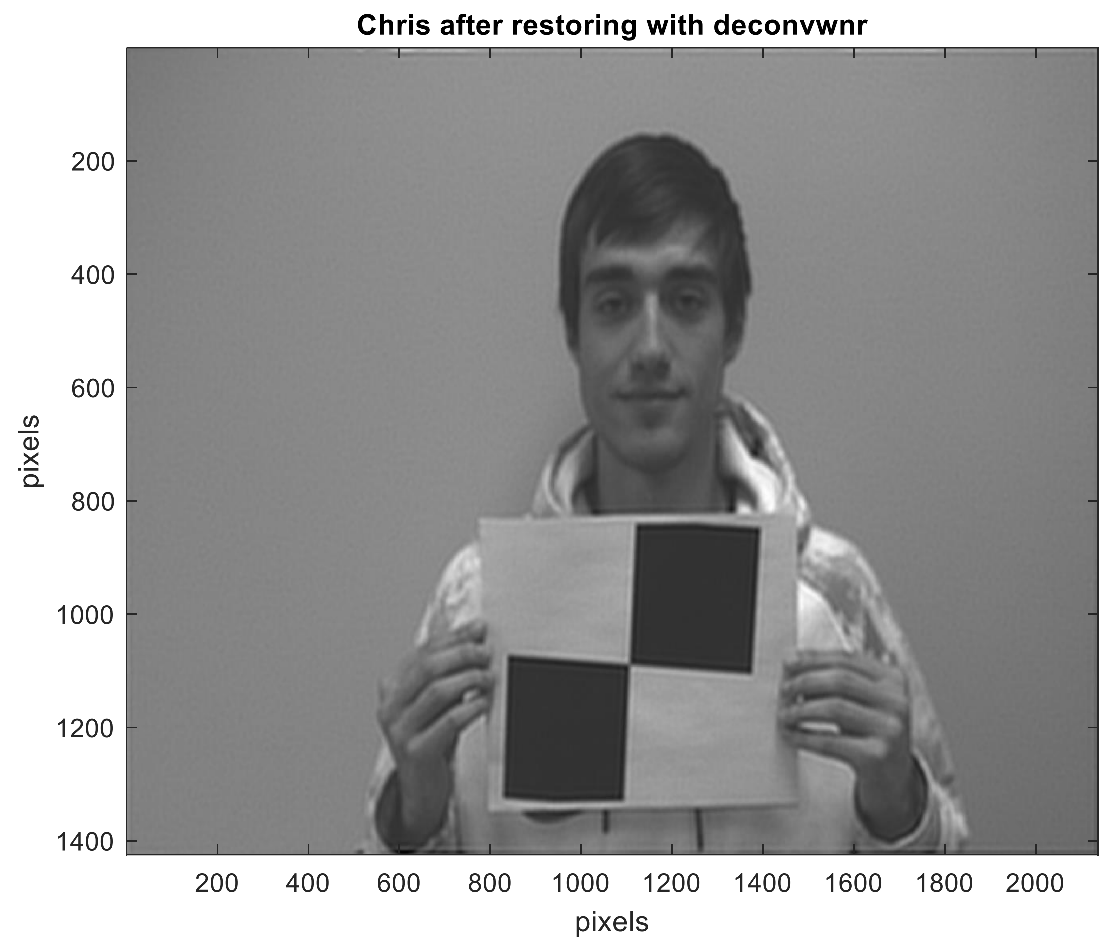

Figure 14: Chris Filtered with deconvwnr

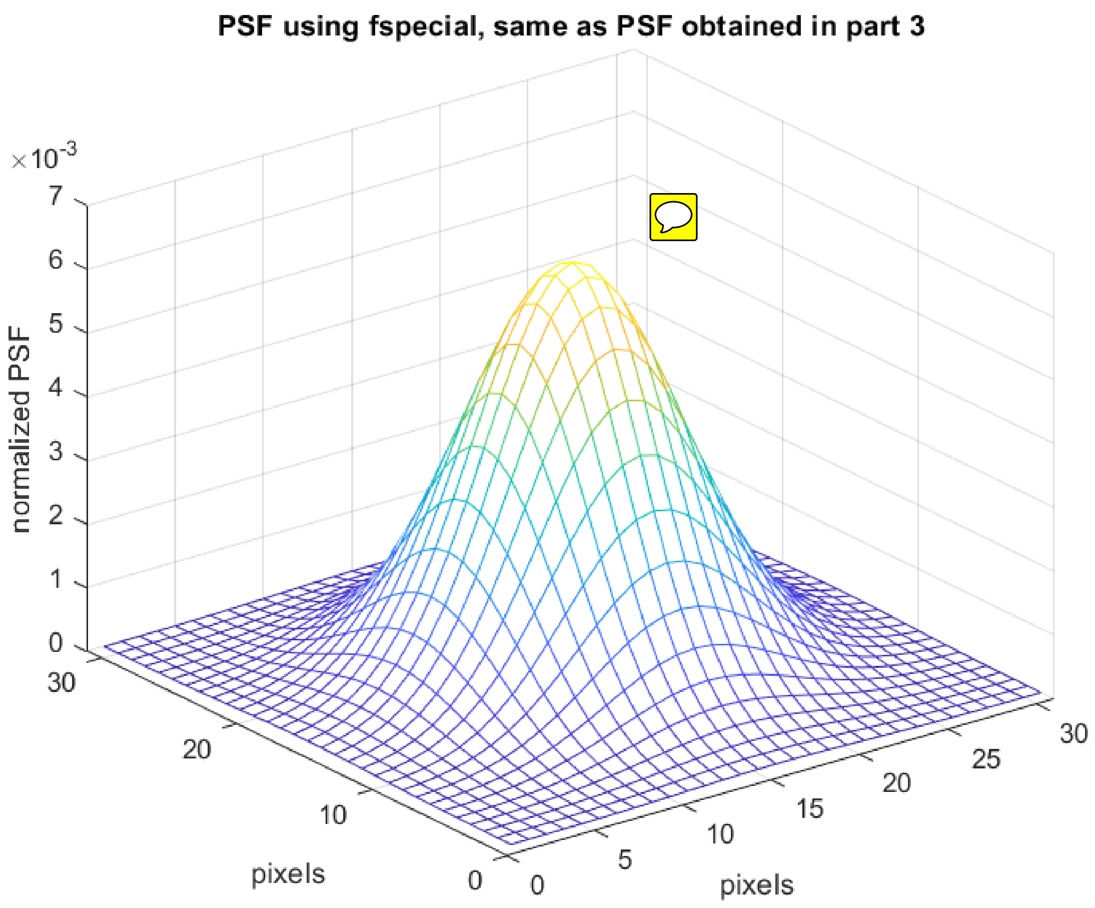

Figure 15: PSF created using fspecial, same as PSF from part 3

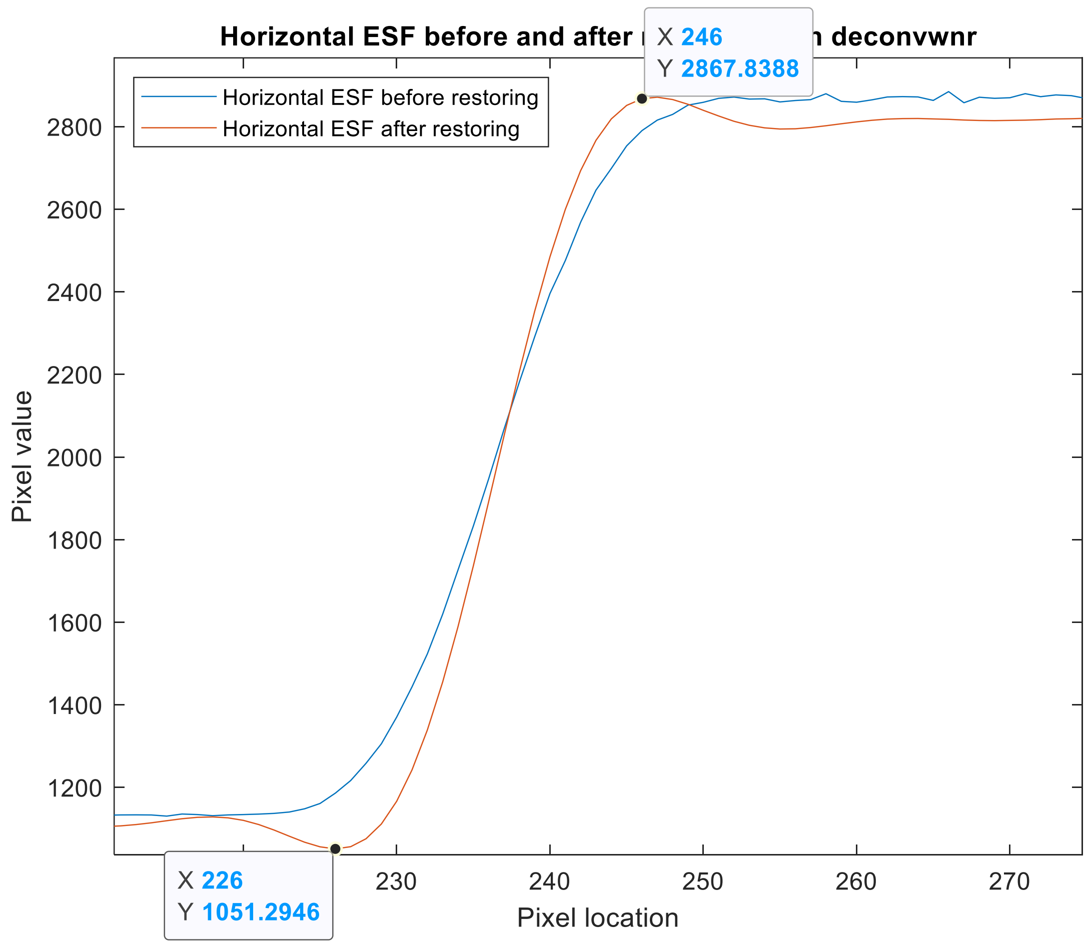

Figure 16: Horizontal profile, before and after filtering with deconwnr

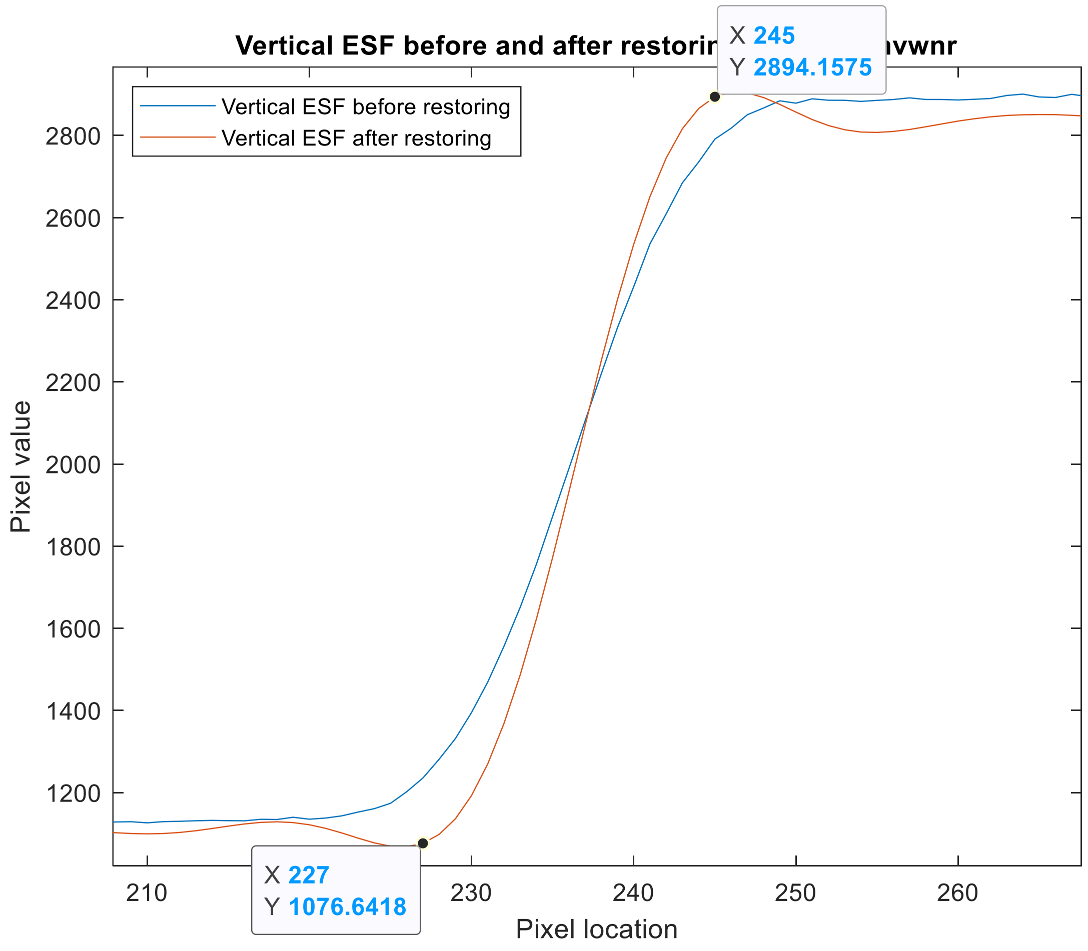

Figure 17: Vertical profile, before and after filtering with deconvwnr

Table 4: Degree of restoration for deconvwnr

	Slope before filtering	Slope after filtering
Horizontal Edge	65.83	90.82
Vertical Edge	73.59	100.97

Table 5: Noise comparison before and after filtering with deconvwnr

Noise STD calculated using std2 MATLAB function	Standard Deviation
Blurred Chris	49.49
Filtered Chris	25.39

After looking at the before and after images, the filtered image looked sharper and brighter as compared to the original image. On comparing the ESFs, it was also noticed that the ESF of the filtered image had higher slope as compared to the original image. This was expected, since visually the filtered image looks sharper too. However, on comparing the results in step 6 with the results from step 5, it was noticed that the filtered image from step 6 had less ringing as compared to the filtered image from step 5. This can also be seen in the ESFs of filtered image of step 6, since the step 6 ESFs of filtered image have less ripples and less overshoot as compared to the ESF of the filtered image from step 5.

Furthermore, on comparing the noise standard deviation of the filtered in both step 5 and 6, it was noticed that the standard deviation of noise in step 6 is lesser than standard deviation of noise in step 5.

7. Restore the image using either approach (Step 5 or Step 6), but modify your blur and noise estimates to improve the restored image. Justify your modified approach both qualitatively and quantitatively.

   Steps and analysis:

   The deconvwnr was used for part 7. The standard deviation of the PSF was changed. It was noticed that the increase in the standard deviation results in ringing and the decrease I standard deviation results in increase in noise. Hence, standard deviation was not changed, and the same value was used as part 6 and part 5.
   Later, the NSR was changed. It was noticed the with the decrease in NSR the image got worse. Hence, the NSR was increased and experimentally it was determined that NSR = 0.05, yielded the best-looking results. Even though the NSRs> 0.05 gave more noise reduction than NSR = 0.05 but they were not visually looking good. Later, noise standard deviation was computed for NSR = 0.05 and was compared to the noise standard deviation of the original image, table 6. The increase in NSR did decrease the standard deviation of the noise and therefore better-looking picture.

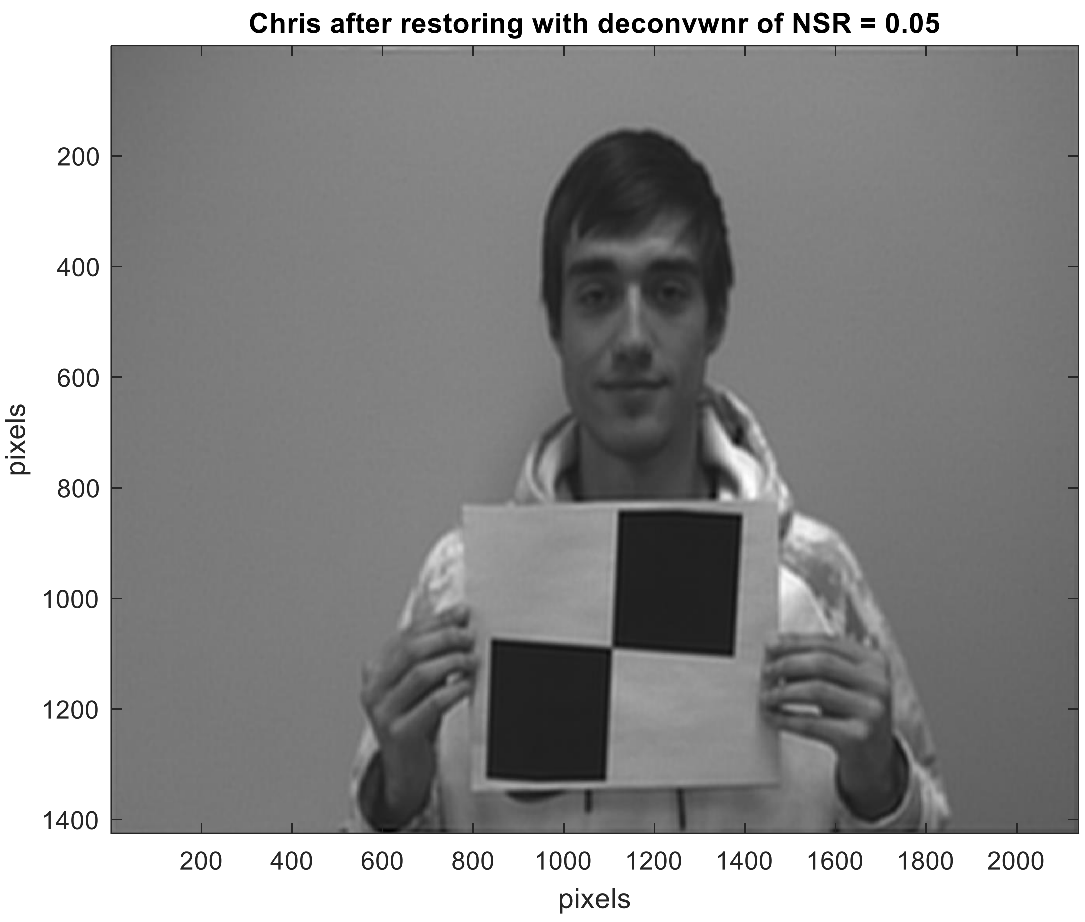

Figure 18: Chris at NSR = 0.05

Table 6: Noise comparison before and after filtering with deconvwnr with NSR = 0.05

Noise STD calculated using std2 MATLAB function	Standard Deviation
Blurred Chris	49.95
Filtered Chris	26.10

Table 7: Noise standard deviation comparison at different NSRs

NSRs	Standard Deviation
0.05	26.10
0.06	25.70
0.07	25.25
0.08	24.55
0.15	22.43

8. Append your Matlab code to the report you turn in.

Appendix

```
%Author: Harsh Dubey
%Worked with Lin Zeng and Gagan Singla
%Date: 11/10/19

clear all
close all

chrisOriginal=double(imread("ChrisBlur.tiff"));
figure;imagesc(chrisOriginal);colormap(gray(256));title('Ch
ris Original')

%% Part 1 and Part 2: Starts

x = [866 1302 1302 866 866];
y = [1106 1106 1308 1308 1106];
croppes=poly2mask(x,y,1424,2136);
H_cropped=chrisOriginal.*croppes;H_cropped(H_cropped==0)=na
n;
figure;imagesc(H_cropped);colormap(gray(256));title('Horizo
ntally cropped checkerboard');


x = [1148 1350 1350 1148 1148];
y = [863 863 1299 1299 863];
croppes=poly2mask(x,y,1424,2136);
V_cropped=chrisOriginal.*croppes;V_cropped(V_cropped==0)=na
n;
figure;imagesc(V_cropped);colormap(gray(256));title('Vertic
ally cropped checkerboard');

%ESF: HORIZONTAL
Horizontal_ESF = mean(H_cropped(:,866:1302),
1,'omitnan');figure;plot(Horizontal_ESF);hold on;
%FERMI FIT
x=1:437;%range
d=1127.8;%offset
a=1759.3;%amp
b=235;%edge location
c=-4;%curv
fermiFunc=(a./(exp((x-b)/c)+1))+d;%fermi function
plot(x,fermiFunc);hold off;
title('Horizontal ESF');
```

```
legend1 = sprintf('Horizontal ESF')
legend2 = sprintf('Horizontal ESF, fermi fit')
legend({legend1,legend2},'location','west')

%ESF: Vertical
Vertical_ESF = mean(V_cropped(863:1299,:), 2,'omitnan');
figure;plot(Vertical_ESF);hold on;
%FERMI FIT
x=1:437;%range
d=1127.8;%offset
a=1759.3;%amp
b=235;%edge location
c=-4;%curv
fermiFuncV=(a./(exp((x-b)/c)+1))+d;
fermiFuncV=fermiFuncV.';
plot(x,fermiFuncV);hold off;title('Vertical ESF');
legend1 = sprintf('Vertical ESF')
legend2 = sprintf('Vertical ESF, fermi fit')
legend({legend1,legend2},'location','west')

LSH_Horizontal=diff(Horizontal_ESF);
figure;plot(LSH_Horizontal);
LSH_Vertical=diff(Vertical_ESF);
figure;plot(LSH_Vertical);

%LSF
d=-15:15;
LSF_Horizontal=diff(fermiFunc);LSF_Horizontal=LSF_Horizonta
l(:,220:250);
figure;plot(d,LSF_Horizontal);title('Horizontal LSF');
xlabel('Pixels');ylabel('LSF');
LSF_Vertical=diff(fermiFuncV);LSF_Vertical=LSF_Vertical(220
:250,:);
figure;plot(d,LSF_Vertical);title('Vertical LSF');
xlabel('Pixels');ylabel('LSF');


%% PSF
PSF=abs(LSF_Horizontal).*abs(LSF_Vertical);
PSF=PSF/sum(PSF(:));
figure;mesh(PSF);
title('Point spread function')
figure;plot(PSF);

for f = 32:1424
```

```
    for g=32:2136
        PSF(f,g)=0;
    end
end

%NSR estimation from Dr. Helder's code
noise = std2(chrisOriginal(120:800,440:720));
signal= mean2(chrisOriginal(120:800,440:720));
NSR = noise/signal;
%Fourier transform of chrisOriginal
FFTofChrisOrig=((fftshift((fftn(chrisOriginal)))));
%Modulation transfer function
MTFofLSF=((fftshift(fftn(PSF))));
%FFT
BlurrCong=conj(MTFofLSF);
BlurrMag=abs(MTFofLSF).^2;
W = BlurrCong./ (BlurrMag+NSR);
%deconvolution
FFTofGoodBoyChris=W.*FFTofChrisOrig;
%after taeking inverse fft
GoodBoyChris=abs(ifftn((FFTofGoodBoyChris)));
figure;imagesc(GoodBoyChris);
colormap(gray(256));
title('Chris after filtering with math equation of wiener
filter');
xlabel('pixels');ylabel('pixels');
noise2 = std2(GoodBoyChris(120:800,440:720));
%% Before and after comparision with H(u,v)

x = [866 1302 1302 866 866];
y = [1106 1106 1308 1308 1106];
H_Crop=poly2mask(x,y,1424,2136);
H_before=chrisOriginal.*H_Crop;
H_before(H_before==0)=nan;
H_after=GoodBoyChris.*H_Crop;
H_after(H_after==0)=nan;

x = [1148 1350 1350 1148 1148];
y = [863 863 1299 1299 863];
V_crop=poly2mask(x,y,1424,2136);
V_before=chrisOriginal.*V_crop;
V_before(V_before==0)=nan;
V_after=GoodBoyChris.*V_crop;
V_after(V_after==0)=nan;
```

```
%ESF x direction
ESF_H_before = mean(H_before(:,866:1302), 1,'omitnan');
ESF_H_after = mean(H_after(:,866:1302), 1,'omitnan');
figure;plot(ESF_H_before);hold on;
plot(ESF_H_after);hold off;
title('Horizontal ESF before/after restoring with transfer
function');
xlabel('Pixel location');
ylabel('Pixel value');
legend1 = sprintf('Horizontal ESF before restoring');
legend2 = sprintf('Horizontal ESF after restoring');
legend({legend1, legend2},'location','northwest');

%ESF y direction
ESF_V_before = mean(V_before(863:1299,:), 2,'omitnan');
ESF_V_after = mean(V_after(863:1299,:), 2,'omitnan');
figure;plot(ESF_V_before);hold on;plot(ESF_V_after);hold
off;
title('Vertical ESF before/after restoring with transfer
function');
xlabel('Pixel location');
ylabel('Pixel value');
legend1 = sprintf('Vertical ESF before restoring');
legend2 = sprintf('Vertical ESF after restoring');
legend({legend1, legend2},'location','northwest');


%% MATLAB deconvwnr approach
%Estimating the noise
noise = std2(chrisOriginal(110:780,450:740));
signal= mean2(chrisOriginal(110:780,450:740));
NSR = noise/signal;
%Creating our own PSF using fspecial
PSF = fspecial('gaussian',31,5);
figure;mesh(PSF);
xlabel('pixels');
ylabel('pixels');
zlabel('normalized PSF')
title('PSF using fspecial, same as PSF obtained in part 3')
%Deconvolving
goodBChris = deconvwnr(chrisOriginal,PSF,NSR);
figure;imagesc(goodBChris);
colormap(gray(256));
```

```
title('Chris after restoring with deconvwnr');
xlabel('pixels');ylabel('pixels');
noise3 = std2(goodBChris(120:800,440:720));

%% Before and after for deconvwnr
%% Before and after comparision with H(u,v)
%Before and after horizontal and vertical ESF
%Horizontally crop the checkboard area from image
x = [866 1302 1302 866 866];
y = [1106 1106 1308 1308 1106];
H_Crop=poly2mask(x,y,1424,2136);
H_before=chrisOriginal.*H_Crop;
H_before(H_before==0)=nan;
H_after=goodBChris.*H_Crop;
H_after(H_after==0)=nan;

%Vertically crop the checkboard area from image
x = [1148 1350 1350 1148 1148];
y = [863 863 1299 1299 863];
V_crop=poly2mask(x,y,1424,2136);
V_before=chrisOriginal.*V_crop;
V_before(V_before==0)=nan;
V_after=goodBChris.*V_crop;
V_after(V_after==0)=nan;

%ESF x axis
ESF_H_before = mean(H_before(:,866:1302), 1,'omitnan');
ESF_H_after = mean(H_after(:,866:1302), 1,'omitnan');
figure;plot(ESF_H_before);hold on;
plot(ESF_H_after);hold off;
title('Horizontal ESF before and after restoring with
deconvwnr');
xlabel('Pixel location');
ylabel('Pixel value');
legend1 = sprintf('Horizontal ESF before restoring');
legend2 = sprintf('Horizontal ESF after restoring');
legend({legend1, legend2},'location','northwest');

%ESF y axis
ESF_V_before = mean(V_before(863:1299,:), 2,'omitnan');
ESF_V_after = mean(V_after(863:1299,:), 2,'omitnan');
figure;plot(ESF_V_before);hold on;plot(ESF_V_after);hold
off;
```

```
title('Vertical ESF before and after restoring with
deconvwnr');
xlabel('Pixel location');
ylabel('Pixel value');
legend1 = sprintf('Vertical ESF before restoring');
legend2 = sprintf('Vertical ESF after restoring');
legend({legend1, legend2},'location','northwest');

%% Part 7: Same as deconvwnr
%% MATLAB deconvwnr approach
%Estimating the noise
noise = std2(chrisOriginal(110:780,450:740));
signal= mean2(chrisOriginal(110:780,450:740));
NSR = noise/signal;
%Creating our own PSF using fspecial
stdk = 5;
PSF = fspecial('gaussian',31,stdk);
figure;mesh(PSF);
xlabel('pixels');
ylabel('pixels');
zlabel('normalized PSF')
title('PSF using fspecial, modified to better results')
%Deconvolving
goodBChris = deconvwnr(chrisOriginal,PSF,0.05);
figure;imagesc(goodBChris);
colormap(gray(256));
title('Chris after restoring with deconvwnr of NSR =
0.05');
xlabel('pixels');ylabel('pixels');
noise3 = std2(goodBChris(110:780,450:740));
%% Noise estimation

x = [571 820 822 570 571];
y = [330 325 480 469 330];
croppes=poly2mask(x,y,1424,2136);
noise_1=chrisOriginal.*croppes;noise_1(noise_1==0)=nan;
figure;histogram(noise_1);title('Noise Distribution of
region 1');
legend1 = sprintf('Standard Deviation = 45.18')
legend(legend1)
x = [1637 1974 1974 1637 1637];
y = [177 177 335 334 117];
croppes=poly2mask(x,y,1424,2136);
noise_2=chrisOriginal.*croppes;noise_2(noise_2==0)=nan;
```

```
figure;histogram(noise_2);title('Noise Distribution of
region 2');
legend1 = sprintf('Standard Deviation = 53.46')
legend(legend1)
x = [206 526 526 206 206];
y = [639 639 798 798 639];
croppes=poly2mask(x,y,1424,2136);
noise_3=chrisOriginal.*croppes;noise_3(noise_3==0)=nan;
figure;histogram(noise_3);title('Noise Distribution of
region 3');
legend1 = sprintf('Standard Deviation = 47.50')
legend(legend1)
%Standard deviation of noise
stdnoise(1,:)=std(noise_1,0,'all','omitnan');stdnoise(2,:)=
std(noise_2,0,'all','omitnan');stdnoise(3,:)=std(noise_3,0,
'all','omitnan');
```