Report

2.1 Contrast Stretching

c. Check the minimum and maximum intensities present in the image:

`min(P(:)), max(P(:))`

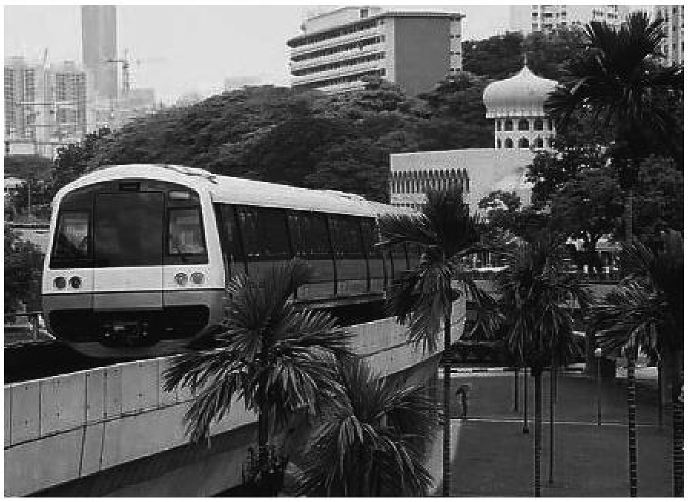
The minimun value is 13 and maximum value is 204.

d. Next, write two lines of MATLAB code to do contrast stretching. Hint: This involves a subtraction operation followed by multiplication operation (note that for images which contain uint8 elements, you will need to use imadd, imsubtract, etc. for the arithmetic instead of the usual operators; alternatively you can convert to a double-valued matrix first using double, but you will need to convert back via uint8 prior to using imshow — see below).

```
P_streched = 255/191*imsubtract(P,13);
imshow(P_streched)
```

Actually the command below works as well.

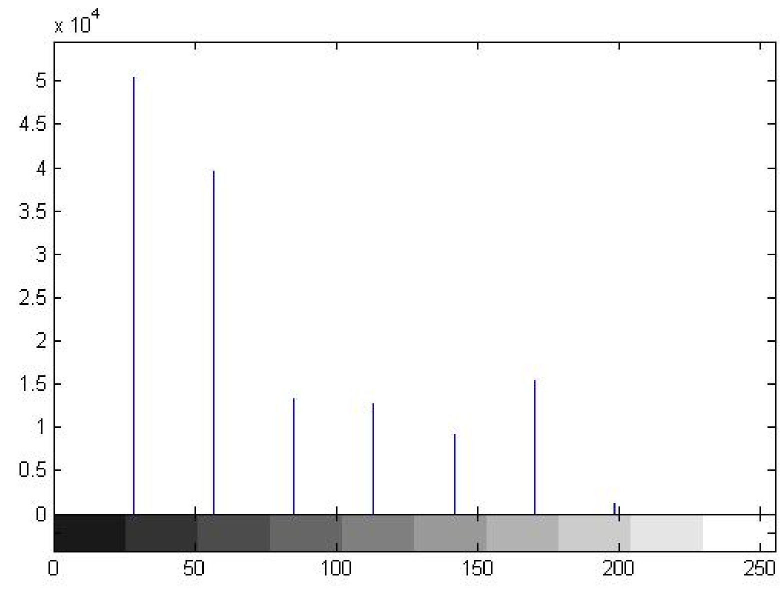
```
P_streched = 255/191*(P-13);
```

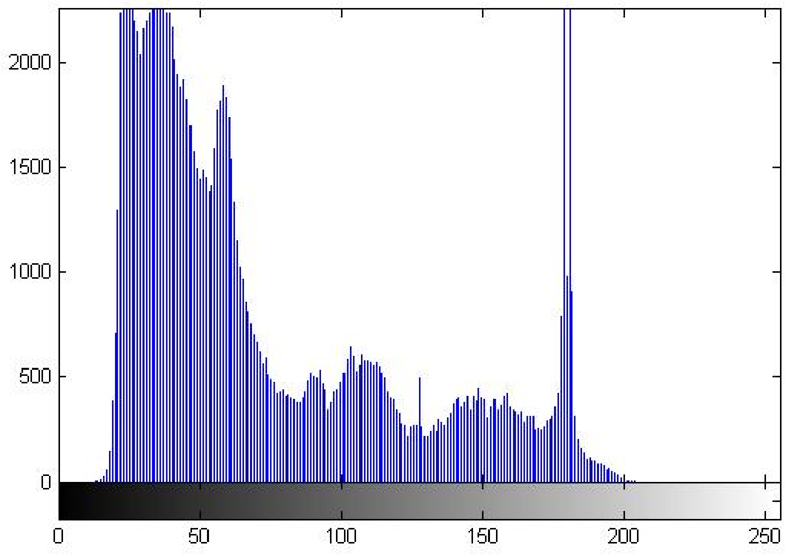


The minimum value is 0 and maximum is 255.

2.2 Histogram Equalization

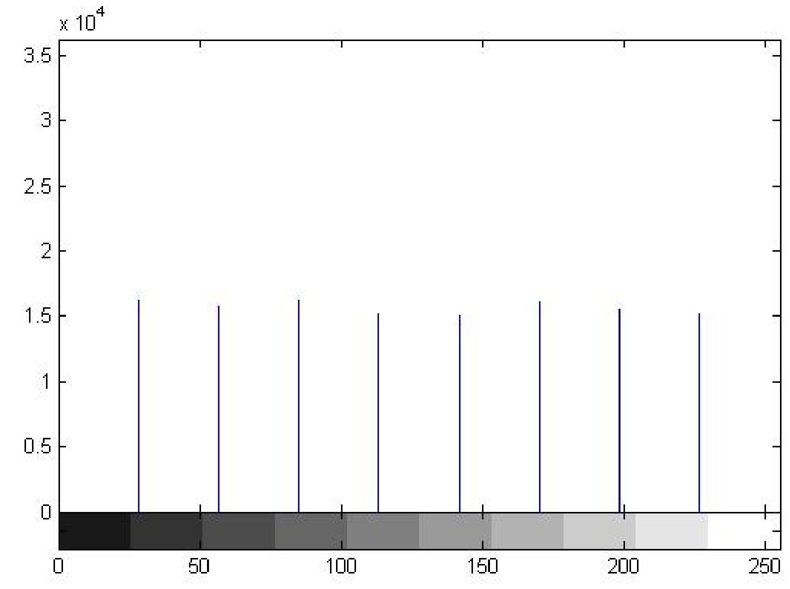
a. Display the image intensity histogram of P using 10 bins by imhist(P,10); Next display a histogram with 256 bins. What are the differences?

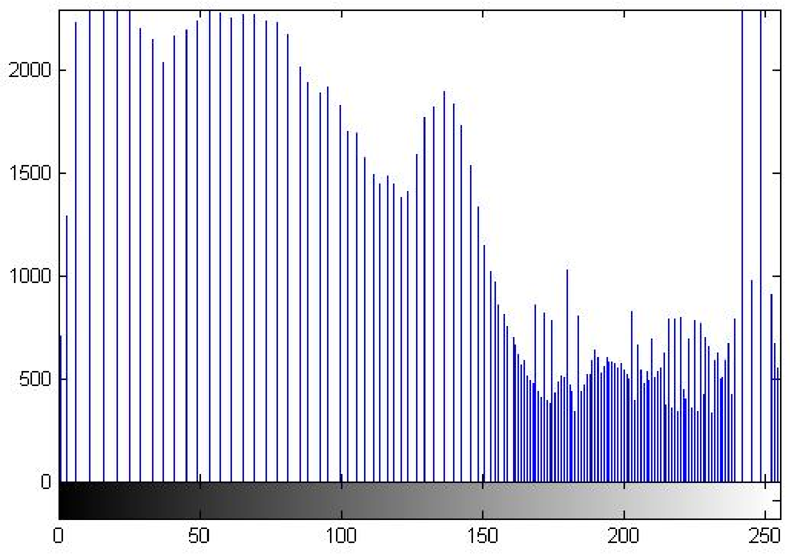




The average number of pixels in each bin becomes lesser, as we display the histogram with more bins. And it provides more information about the image.

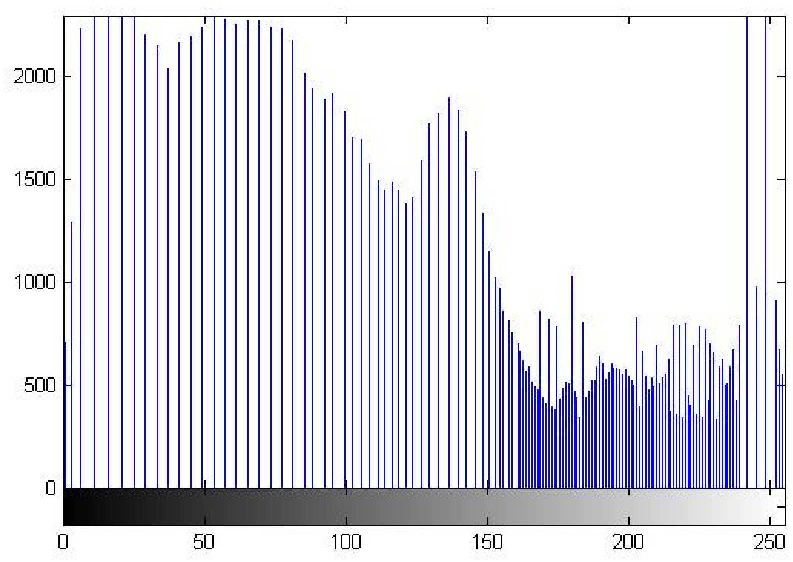
b. Next, carry out histogram equalization as follows:P3 = histeq(P,255); Redisplay the histograms for P3 with 10 and 256 bins. Are the histograms equalized? What are the similarities and differences between the latter two histograms?





Both the two histograms are more equalized than the original one. The histograms with 10 bins is more equalized. When we apply histogram equalization, the value for each gray-level may not be the same since it is discrete. However, the number of pixels fall in each range of gray-levels should be about the same when we divide the gray-levels into 10 bins.

c. Rerun the histogram equalization on P3. Does the histogram become more uniform? Give suggestions as to why this occurs.



No, P3 remained the same. In the first equalization process, the pixels are already maped to new gray-level based on the average number of pixel for each bin and the total number of pixels less than the original gray level. After this process the number of pixels less than the new gray-level will be the same as original. So P3 remains the same after we apply histogram equalization again.

2.3 Linear Spatial Filtering

a. Generate the Gaussian smoothing filters:

(i) Y and X-dimensions are 5 and σ = 1.0

```
h1 = fspecial('gaussian',[5 5],1);
```

(ii) Y and X-dimensions are 5 and σ = 2.0

```
h2 = fspecial('gaussian',[5 5],2);
```

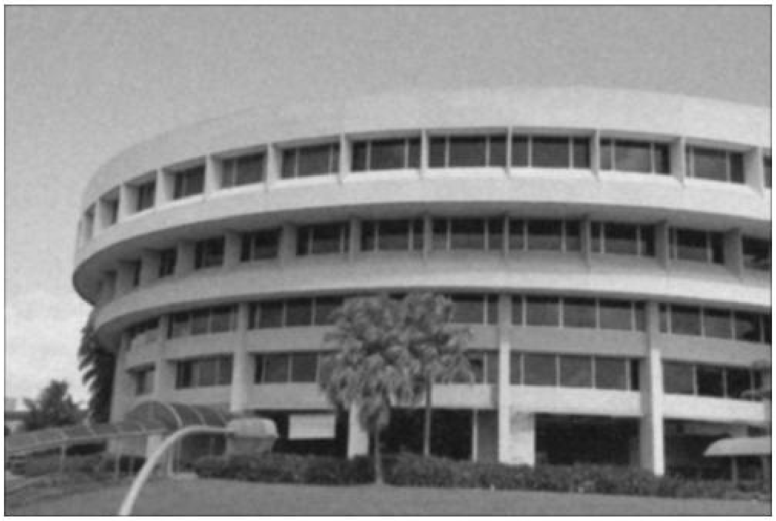
c. Filter the image using the linear filters that you have created above using the conv2 function, and display the resultant images. How effective are the filters in removing noise? What are the trade-offs between using either of the two filters, or not filtering the image at all?

```
P_double = im2double(P);
P1_double = conv2(P_double,h1);
imshow(P1_double)
P2_double = conv2(P_double,h2);
imshow(P2_double)
```

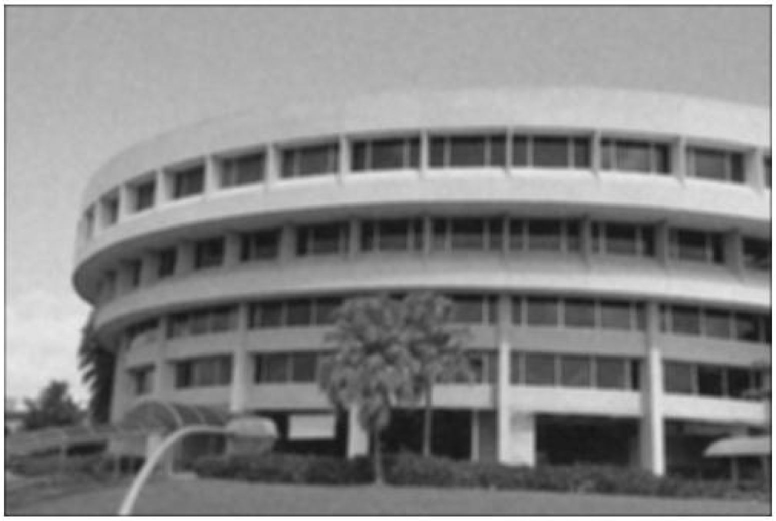
or simply

```
P1 = imfilter(P,h1);
P2 = imfilter(P,h2);
```

Filtered by h1:



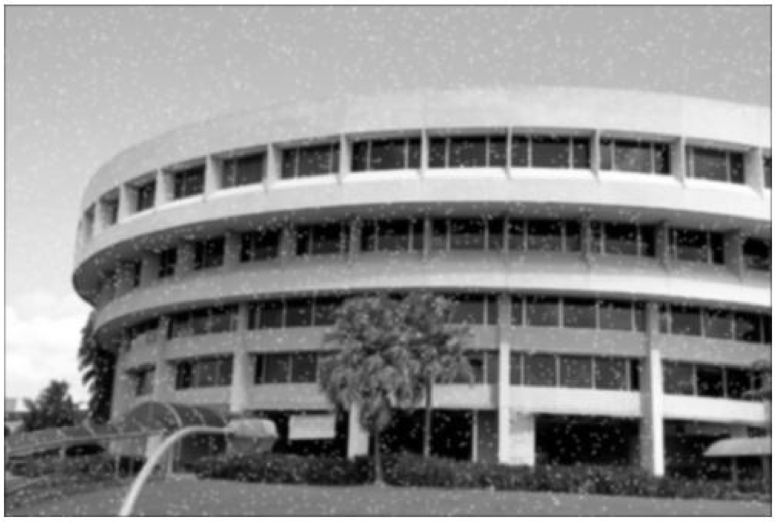
Filtered by h2:

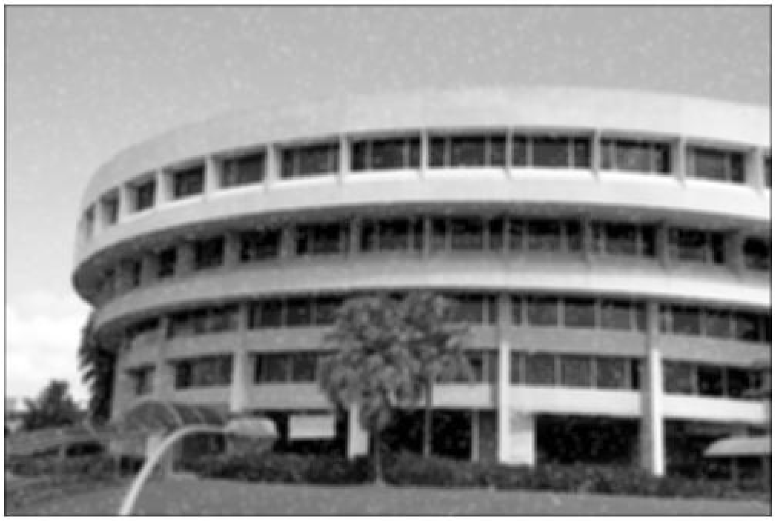


h2 is more effective inremoving the noise. But it makes the image more blurred.

e. Repeat step (c) above. Are the filters better at handling Gaussian noise or speckle noise?

```
P = imread('resource/ntusp.jpg');
P1 = imfilter(P,h1);
P2 = imfilter(P,h2);
```



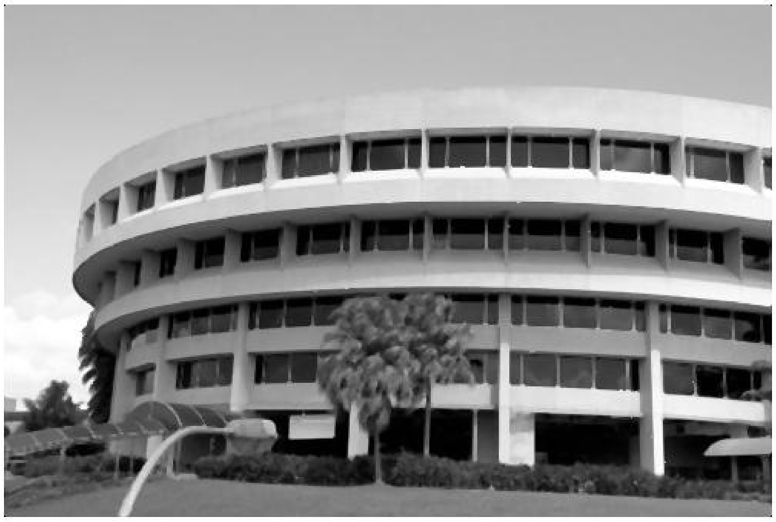


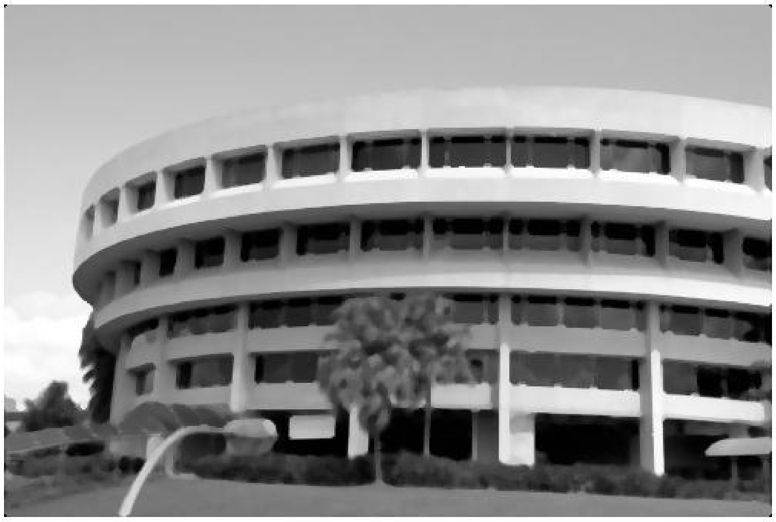
The speckle noise was not removed by Gaussian filters. Gaussian filters are better at handling Gaussian noise.

2.4 Median Filtering

Median filtering is a special case of order-statistic filtering. For each pixel, the set of intensities of neighbouring pixels (in a neighbourhood of specified size) are ordered. Median filtering involves replacing the target pixel with the median pixel intensity. Repeat steps (b)-(e) in Section 3, except instead of using h(x,y) and conv2, use the command medfilt2 with different neighbourhood sizes of 3x3 and 5x5. How does Gaussian filtering compare with median filtering in handling the different types of noise? What are the tradeoffs?

```
P1 = medfilt2(P);
P2 = medfilt2(P, [5 5]);
```



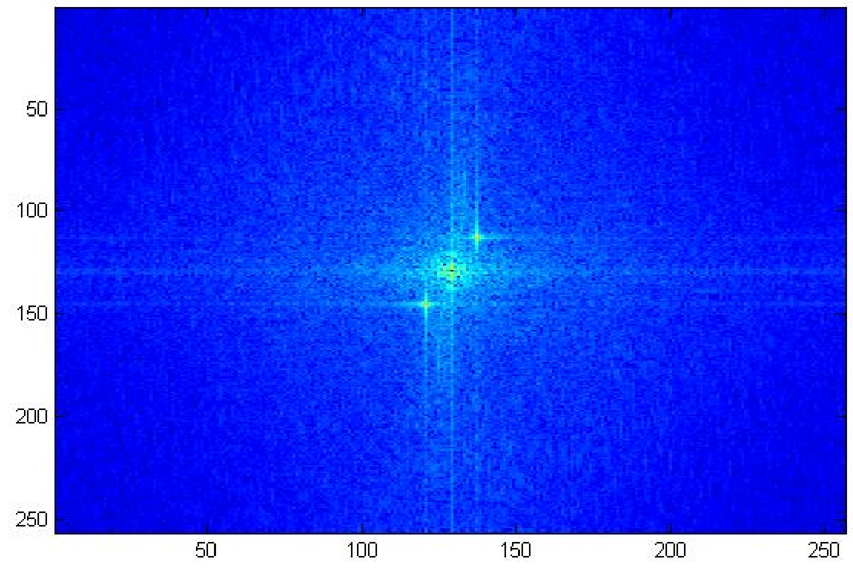


Gaussian filtering is better in removing Gaussian noise while median filtering is better in removing the speckle noise. Median filtering makes the edges in the image blurred.

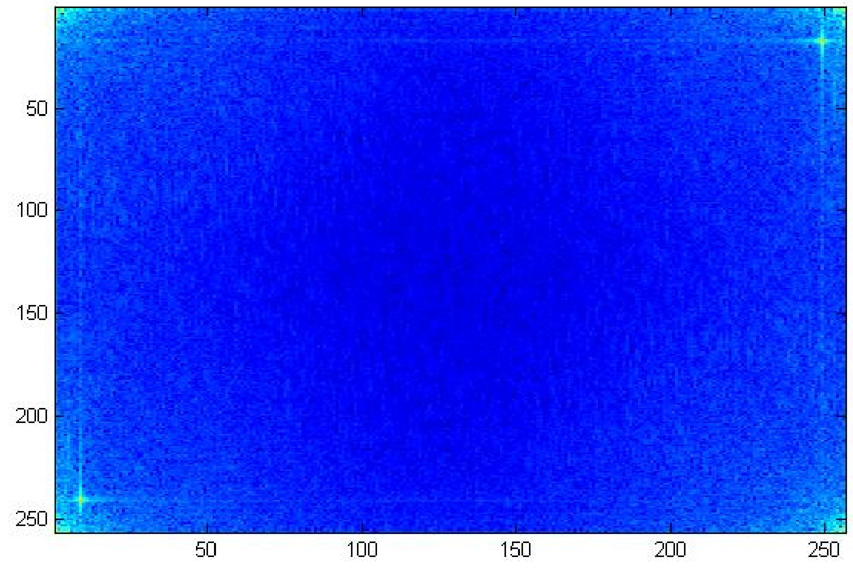
2.5 Suppressing Noise Interference Patterns

b. Obtain the Fourier transform F of the image using fft2, and subsequently compute the power spectrum S. Note that F should be a matrix of complex values, but S should be a real matrix.

```
F = fft2(P);
S=abs(F).^2;
imagesc(fftshift(S.^0.1));
```



c. Redisplay the power spectrum without fftshift. Measure the actual locations of the peaks.You can read the coordinates off the x and y axes, or you can choose to use the ginput function.



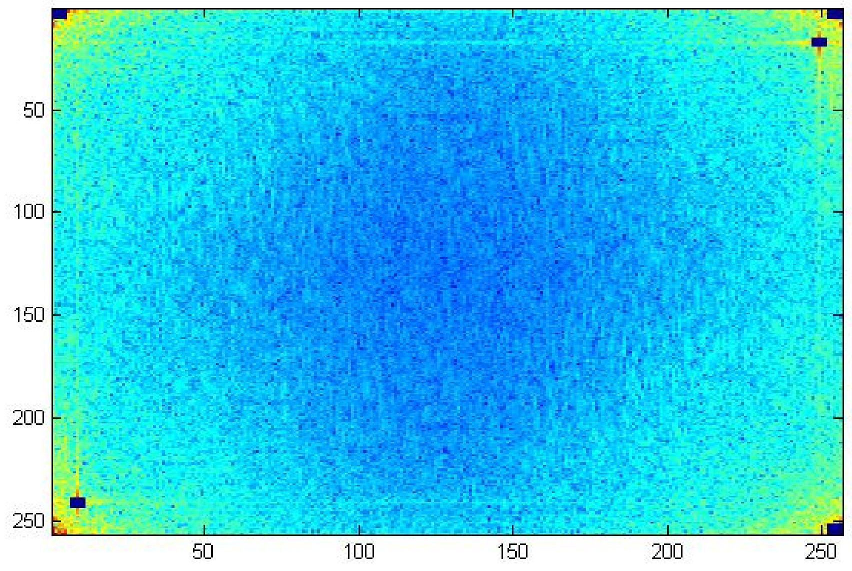
The peaks are (249,17), (255,1), (1,1), (255,256) and (9, 241).

d. Set to zero the 5x5 neighbourhood elements at locations corresponding to the above peaks in the Fourier transform F, not the power spectrum. Recompute the power spectrum and display it as in step (b).

```
F(15:19, 247:251) = 0;
```

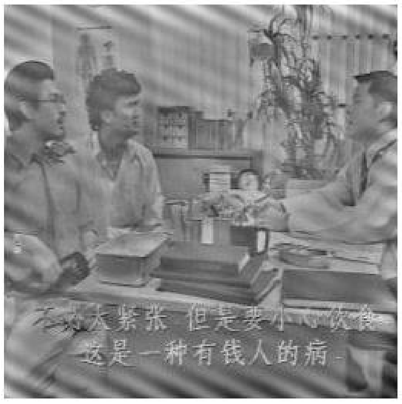
And do the same to the other peaks. Then compute the power spectrum again:

```
S=abs(F).^2;
```



e. Compute the inverse Fourier transform using ifft2 and display the resultant image. Comment on the result and how this relates to step (c). Can you suggest any way to improve this?

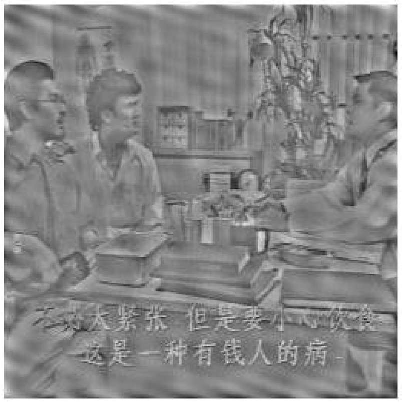
```
P2=ifft2(F);
P2=real(P2);
P2=P2+130;
P2=uint8(P2);
imshow(P2)
```



The parallel lines were not totally removed.

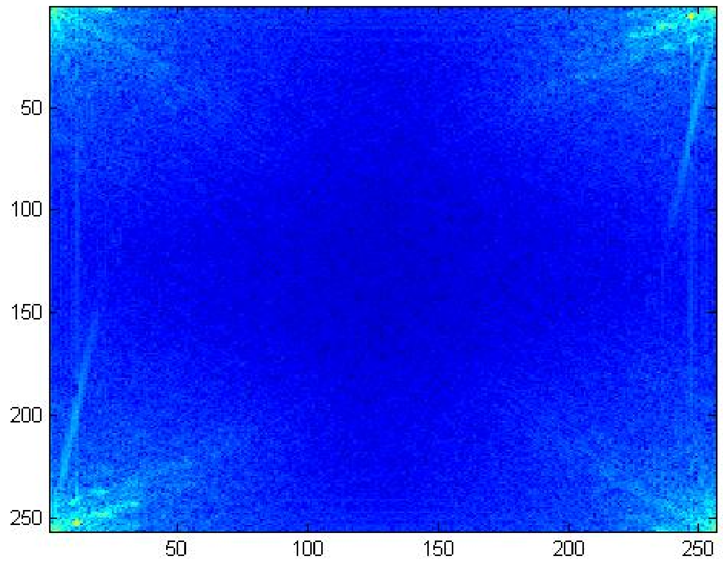
I iterate through the pixels again and set the threshold=3*10^9

```
for i = 1:256
    for j = 1:256
        if S(i,j)> 3*10^9
            for u=-2:2
                for v = -2:2
                    if i+u>0 && i+u<257 && j+v>0 && j+v<257
                        F(i,j)=0;
                    end
                end
            end
        end
    end
end
```



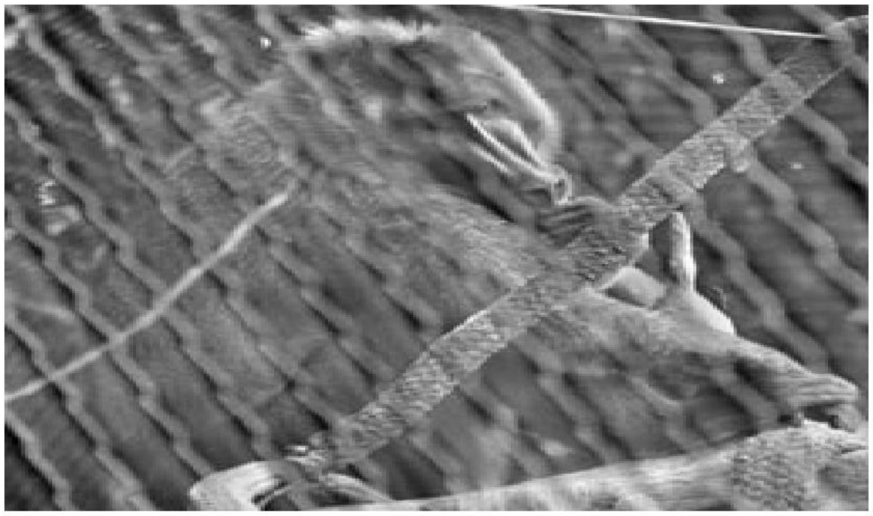
And this time most of the parallel lines are removed. And redo this whole process can further remove the undesired lines as well.

f. Download the image 'primate-caged.jpg' from edveNTUre which shows a primate behind a fence. Can you attempt to “free” the primate by filtering out the fence? You are not likely to achieve a clean result but see how well you can do.



```
F(4:8, 245:249)=0;
F(1:1, 1:1)=0;
F(250:254, 9:13)=0;
F(255:256, 1:4)=0;
F(248, 20:23)=0;
F(10, 235:238)=0;
F(2, 255)=0;
F(253, 252)=0;
P2=ifft2(F);
P2 = real(P2);
P2=P2+132;
P2=P2*255/298;
P2=uint8(P2);
```

```
imshow(P2);
```



It is hard to achieve clean result as the lines from the cage are not parallel.

2.6 Undoing Perspective Distortion of Planar Surface

b. The ginput function allows you to interactively click on points in the figure to obtain the image coordinates. Use this to find out the location of 4 corners of the book, remembering the order in which you measure the corners. [X Y] = ginput(4) Also, specify the vectors x and y to indicate the four corners of your desired image (based on the A4 dimensions), in the same order as above.

X = [143.5000 309.5000 257.0000 4.0000]

Y = [27.7500 46.2500 215.2500 159.2500]

x=[0 210 210 0]

y = [0 0 290 290]

c. Set up the matrices required to estimate the projective transformation based on the equation (*) above.

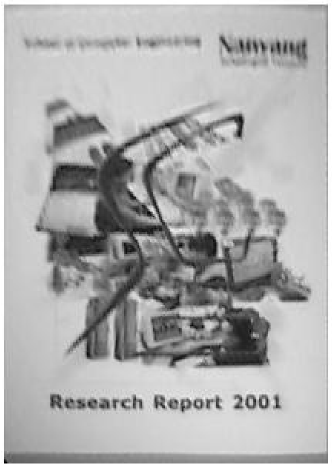
```
v=[0;0;210;0;210;290;0;290]
A = [
    X(1),Y(1),1,0,0,0,-x(1)*X(1), -x(1)*Y(1);
    0,0,0,X(1),Y(1),1,-y(1)*X(1), -y(1)*Y(1);
    X(2),Y(2),1,0,0,0,-x(2)*X(2), -x(2)*Y(2);
    0,0,0,X(2),Y(2),1,-y(2)*X(2), -y(2)*Y(2);
    X(3),Y(3),1,0,0,0,-x(3)*X(3), -x(3)*Y(3);
    0,0,0,X(3),Y(3),1,-y(3)*X(3), -y(3)*Y(3);
    X(4),Y(4),1,0,0,0,-x(4)*X(4), -x(4)*Y(4);
    0,0,0,X(4),Y(4),1,-y(4)*X(4), -y(4)*Y(4);
]
```

Does the transformation give you back the 4 corners of the desired image?

Yes.

e. Display the image. Is this what you expect? Comment on the quality of the transformation and suggest

reasons.



Yes. Some of the text on the book is not very clear. We can choose more than four points to make the transformation system more accurate.