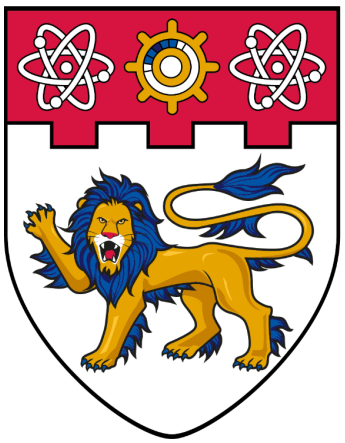


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4003 Computer Vision

Lab 1: Point Processing + Spatial Filtering + Frequency Filtering + Imaging Geometry

Sherwin Samson
U2020911J
06/10/2022

2.1 Contrast Stretching

(a) Display:

Input->

```
Pc = imread('mrt-train.jpg');
whos Pc;
```

Output->

Name	Size	Bytes	Class	Attributes
Pc	320x443x3	425280	uint8	

Input->

```
P = rgb2gray(Pc);
whos P;
```

Output->

Name	Size	Bytes	Class	Attributes
P	320x443	141760	uint8	

(b) View the grey scale image:

Input->

```
imshow(P)
```

Output->

(c) Finding the minimum and maximum values of intensities:

Input->

```
minP = double(min(P(:)));
maxP = double(max(P(:)));
```

Output->

```
Initial value of Pixels =
[13,204]
```

maxP	204	1×1	double
minP	13	1×1	double

(d) Contrast Stretching:

(i) Subtract every pixel with the lower bound pixel value to set the new minimum as zero.

(ii) Next to rescale these values from 0 - 255, we multiply the resulting pixel value with 255.

(iii) We then divide this by the current scale of (maxP - minP= 204 -13).

Input->

```
sub = imsubtract(P,double(minP));
P2 = immultiply(sub,255/191);
minP2 = min(P2(:));
maxP2 = max(P2(:));
```

Output->

```
minP2 = 0            New value of Pixels = [0,255]
maxP2 = 255
```

(e) Display the stretched image:

Input->

```
imshow(P2);
```

Output->

Image with Pixels = [0,255]

2.2 Histogram Equalization

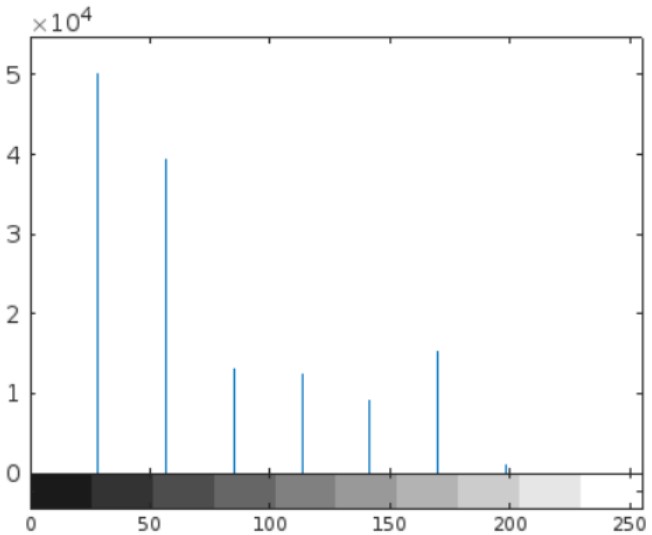
(a) Display the image intensity histogram of P:

Using 10 bins:

Input->

```
imhist(P,10);
```

Output->

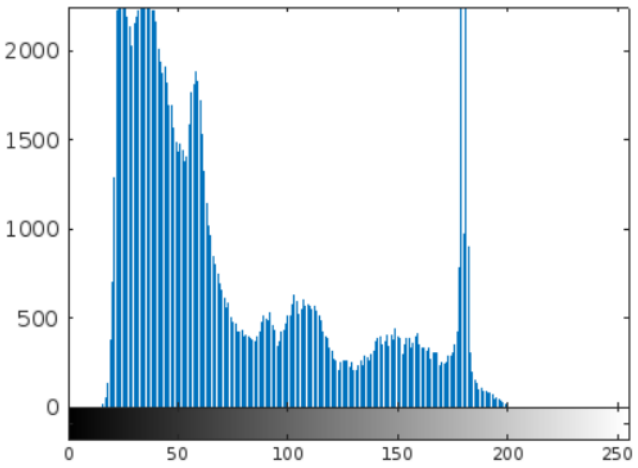


Using 256 bins

Input->

```
imhist(P,256);
```

Output->



The histogram with 256 bins is much more detailed than the histogram with 10 bins. This is because, when using 10 bins, 256 intensity levels are divided by 10 bins where each bin represents about 25 different intensities. When using 256 bins, each intensity level is held by each bin. Hence some details may not be seen when using lesser bins.

E.g.- The sharp increase at 180-190 can be seen when using 256 bins but not with 10 bins.

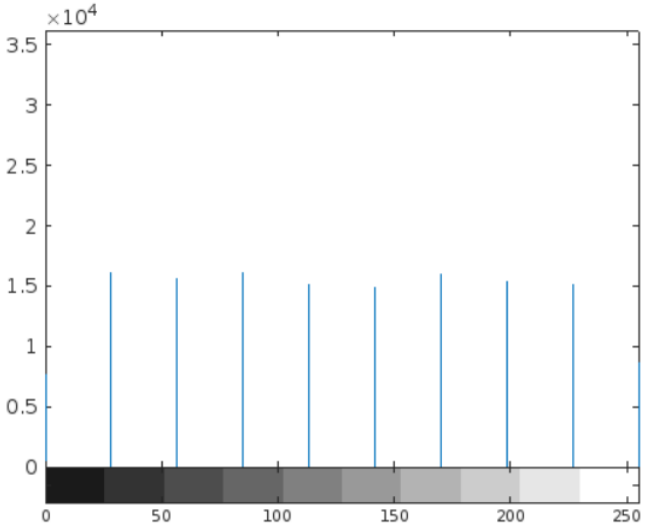
(b) Histogram equalization of P.

Display with 10 bins

Input->

```
P3 = histeq(P,255);
imhist(P3,10);
```

Output->

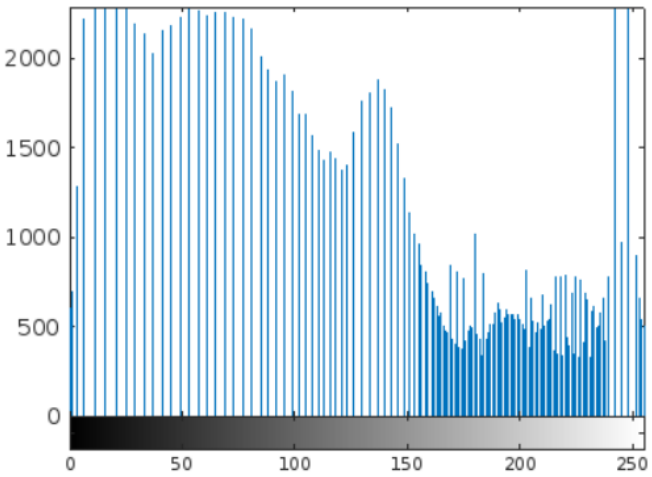


Display with 256 bins:

Input->

```
P3 = histeq(P,255);
imhist(P3,256);
```

Output->



After the Histogram equalization,

Similarities:

- The intensities are more evenly distributed over the different bins.
- Some bins which were previously empty received pixels from other bins.
- We can also observe that now there are many intensity levels that do not carry any pixels while some regions are highly packed.

Differences:

- The histogram with 10 bins has an almost similar number of pixels in each bin while the histogram with 256 bins has a high deviation in the number of pixels for each bin.
- The histogram with 256 bins has many empty bins while the histogram with 10 bins are not empty and mostly full.

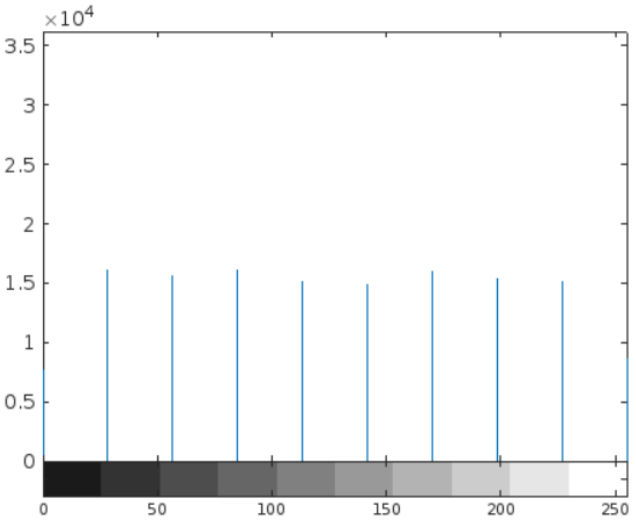
(c) Rerun histogram equalization on P3:

With 10 bins

Input->

```
P4 = histeq(P,255);
imhist(P4,10);
```

Output->

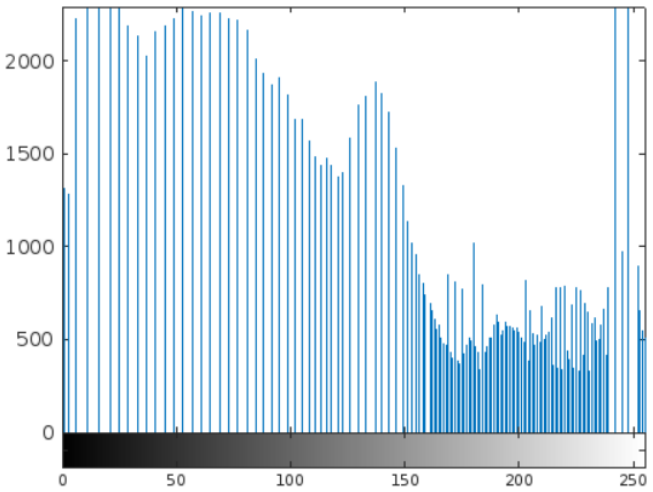


With 256 bins:

Input->

```
P4 = histeq(P,255);
imhist(P4,256);
```

Output->



- After performing histogram equalization again, there are no changes observed.
- As histogram equalization is an idempotent function and the pixels are already assigned to the bins based on the distribution function initially.
- On equalization, the bins are only merged and not split.Hence, the distribution remains unchanged.
- This shows that repeated equalizations will not necessarily reduce to a flat-histogram.

2.3 Linear Spatial Filtering:

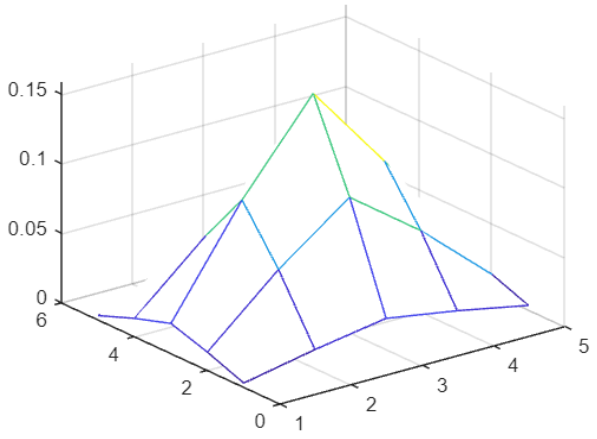
(a) Generating a filter for the point spread function:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Filter H1: Y and X-dimensions are 5 and σ = 1.0

Input->

```
sigma1 = 1.0;
size1 = -2:2;
[X1,Y1] = meshgrid(size1);
H1 = ( 1 / ( 2 * pi * sigma1.^2) ) *
exp( -((X1).^2 +
(Y1).^2)/(2*sigma1.^2));
mesh(H1);
```

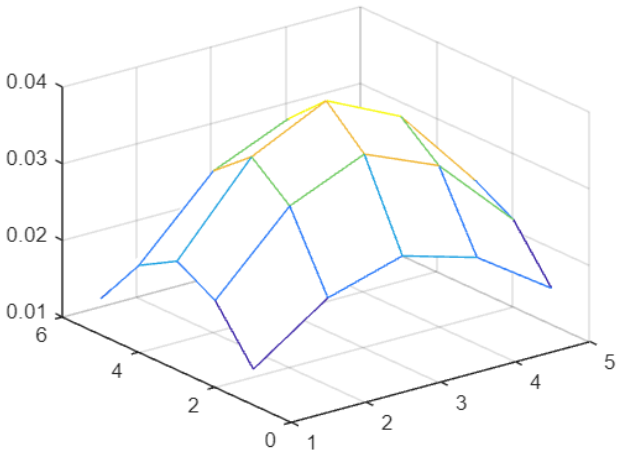


Output->

Filter H2: Y and X-dimensions are 5 and σ = 2.0

Input->

```
sigma2 = 2.0;
size2 = -2:2;
[X2,Y2] = meshgrid(size2);
H2 = ( 1 / ( 2 * pi * sigma2.^2) ) *
exp( -((X2).^2 +
(Y2).^2)/(2*sigma2.^2));
      mesh(H2);
```



Output->

2.3 (b) Exploration of Ntu-gn.jpg with Gaussian noise:

Input->

```
Pic1 = imread("ntu-gn.jpg");
figure
imshow(Pic1);
```

Output->

2.3 (c) Applying the Linear Filters H1 & H2 on ntu-gn.jpg:

Filter H1

Input->

```
Pic1_Filter1 = uint8(conv2(Pic1,H1));
imshow(Pic1_Filter1);
```

Output->

Filter H2:

Input->

```
Pic1_Filter2 = uint8(conv2(Pic1,H2));
imshow(Pic1_Filter2);
```

Output->

- The Filters H1 & H2 are not the most optimal against the image with gaussian noise.
- The 5x5 filter H2 removes more noise than the 3x3 filter H1.
- The 5x5 filter H2 however also causes more blurring than the 3x3 filter H1.
- The filters also enhance the frame of the image as there is a dark layer along the sides of the frame.
- H2 filter is more efficient than H1 Filter against gaussian noise.
- After applying filters H1 & H2, some of the noise is removed. However, this comes with the tradeoff of reducing the image's definitions and blurring its features.

2.3 (d) Exploration of Ntu-sp.jpg with Speckle noise:

Original Image

Input->

```
Pic2 = imread("ntu-sp.jpg");
imshow(Pic2);
```

Output->

2.3 (e) Applying the Linear Filters H1 & H2 on ntu-sp.jpg:

Filter H1:

Input->

```
Pic2_Filter1 = uint8(conv2(Pic2,H1))
imshow(Pic2_Filter1);
```

Output->

Filter H2:

Input->

```
Pic2_Filter2 = uint8(conv2(Pic2,H2));
imshow(Pic2_Filter2);
```

Output->

- The H1 & H2 gaussian filters handle speckle noise poorly.
- Speckle noise could still be seen in the images after the filter is applied.
- This is because the noisy pixels were spread out among the neighbouring pixels.
- The filters also enhance the frame of the image as there is a dark layer along the sides of the frame.
- The images are more blurred with diminished features.
- Comparing between the gaussian noise and speckle noise, the Gaussian filters H1 & H2 handle gaussian noise better.

2.4 Median Filtering:

2.4 (i) Exploration of 3x3 and 5x5 Median Filters on Gaussian noise:

Original Image:

Input->

```
Pic1 = imread("ntu-gn.jpg");
figure
imshow(Pic1);
```

Output->

3x3 Median Filter:

Input->

```
Pic1_3x3 = uint8(medfilt2(Pic1,[3,3]));
figure
imshow(Pic1_3x3);
```

Output->

5x5 Median Filter:

Input->

```
Pic1_5x5 = uint8(medfilt2(Pic1,[5,5]));
figure
imshow(Pic1_5x5);
```

Output->

- The 5x5 Median filter removes more gaussian noise as compared to the 3x3 Median filter.
- The 5x5 Median filter causes more smoothing with the loss of some image sharpness along the edges as compared to the 3x3 Median filter.
- As the pixel in the noisy region will have the median intensity, the resulting image will have lesser definitions.
- The Gaussian filter can still handle the gaussian noise better than the Median filter.

2.4 (ii) Exploration of 3x3 and 5x5 Median Filters on Speckle noise:

Original Image:

Input->

```
Pic2 = imread("ntu-sp.jpg");
figure
imshow(Pic2);
```

Output->

3x3 Median Filter:

Input->

```
Pic2_3x3 = uint8(medfilt2(Pic2,[3,3]));
figure
imshow(Pic2_3x3);
```

Output->

5x5 Median Filter:

Input->

```
Pic2_5x5 = uint8(medfilt2(Pic2,[5,5]));
figure
imshow(Pic2_5x5);
```

Output->

- Median filtering proves to handle speckle noise very well.
- The median filter can preserve the edges better, without blurring the image.
- The speckle noise is almost not visible after applying the median filters.
- Median filters handle speckle noise much better than the gaussian filters.

In Conclusion, Gaussian filters handle gaussian noise better while Median filters handle speckle noise better.

2.5 Suppressing Noise Interference Patterns:

2.5 (a) Display original pck-int.jpg:

Input->

```
Pic3 = imread("pck-int.jpg");
figure
imshow(Pic3);
```

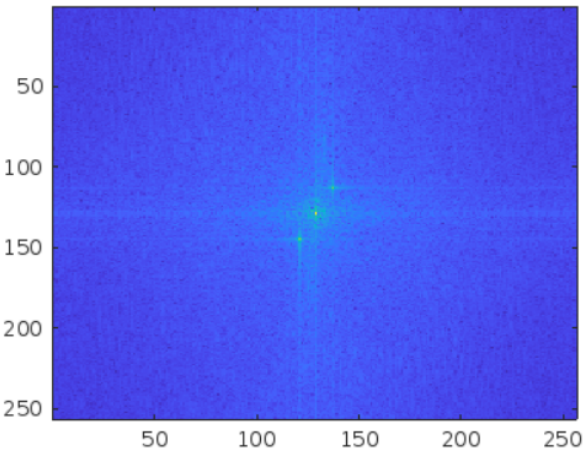
Output->

2.5 (b) Fourier Transform using fft2:

Input->

```
Pic3ft = fft2(Pic3);
Pic3S = abs(Pic3ft).^2/ length(Pic3);
figure
imagesc(fftshift(Pic3S.^0.1));
colormap('default')
```

Output->



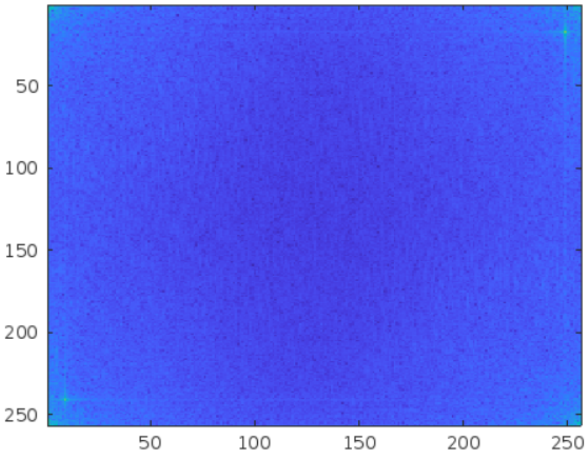
2.5 (c) Display power spectrum without fftshift:

Input->

```
imagesc(Pic3S.^0.1);
colormap('default');
% Location of peaks:
% a1,b1 = 249,17
% a2,b2 = 9,241
```

Output->

```
% 2 visible peaks observed.
%(After rounding the location
coordinates to the nearest integer
value)
```

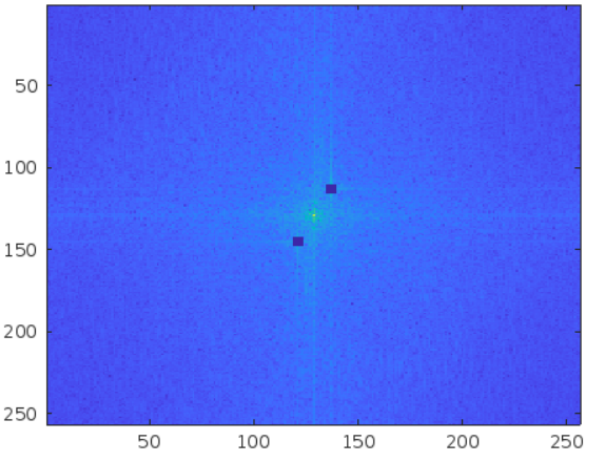


2.5 (d) Computing power spectrum after neighbourhood 5x5 elements = 0:

Input->

```
a1 = 249;b1 = 17;
a2 = 9; b2 = 241;
Pic3ft(b1-2 : b1+2, a1-2 : a1+2) = 0;
Pic3ft(b2-2 : b2+2, a2-2 : a2+2) = 0;
newS = abs(Pic3ft).^2 / length(Pic3);
figure
imagesc(fftshift(newS.^0.1));
colormap('default');
```

Output->

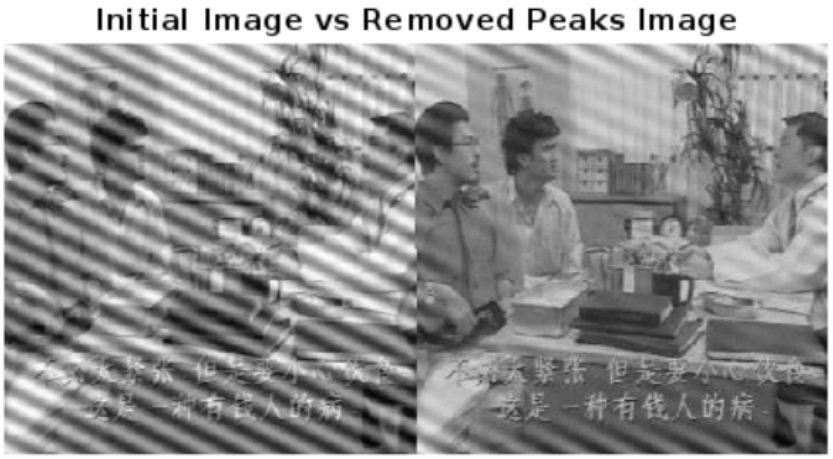


2.5 (e) Inverse Fourier Transform using ifft2

Input->

```
Pic3Inv =
uint8(ifft2(Pic3ft));
figure
imshow(Pic3Inv)
```

Output->



- The final image after applying inverse fourier transform has much less noise than the original image.
- The middle of the image has better visibility with lesser noise as compared to the corners.
- This is because we had only removed the 2 visible peaks in the fourier spectrum (c).
- The contrast of the image had also been reduced and appears blurred.

> Improvement of current filter:

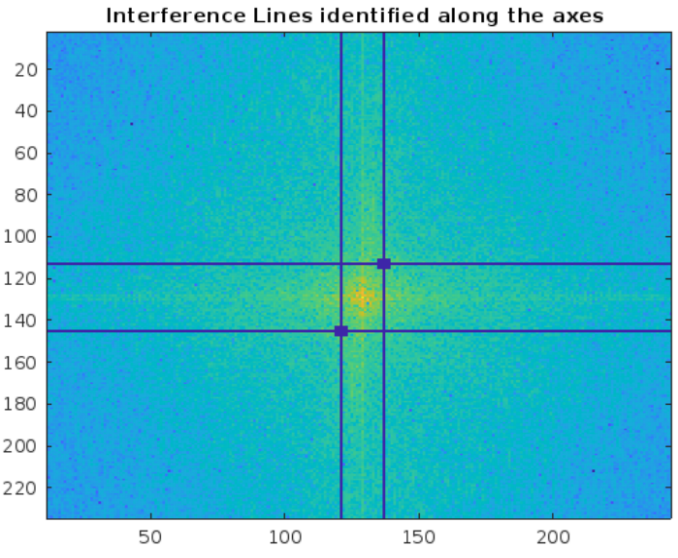
To further Improve this noise filter, I built upon the 5x5 filter by:

(i) Targeting the horizontal and vertical lines along the axes of the interference points.

- For this, I start with the already suppressed peaks in the fourier spectrum.
- Then, I select the respective x and y components from the coordinates and set them to zero in the Fourier spectrum.
- This will block out both the Horizontal and vertical lines passing through a given interference point.

Input->

```
a1 = 249;b1 = 17;
a2 = 9; b2 = 241;
Pic3ft(b1, :) = 0;
Pic3ft(b2, :) = 0;
Pic3ft(:, a1) = 0;
Pic3ft(:, a2) = 0;
S = abs(Pic3ft).^2 / length(Pic3);
figure
imagesc(fftshift(log10(S)))
newPckPeak = uint8(ifft2(Pic3ft));
figure
imshowpair(Pic3Inv,real(newPckPeak),
'montage')
```



```
title('Initial Removed Peak Image VS Removed Peak & Lines')
```

Output->

After removing the lines passing through the interference points, we can see a good improvement in the noise filtering. Along with filtering the middle portion of the image, the top half, bottom half and even the corners have better clarity and reduced noise.

(ii) To further improve, I have done Contrast stretching to enhance the image since the range of the pixels were reduced because of the fourier transformation.

Input->

```
pckMinP = double(min(newPck(:)));
pckMaxP = double(max(newPck(:)));
newPckContrast = uint8(255*(double(newPck) - pckMinP) / (pckMaxP -
pckMinP));
figure
imshowpair(Pic3Inv,real(newPckContrast), 'montage')
title('Initial Removed Peak Image VS After Improvements')
```

Output->

The final improvement after contrast stretching is also significant in seeing the depth of the image and even the subtitles in the image are more legible. Overall a much better improvement from just removing the Peaks.

2.5 (f) “Free” primate task

Original Image

Input->

```
Pic4 = imread('primatecaged.jpg');
Pic4 = rgb2gray(Pic4);
figure
imshow(Pic4);
```

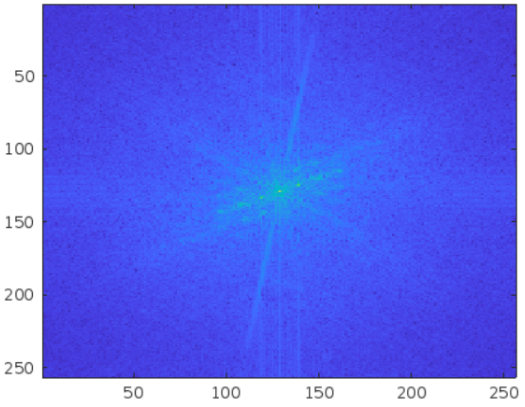
Output->

Fourier Transformed spectrum

Input->

```
Pic4ft = fft2(Pic4);
Pic4S = abs(Pic4ft).^2/
length(Pic4);
figure
imagesc(fftshift(Pic4S.^0.1)); %10th
root
colormap('default')
```

Output->

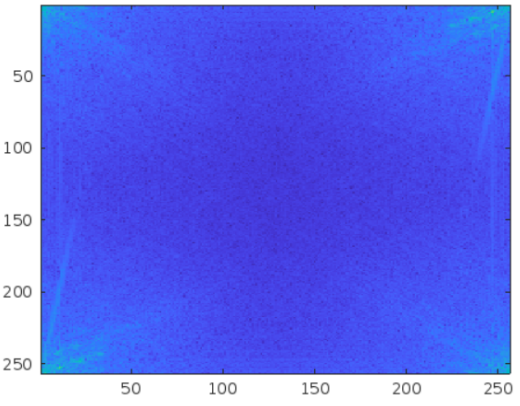


Find location of peaks

Input->

```
imagesc(Pic4S.^0.1);
colormap('default');
% p1,q1 = 253,11; p2,q2 = 6,247;
% p3,q3 = 248,21; p4,q4 = 10,236;
```

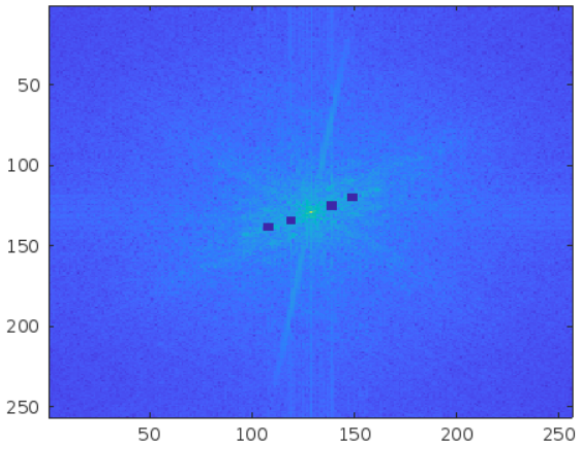
Output->



Set neighbourhood 5x5 elements = 0

Input->

```
p1 = 253; q1 = 11; p2 = 6; q2 = 247;
p3 = 248; q3 = 21; p4 = 10; q4 = 236;
Pic4ft(p1-2 : p1+2, q1-2 : q1+2) = 0;
Pic4ft(p2-2 : p2+2, q2-2 : q2+2) = 0;
Pic4ft(p3-2 : p3+2, q3-2 : q3+2) = 0;
Pic4ft(p4-2 : p4+2, q4-2 : q4+2) = 0;
newPic4S = abs(Pic4ft).^2 /
length(Pic4);
figure
imagesc(fftshift(newPic4S.^0.1));
colormap('default');
```



Output->

Inverse Fourier Transform

Input->

```
Pic4Inv = uint8(ifft2(Pic4ft));
figure
imshow(Pic4Inv)
```

Output->

Analysis of current filter:

- From the fourier transform of the primate's image, after rounding the coordinates to the nearest integer, I have identified 4 peaks which represent the fence: (253,11), (6,247), (248,21) and (10,236).
- After repeating the same steps in 2.5 (e) to remove the interferences as peaks in the spectrum, the fence is still not filtered very well.
- Removing the 4 interference peaks had also caused the image to be more blurred.

> Improvement of current filter:

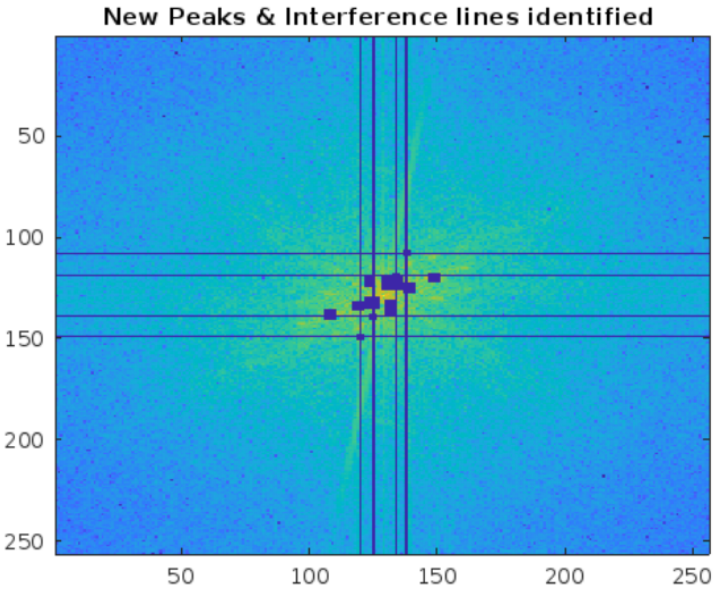
I have adopted a similar approach from 2.5 (e) to attempt and improve this filter.

(i) Identifying more Interference peaks and targeting the horizontal and vertical lines along the axes of the interference points.

- Starting with the 4 peaks which were already suppressed in the fourier spectrum.
- I have also identified new peaks and interference regions to help reduce the noise.
- Then, I select the respective x and y components from the coordinates and set them to zero in the Fourier spectrum.

- This will block out both the Horizontal and vertical lines passing through a given interference point.

Input->




```
p1 = 253; q1 = 11; p2 = 6; q2
= 247;
p3 = 248; q3 = 21; p4 = 10; q4
= 236;
Pic4ft(q1, :) = 0;
Pic4ft(q2, :) = 0;
Pic4ft(q3, :) = 0;
Pic4ft(q4, :) = 0;
Pic4ft(:, p1) = 0;
Pic4ft(:, p2) = 0;
Pic4ft(:, p3) = 0;
Pic4ft(:, p4) = 0;
Pic4ft(248 : 254, 1 : 8.5) =
0;
Pic4ft(3 : 10, 2: 6) = 0;
Pic4ft(248:252 , 250: 252) =
0;
Pic4ft(2:7.5 , 249.5: 255) = 0;
S = abs(Pic4ft).^2 / length(Pic4);
imagesc(real((log10(S))))
imagesc(real(fftshift(log10(S))))
newPrimatePeak = uint8(ifft2(Pic4ft));
figure
imshowpair(real(Pic4Inv),real(newPrimatePeak), 'montage')
title('Initial Removed Peak Image VS Removed Peak & Lines')
```

Output->

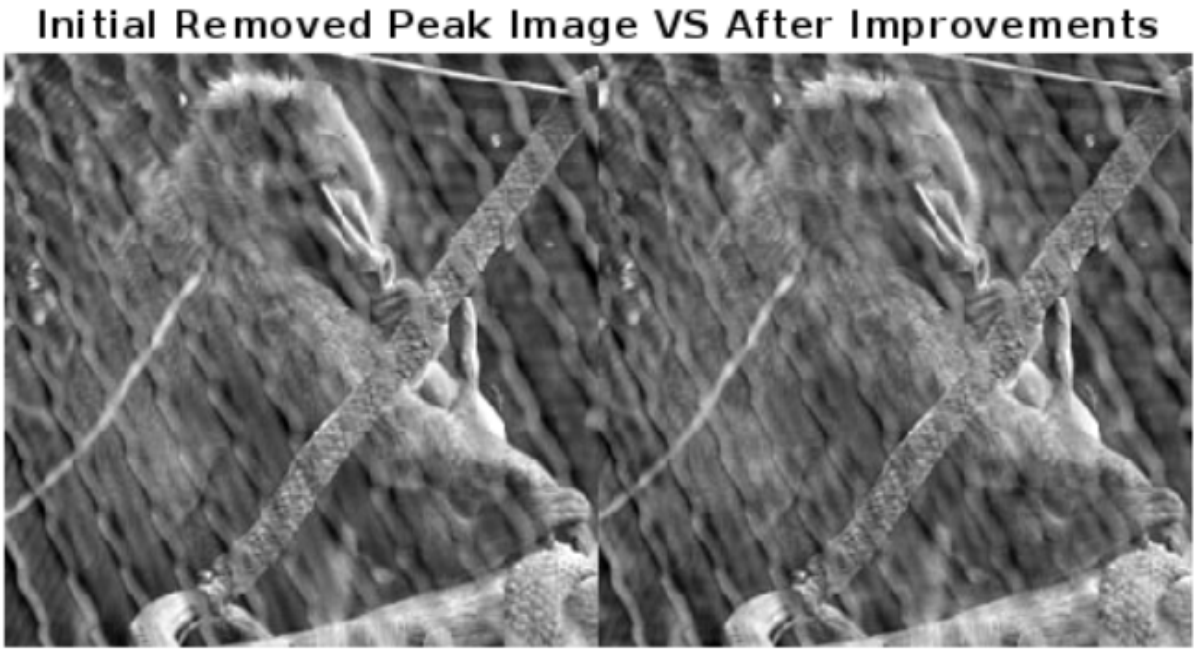
After removing the lines passing through the interference points, not much improvement can be seen in filtering the noise. The image has also become more blurred with diminished features.

(ii) To further improve, I have done Contrast stretching to enhance the image since the range of the pixels were reduced because of the fourier transformation.

Input->

```
primateMinP = min(double(newPrimate(:)));
primateMaxP = max(double(newPrimate(:)));
newPrimateContrast = uint8(255*(double(newPrimate) - pckMinP) /
(pckMaxP - pckMinP));
figure
imshowpair(real(Pic4Inv),real(newPrimateContrast), 'montage')
title('Initial Removed Peak Image VS After Improvements')
```

Output->



After contrast stretching, there still is not any obvious change or improvement in the noise filtering. Only a very minor change is observed at the middle portion of the image but still not very significant.

This could be due to the pattern of frequencies in this Primate's image. This interference's frequency can be inconsistent and may not be suppressed in using the same approach as the filtering for the Pck's image.

Thus, the frequency of the cage here can have a higher amplitude with respect to the Primate and other objects in the image. Due to this it is harder to separate it from the Primate itself

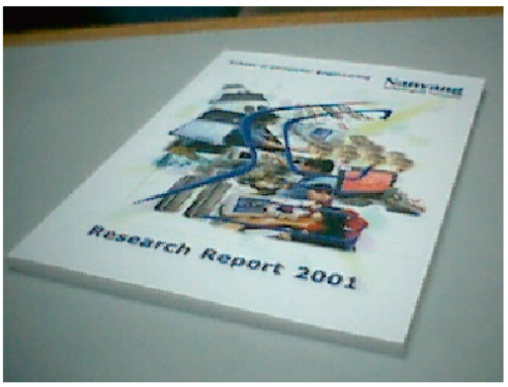
2.6 Undoing Perspective Distortion of Planar Surface

2.6 (a) Exploration of book.jpg:

Original image

Input->

```
Book_Pic = imread('book.jpg');
imshow(Book_Pic);
```



Output->

2.6 (b) Obtaining coordinates of the 4 corners of the book

Target coordinates of the book

Input->

```
[X,Y] = ginput(4);
xTarget = [0 210 210 0];
yTarget = [0 0 297 297];
```

(Target Coordinates to be selected as the 4 corners of the book.)

2.6 (c) Matrices to estimate projective transformation

Setting up matrices for u = A \ v

Input->

```
A = [ X(1) Y(1) 1 0 0 0 -xTarget(1)*X(1) -xTarget(1)*Y(1);
    0 0 0 X(1) Y(1) 1 -yTarget(1)*X(1) -yTarget(1)*Y(1);
    X(2) Y(2) 1 0 0 0 -xTarget(2)*X(2) -xTarget(2)*Y(2);
    0 0 0 X(2) Y(2) 1 -yTarget(2)*X(2) -yTarget(2)*Y(2);
    X(3) Y(3) 1 0 0 0 -xTarget(3)*X(3) -xTarget(3)*Y(3);
    0 0 0 X(3) Y(3) 1 -yTarget(3)*X(3) -yTarget(3)*Y(3);
    X(4) Y(4) 1 0 0 0 -xTarget(4)*X(4) -xTarget(4)*Y(4);
    0 0 0 X(4) Y(4) 1 -yTarget(4)*X(4) -yTarget(4)*Y(4);];
v = [xTarget(1);
   yTarget(1);
   xTarget(2);
   yTarget(2);
   xTarget(3);
   yTarget(3);
   xTarget(4);
   yTarget(4);];
u = A \ v;
U = reshape([u;1],3,3)';
```

```
w = U*[X'; Y'; ones(1,4)];
w = w ./ (ones(3,1) * w(3,:))
```

2.6 (d) Warping the Image:

Target coordinates of the book

Input->

```
T = maketform('projective', U');
P2 = imtransform(Book_Pic, T, 'XData', [0 210], 'YData', [0
297]);
```

2.6 (e) Displaying the Image:

Input->

```
imshow(P2)
```

Output->

- The final image of the book has been transformed into a 2 dimensional image.
- The features of the transformed image are also more defined
- The lower half of the result image P2 has sharper pixels while the top part seems to get blurry.
- The lower half of the book is also more focused while the top part is less focused.
- Hence to fit in our required frame, the bottom part was compressed to the target size while the top part was extended.

2.6 (f) Identifying the rectangular pink area:

Adopting a similar approach to above
(i) Setting the 4 coordinates of the rectangular pink area
(ii) Setting up the matrices U, A and V to estimate the projective transformation
(iii) Warping the image
We can extract the rectangular pink area.

Output->