NANYANG TECHNOLOGICAL UNIVERSITY

CZ4003 Computer Vision

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

Name: Zhou Hongyu

U1722662D

1. Objectives

This laboratory aims to introduce image processing in MATLAB context. In this laboratory you will:

. Become familiar with the MATLAB and Image Processing Toolbox software package.

. Experiment with the point processing operations of contrast stretching and histogram equalization.

. Evaluate how different Gaussian and median filters are suitable for noise removal.

. Become familiar with the frequency domain operations

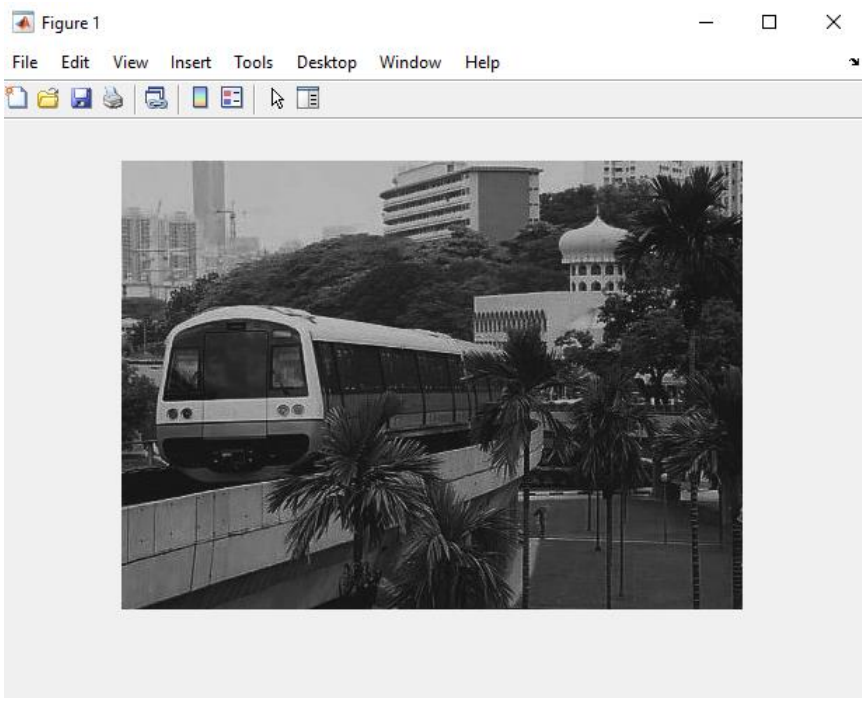
. Understand imaging geometry.

2. Experiments

2.1 Contrast Stretching

ab. Original image

```
>> Pc = imread('mrt-train.jpg');
>> whos Pc
>> P = rgb2gray(Pc);
>> imshow(P);
```



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

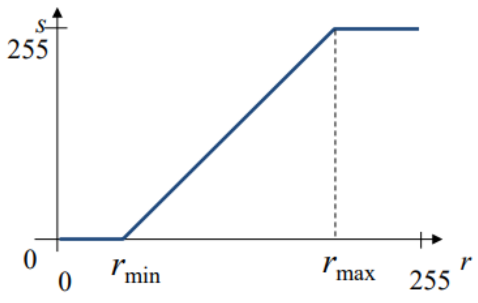
```
>> min(P(:)), max(P(:))
```

```
ans =             ans =

  uint8             uint8

   13                204
```

Then from the output, we know that the minimum intensity is 13 and maximum is 204

d. Contrast stretching

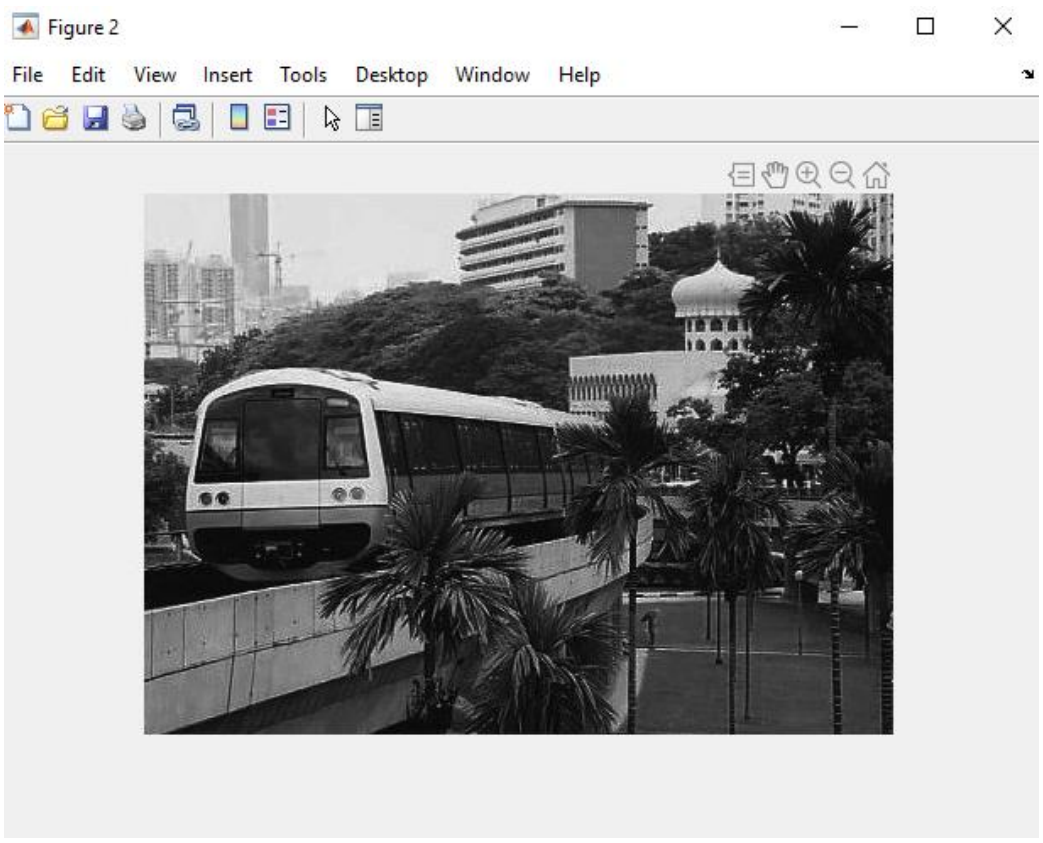
Then the formula for contrast stretching is followed:



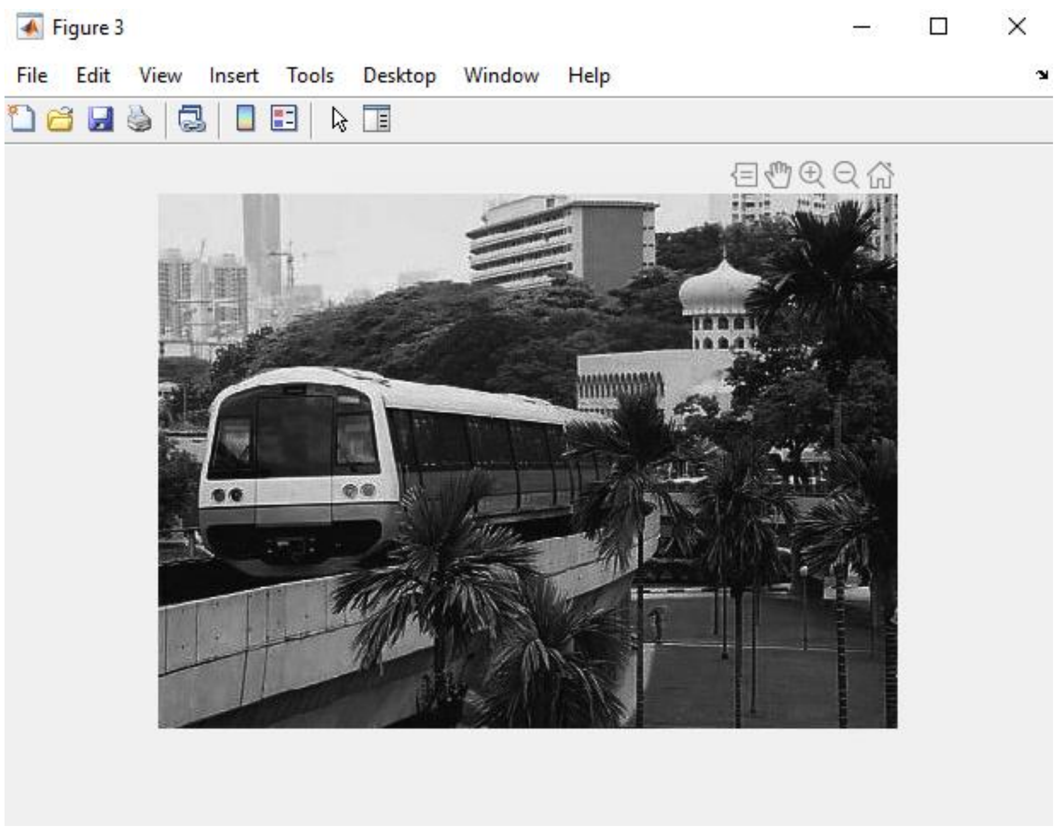
$$s = \begin{cases} 0 & r \le r_{\min} \\ \dfrac{255(r - r_{\min})}{r_{\max} - r_{\min}} & r_{\min} < r < r_{\max} \\ 255 & r \ge r_{\max} \end{cases}$$

```
>>P2 = imsubtract(P, 13);
>>P2 = immultiply(P2, (255/(204-13)));
>>min(P2(:)), max(P2(:))
>> imshow(P2);
```

After contrast stretching, the min value is 0 and max value is 255



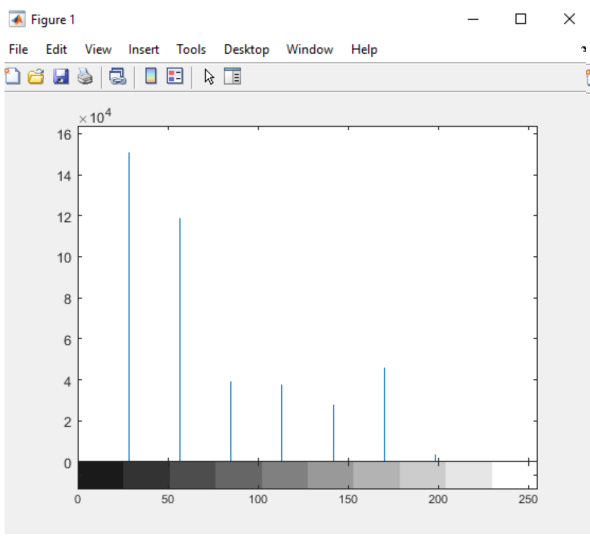
```
>> imshow(P2, []);
```

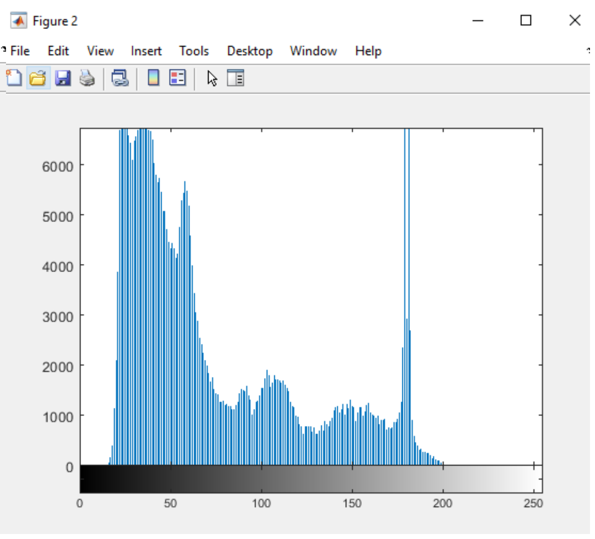


After comparing those two images with the original image, we can clearly see that we have achieve contrast stretching, which is also show by that the min and max value changed to 0 and 255.

2.2 Histogram Equalization

a. Display the image intensity histogram of P using 10 bins by >> imhist(P,10); and also Display the image intensity histogram of P using 256 bins by >> imhist(P,256);





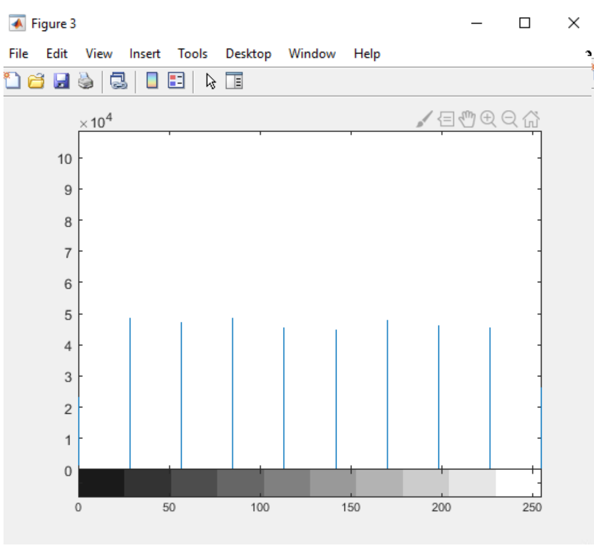
Difference between those two histograms:

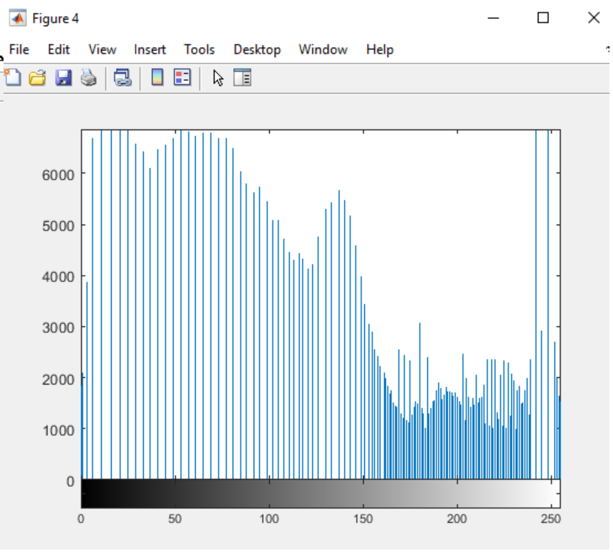
The total number of pixels are the same. But we divided the second image into more bins then for each bin the count of pixels is greater than the second one. The second histogram owns more bins, which means that it contains more details and information of this image. We could see that the peak value in first histogram is around 25 but in the second one it is around 25-50 and also around 180.

b. Next, carry out histogram equalization as follows:

```
>> P3 = histeq(P,255);
```

And redisplay the histograms for P3 with 10 and 256 bins.



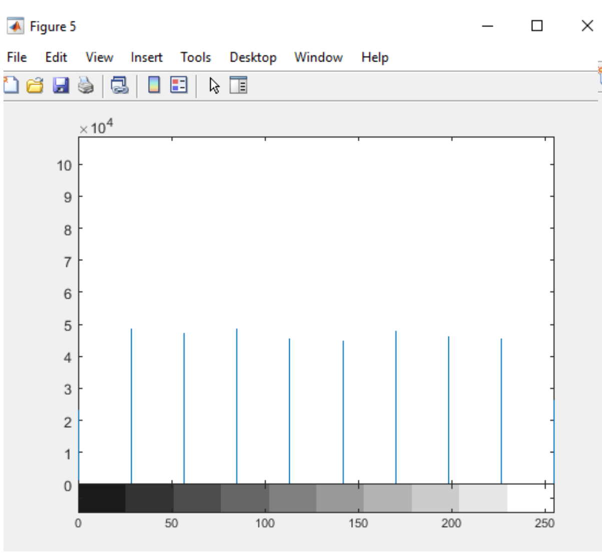


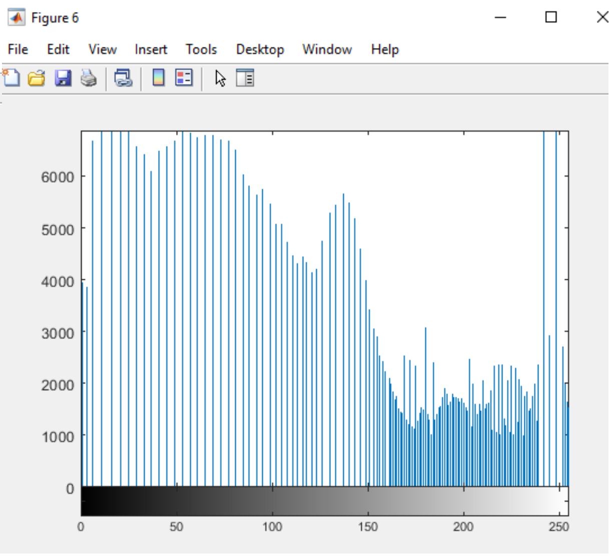
Those two histograms are more equalized than that in part a. Which means we successfully done the histogram equalization. Then the first histogram is more equalized than the second one. The number of each grey level may not be the same but when we divided the gray levels to be 10 bins, the number of pixels in each bin would be around the same. For first histogram, there are some bins with zero pixels but after equalization all the bins are nonempty. In the second histogram, there are still some bins with no pixel.

c. Rerun the histogram equalization on P3

```
>> P4 = histeq(P3, 255);
>> figure
>> imhist(P4, 10);
>> figure
>> imhist(P4, 256);
```

The histogram does not become more uniform. Re-applying the algorithm does not change the result as the lecture note said. The total number of pixels will not change, in the first equalization process, the pixels are already mapped to new grey level based on the average value of pixels and total number of pixels which are less than the original grey level. Then in the second process, the average number of pixels is unchanged because the total number of pixels is not changed. Also, the total number of pixels which are less than the original grey level does not change. Then re-apply the algorithm will not change anything.



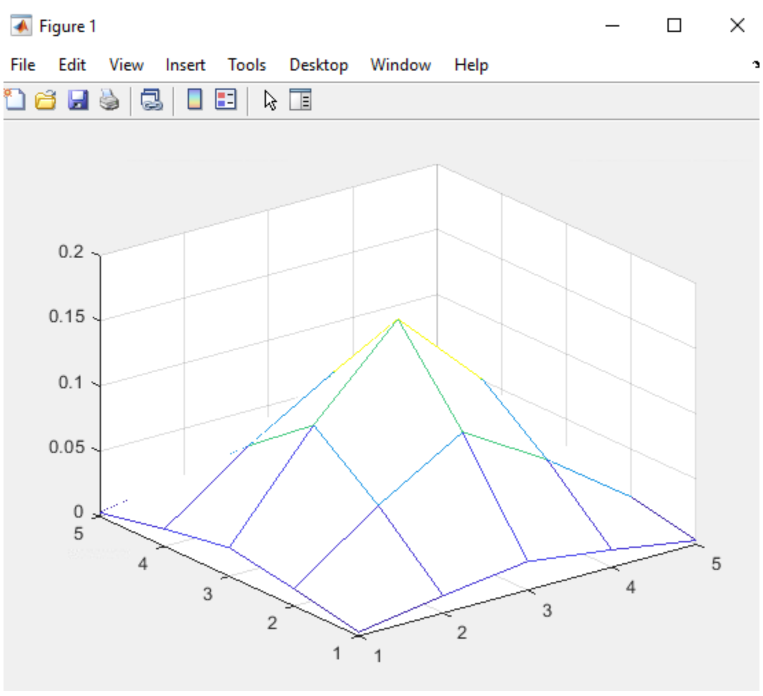


2.3 Linear Spatial Filtering

a. Generate the following filters:

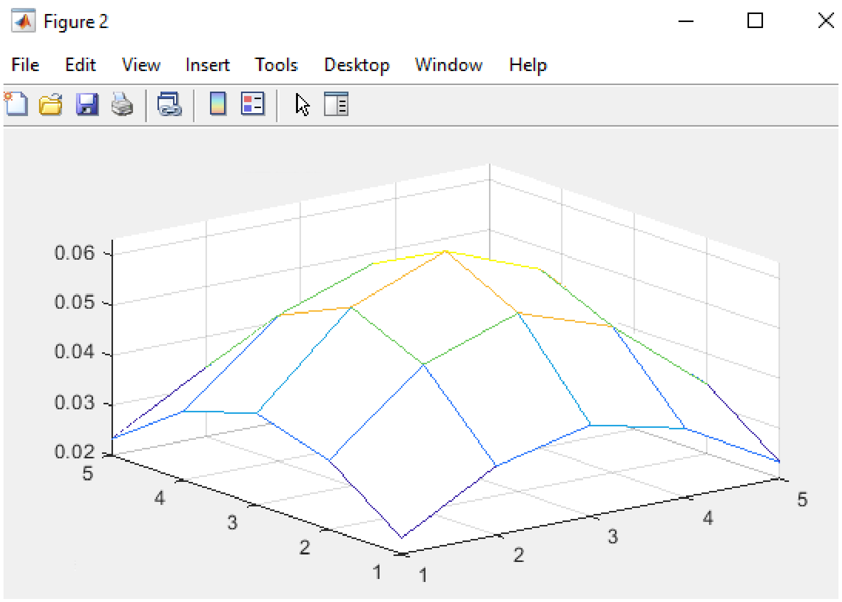
1.Y and X-dimensions are 5 and σ = 1.0

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



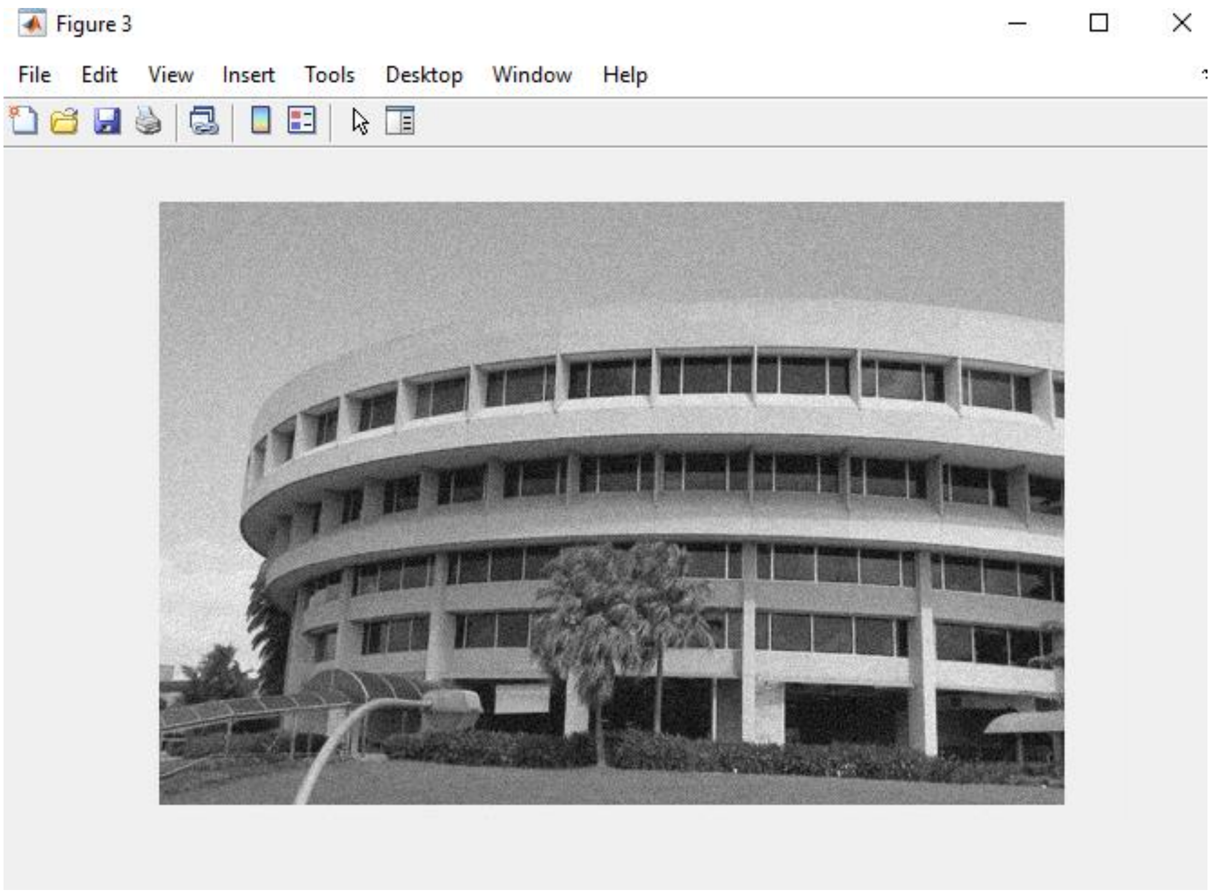
2.Y and X-dimensions are 5 and σ = 2.0

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



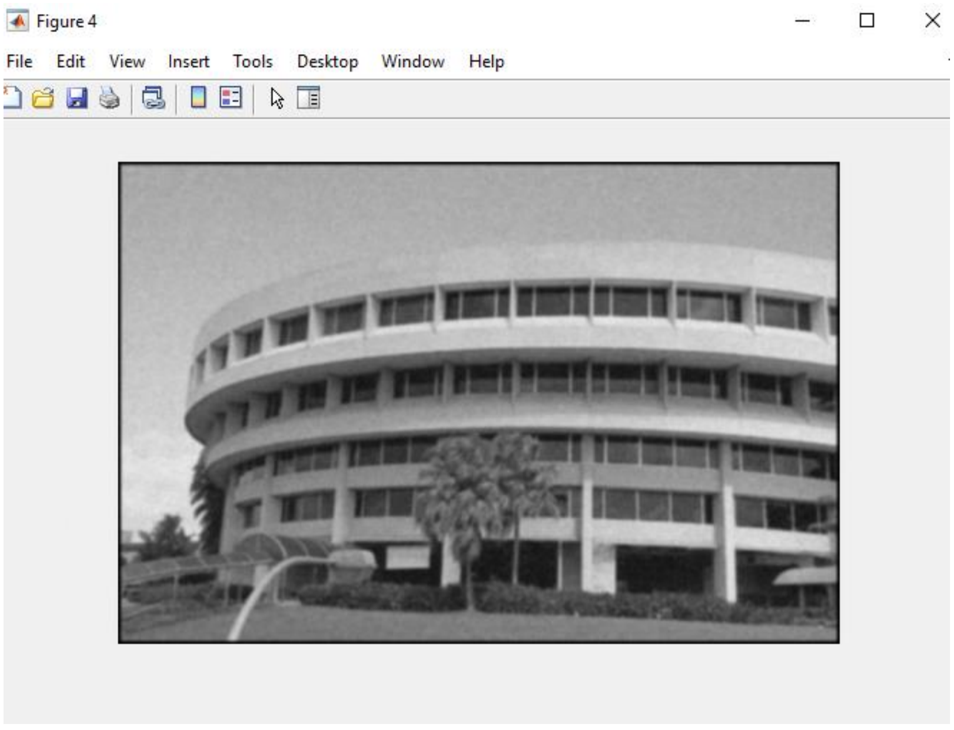
b. Original Image

```
>> P = imread('Images/lib-gn.jpg');
>> figure
>> imshow(P);
```

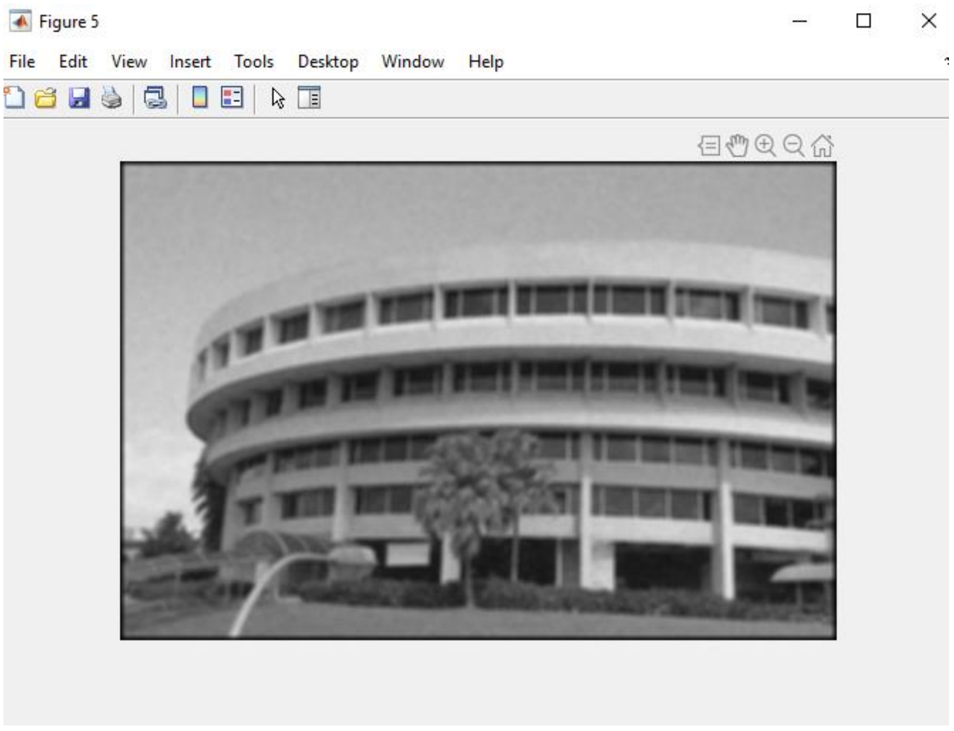


c. Filter the image using the linear filters that you have created above using the conv2 function and display the resultant images.

```
>>P1 = conv2(P,h1);
>>figure
>>imshow(uint8(P1))
```



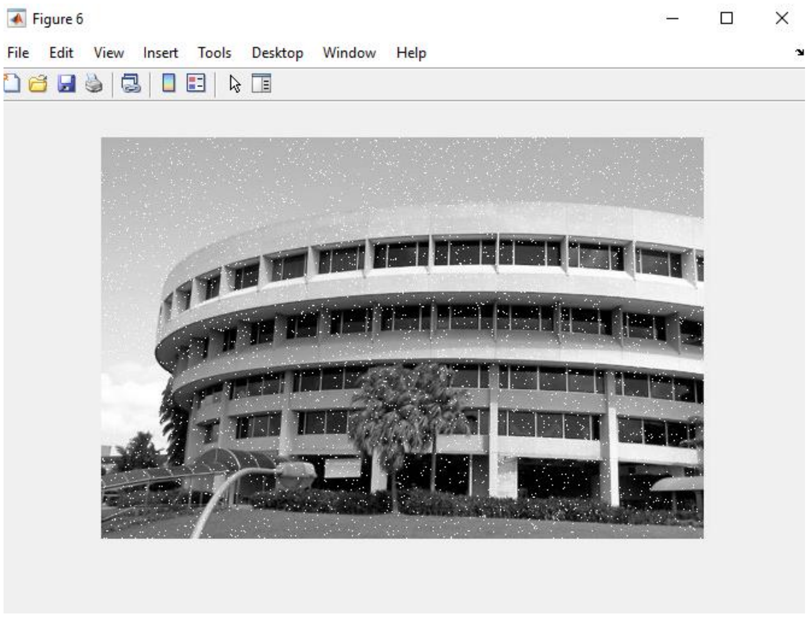
```
>>P2 = conv2(P,h2);
>>figure
>>imshow(uint8(P2))
```



The first filter has σ = 1.0 and the second one is σ = 2.0. We could see that for both images, some noise has been removed. There is less noise in the second picture than in the first picture, which means that the second filter is more effective in removing noise than the first one. But meanwhile, the second picture is more blurred and has lower resolution than the first one. The tradeoff is that removing noise better will cause lower quality pictures.

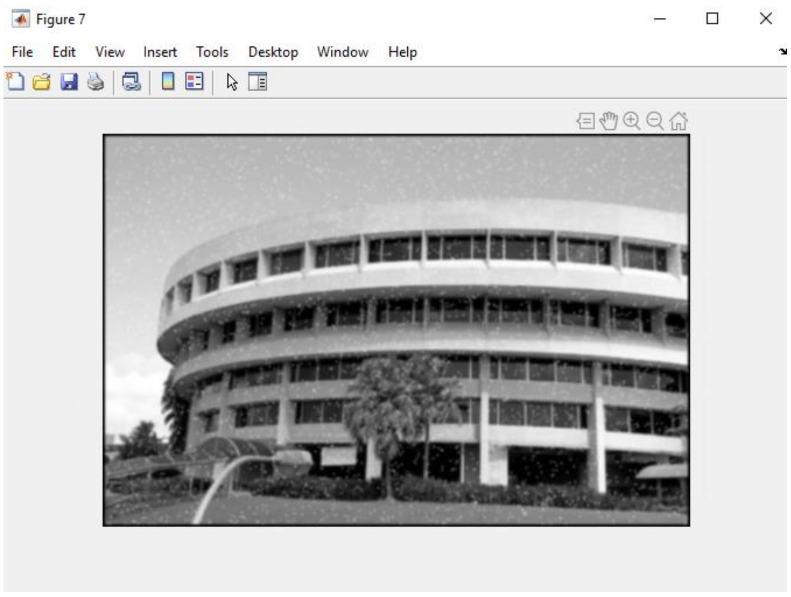
d. Download the image 'ntu-sp.jpg' and view it. Notice that this image has additive speckle noise.

```
>>P3 = imread('Images/lib-sp.jpg');
>>figure
>>imshow(P3);
```

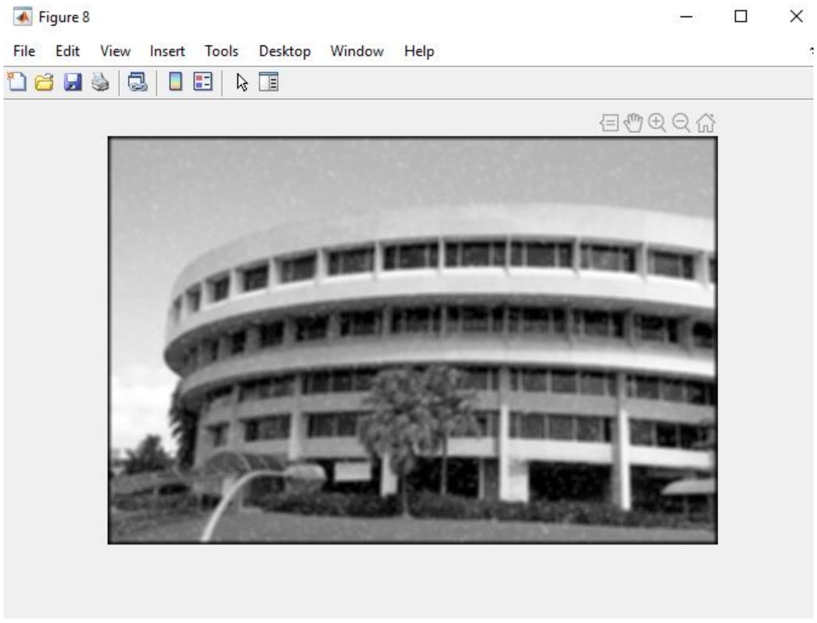


e. Repeat step (c) above

```
>>P4 = conv2(P3,h1);
>>figure
>>imshow(uint8(P4))
```



```
>>P5 = conv2(P3,h2);
>>figure
>>imshow(uint8(P5))
```

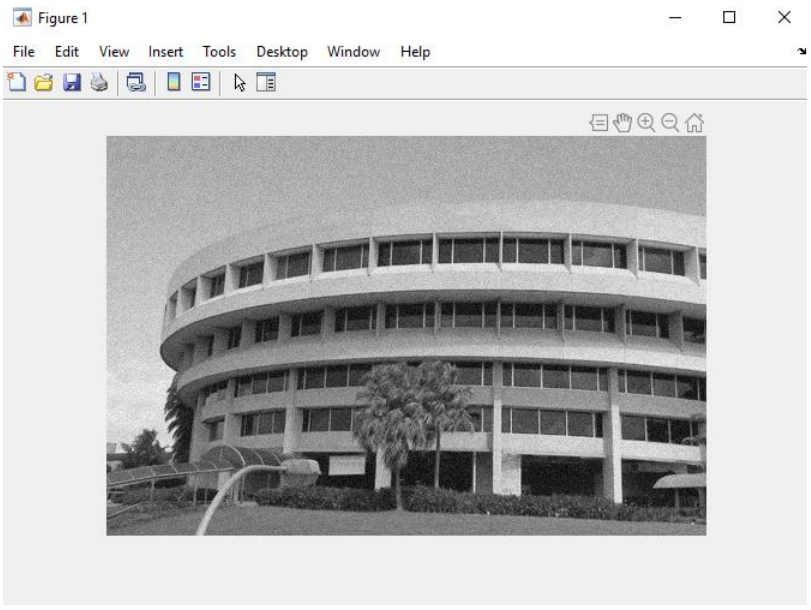


We can see that the Gaussian averaging filters are better at handling Gaussian noise than speckle noise. This is because that the filter will take the nearby pixels and assign different weights for the pixels. But for speckle noise, those noise are totally different from the neighbor pixels then if we apply gaussian averaging filters, it will also be different from the nearby pixels. So the result of applying filter for speckle noise is worse than Gaussian noise.

2.4 Median Filtering

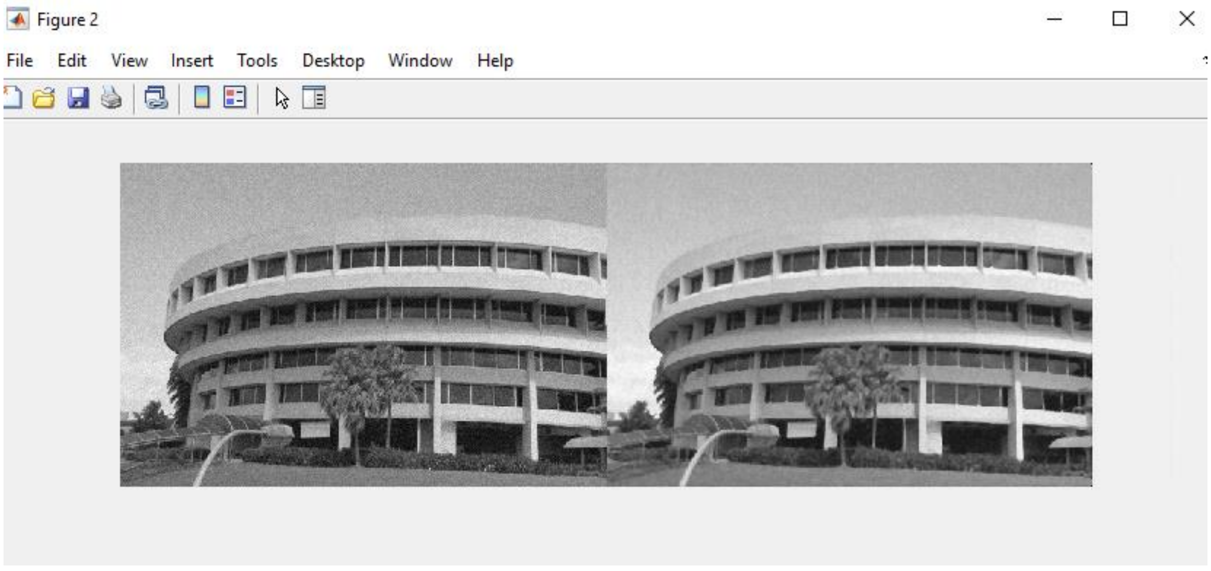
b. Read the image with Gaussian noise

```
>> P = imread('Images/lib-gn.jpg');
>> figure
>> imshow(P);
```

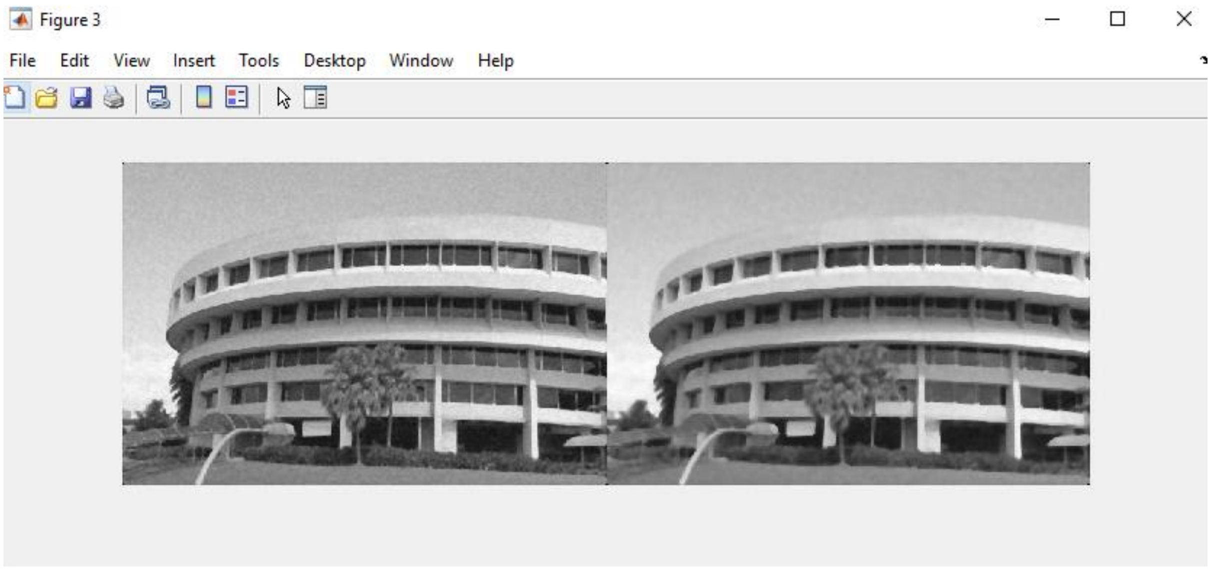


c. Filter the image using medfilt2 with different neighbourhood sizes of 3x3 and 5x5.

```
>> P2 = medfilt2(P); % Default filter size is 3x3
>> figure
>> imshowpair(P, P2, 'montage');
```

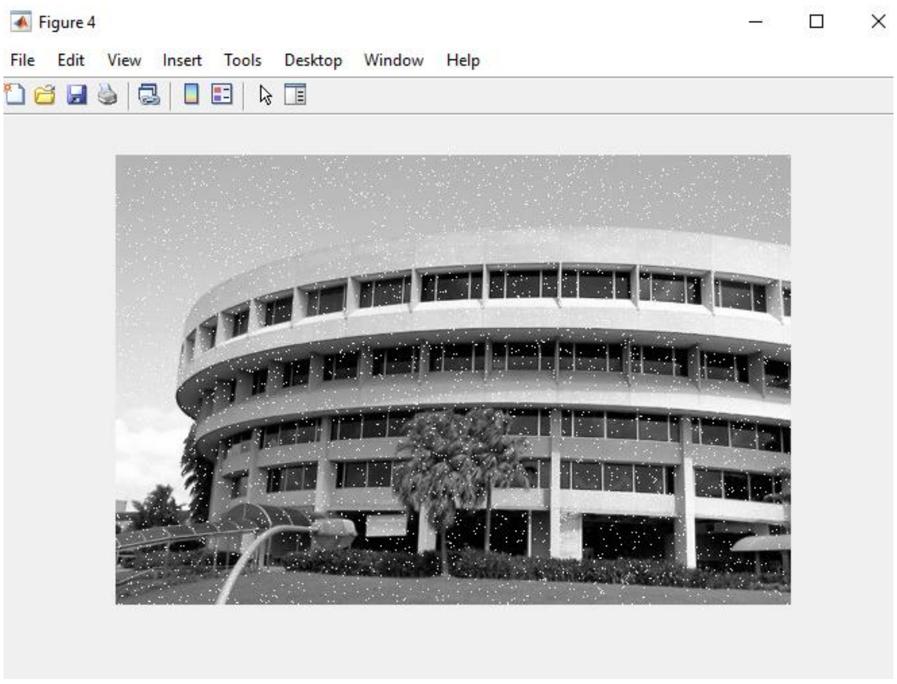


```
>> P3 = medfilt2(P, [5, 5]);
>> figure
>> imshowpair(P2, P3, 'montage');
```



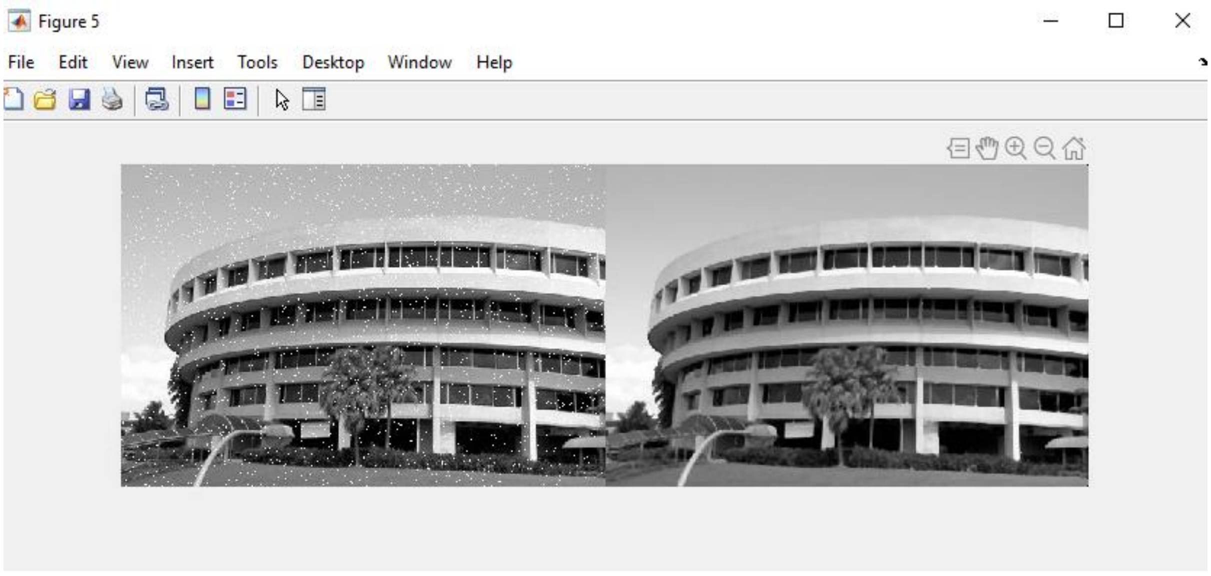
d. Read the image with speckle noise

```
>> P4 = imread('Images/lib-sp.jpg');
>> figure
>> imshow(P4);
```

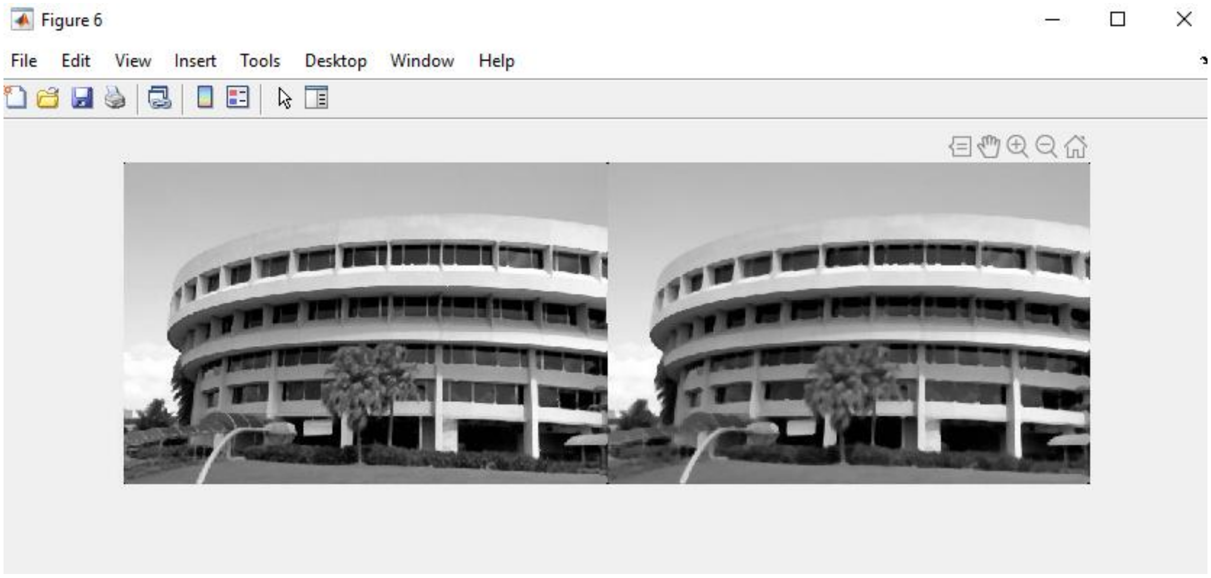


e. Filter the image using medfilt2 with different neighbourhood sizes of 3x3 and 5x5.

```
>> P5 = medfilt2(P4); % Default filter size is 3x3
>> figure
>> imshowpair(P4, P5, 'montage');
```



```
>> P6 = medfilt2(P4, [5, 5]);
>> figure
>> imshowpair(P5, P6, 'montage');
```



We could have the following findings from median and gaussian filters:

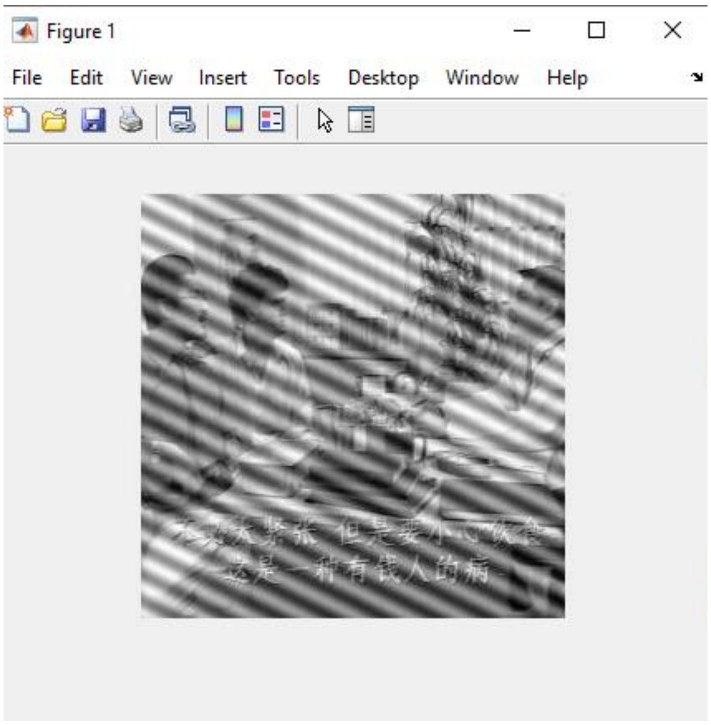
(1) For speckle noise, median filter works better than gaussian filters. And the 5*5 median filter will cause more smoothing of picture.
(2) For gaussian noise, both gaussian filter and median filter work for removing the noise. But gaussian filter works better. Median filter makes the edges of picture blurred. Bigger sigma for gaussian filter will remove noise better but will cause picture more blurred.
(3) For different noise, we can use suitable filters to get better result.
(4) The tradeoff is that bigger sigma for gaussian filter remove noise better but make picture more blurred. Bigger neighborhood size make picture more smooth but the edges will be blurred.

2.5 Suppressing Noise Interference Patterns

Occasionally with poor television reception, parallel lines will be seen on the screen corrupting the desired image. These are interference patterns. Here we explore how bandpass filtering can be used to suppress such interference.

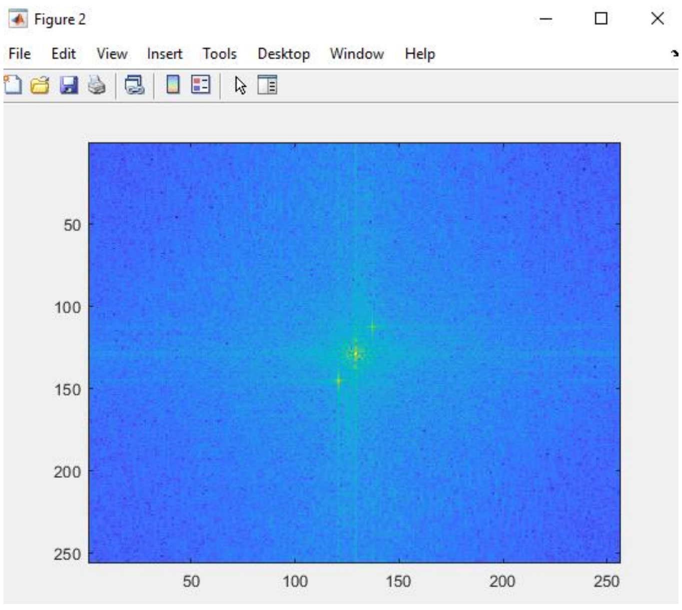
a. Read the image

```
>> P = imread('Images/pck-int.jpg');
>> imshow(P);
```



b. **Obtain the Fourier transform F of the image using fft2, and subsequently compute the power spectrum S.**

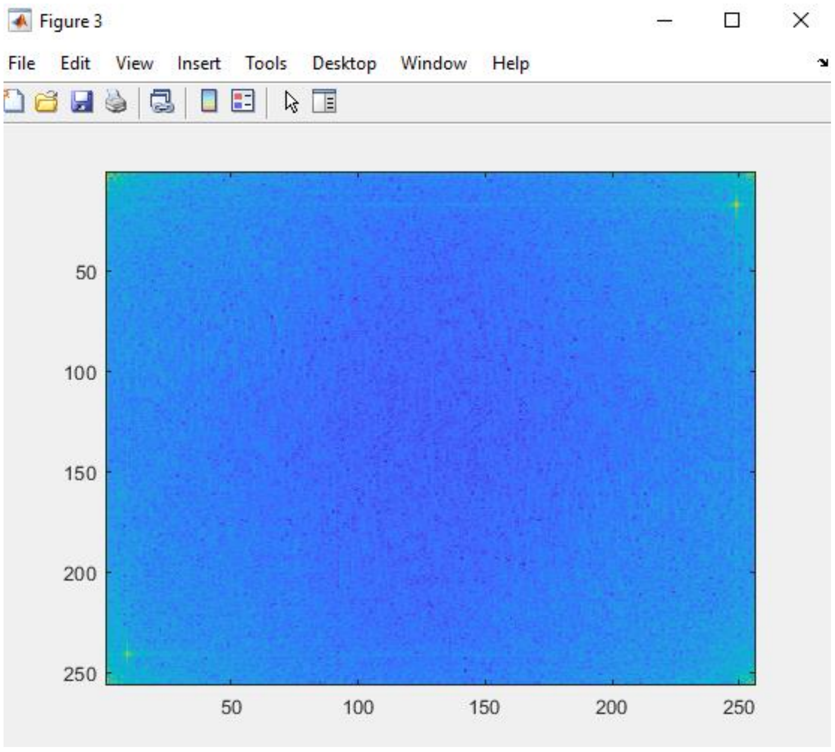
```
>> F = fft2(P);
% compute power spectrum S
>> S = abs(F);
>> figure
>> imagesc(fftshift(S.^0.1));
>> colormap('default');
```



In the above image, there are two distinct, symmetric frequency peaks that are isolated from the central mass. These frequency components correspond to the interference pattern.

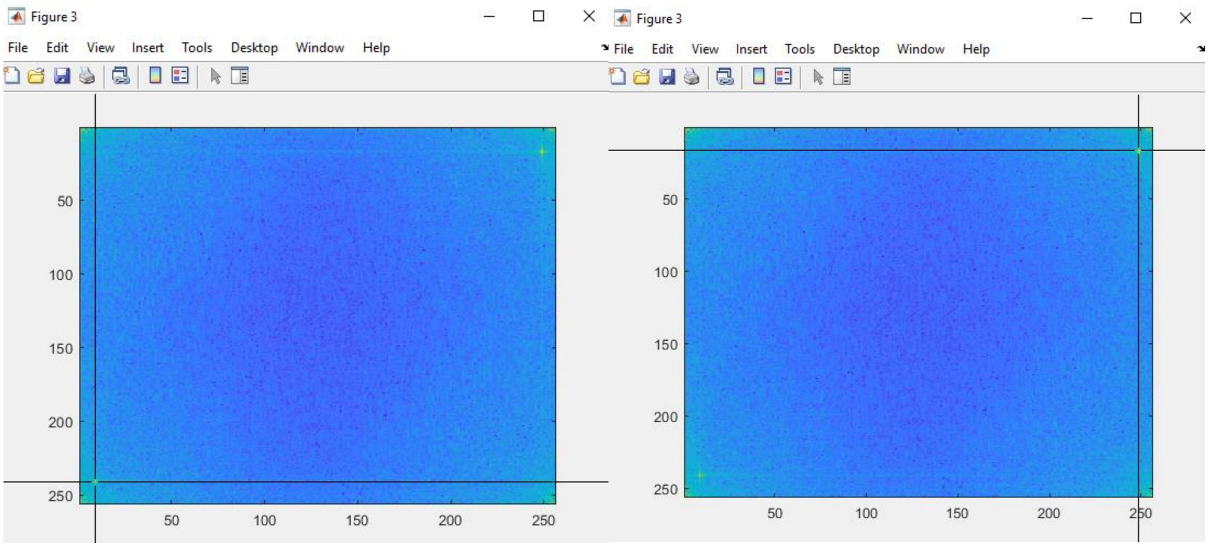
c. Redisplay the power spectrum without fftshift. Measure the actual locations of the peaks

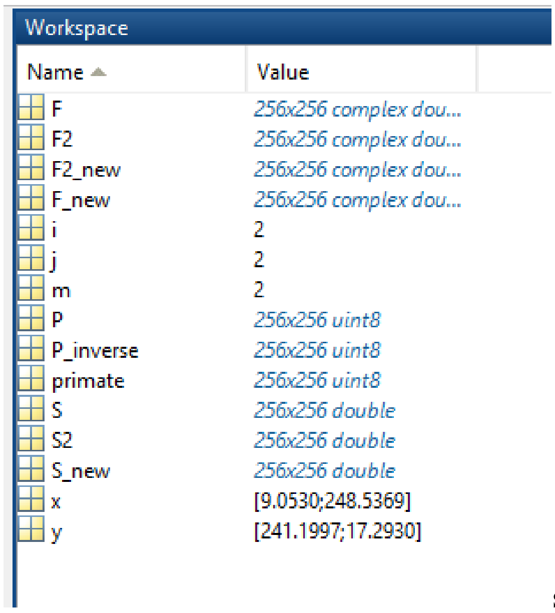
```
>> figure
>> imagesc(S.^0.1);
>> colormap('default');
```



To obtain the location of two peaks, we use ginput() function to get their locations.

```
>> F_new=F;
>> [x, y] = ginput(2);
```



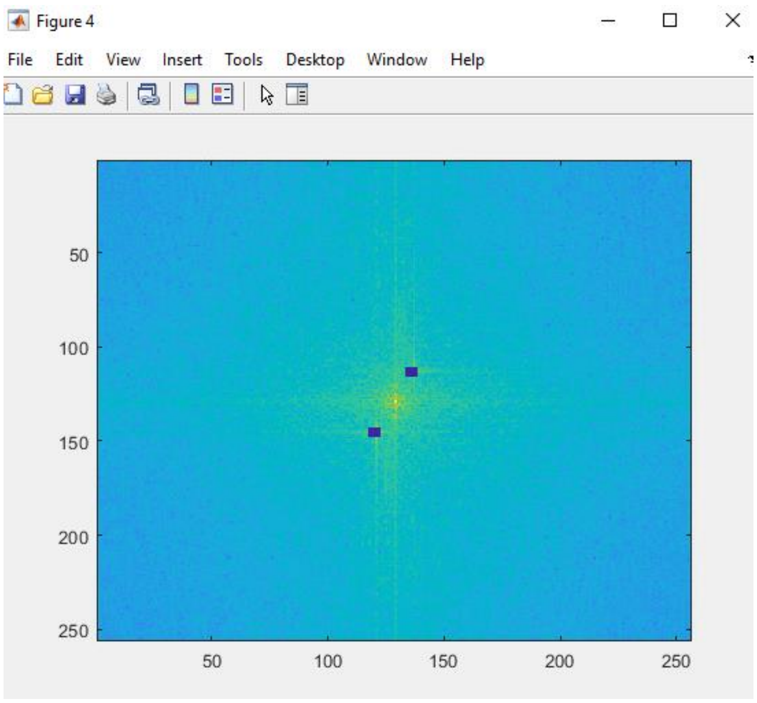


Name	Value
F	256x256 complex dou...
F2	256x256 complex dou...
F2_new	256x256 complex dou...
F_new	256x256 complex dou...
i	2
j	2
m	2
P	256x256 uint8
P_inverse	256x256 uint8
primate	256x256 uint8
S	256x256 double
S2	256x256 double
S_new	256x256 double
x	[9.0530;248.5369]
y	[241.1997;17.2930]

shows the peaks are (9,241) and (249,17)

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_new(239:243,7:11)=0;
%F_new(15:19,247:251)=0;
for i = 1: length(x)
      fprintf('Peaks: (%d, %d)\n', round(x(i)), round(y(i)));
      for j=-2:2
            for m=-2:2
                fprintf('At pixel: (%d, %d)\n', round(x(i))+j, round(y(i))+m);
              F_new(round(y(i))+m, round(x(i))+j) = 0;
            end
      end
end

>> S_new= abs(F_new);
>> figure
>> imagesc(fftshift(S_new.^0.1))
>> colormap('default')
```



e. **Compute the inverse Fourier transform using ifft2 and display the resultant image.**

```
>> P_inverse = uint8(ifft2(F_new));
>> figure
>> imshow(real(P_inverse));
```

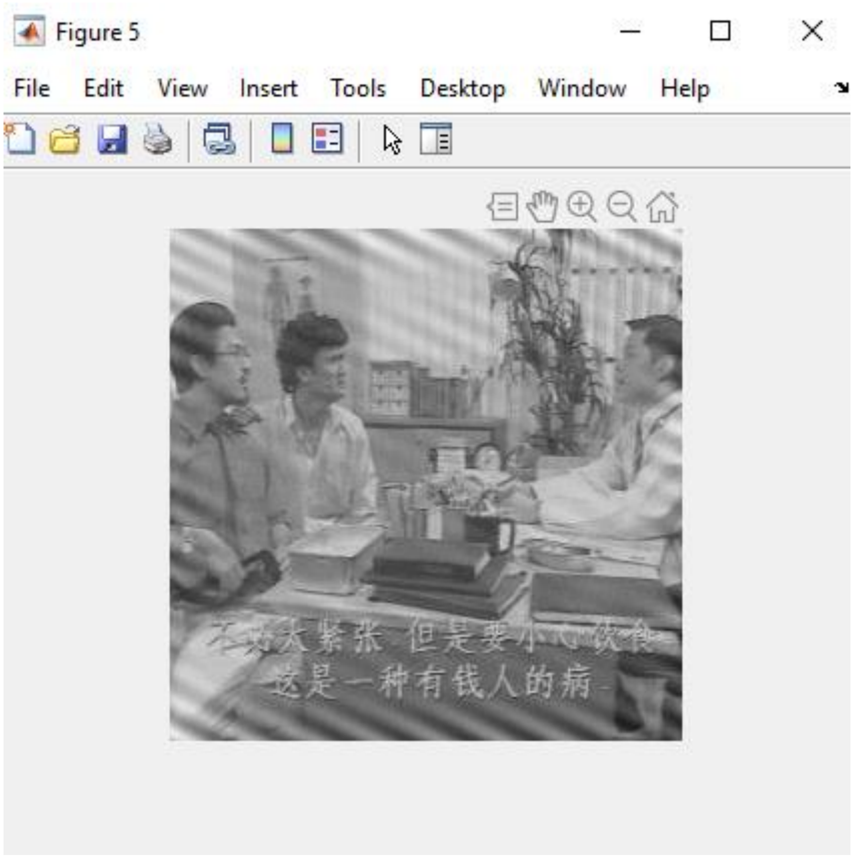


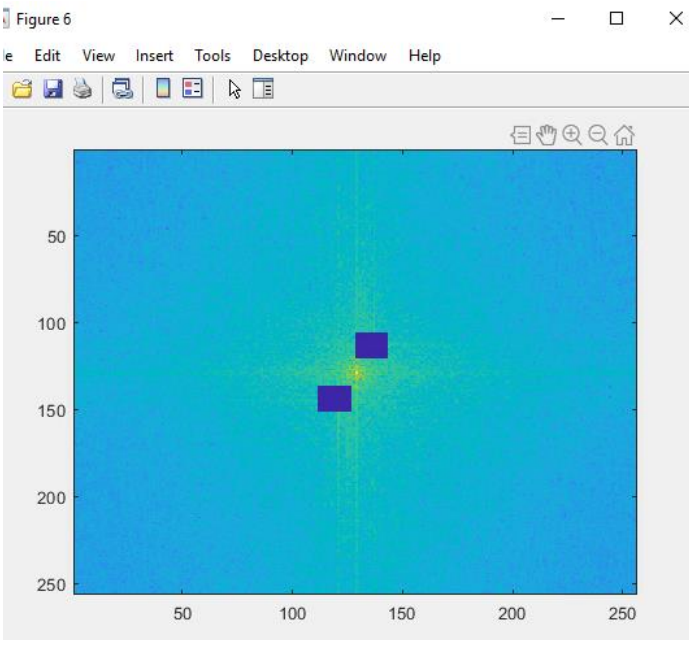
Figure 5
File Edit View Insert Tools Desktop Window Help
不 为 太 紧 张 但是要小心饮食
这是一种有钱人的病

Most of the interference in the picture has been removed especially for the center part. This is because that in step c, we can find that the low frequency part is represent the interference.

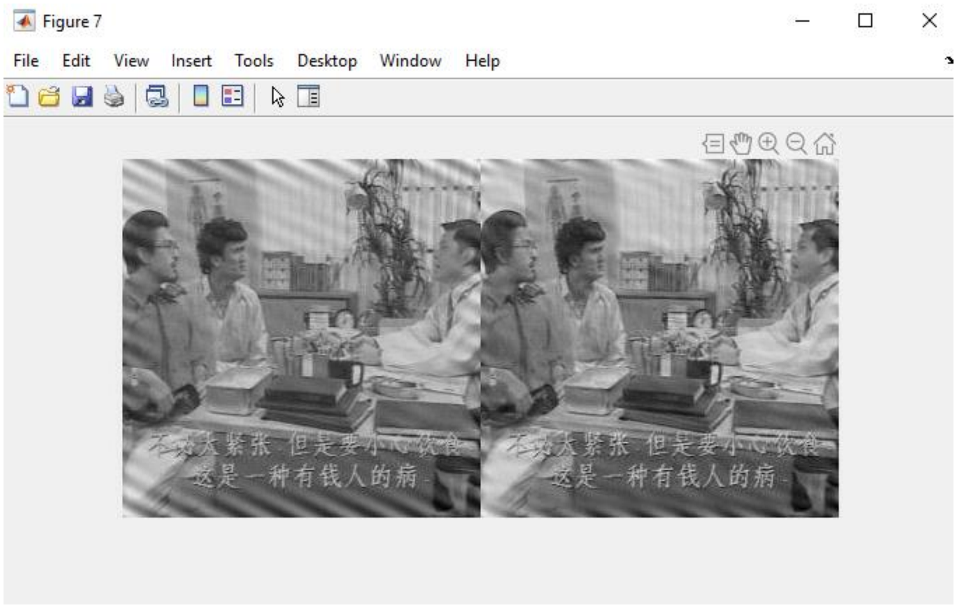
And then we set those part to be zero.

f. To improve it, we could set other low frequency part into zero and further remove the interference. We Set to zero the 15x15 neighbourhood elements at locations corresponding to the above peaks in the Fourier transform F

```
>> S_new_better= abs(F_new_better);
>> figure
>> imagesc(fftshift(S_new_better.^0.1))
>> colormap('default')
```



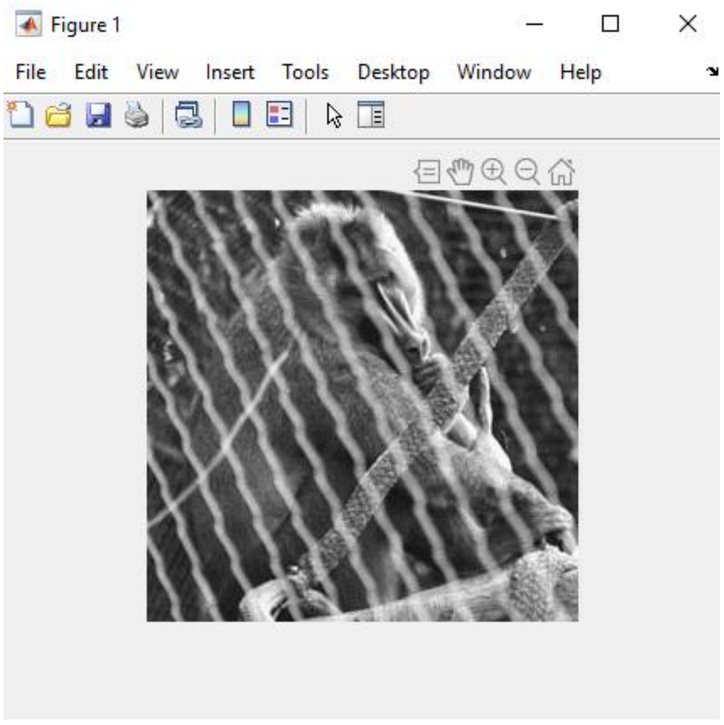
```
>> P_inverse_better = uint8(ifft2(F_new_better));
>> figure
>> imshowpair(real(P_inverse),real(P_inverse_better),'montage');
```



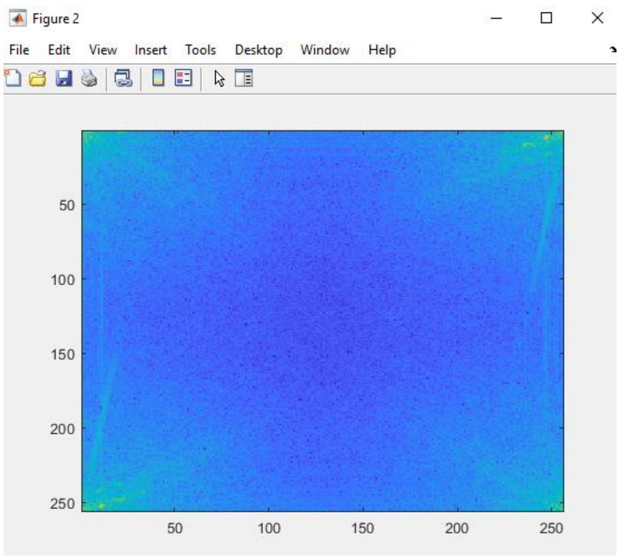
We can see that by removing more low frequency part, the more part of interference has been removed.

g. "Free" the primate by filtering out the fence

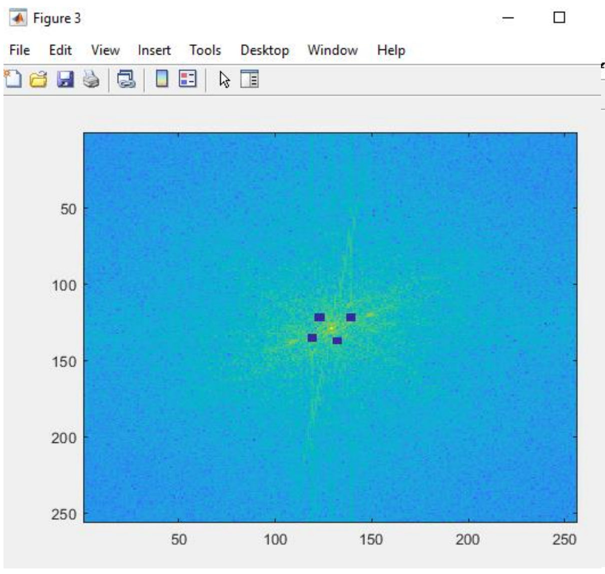
```
>> primate=imread('Images/primate-caged.jpg');
>> primate = rgb2gray(primate);
>> figure
>> imshow(primate);
```

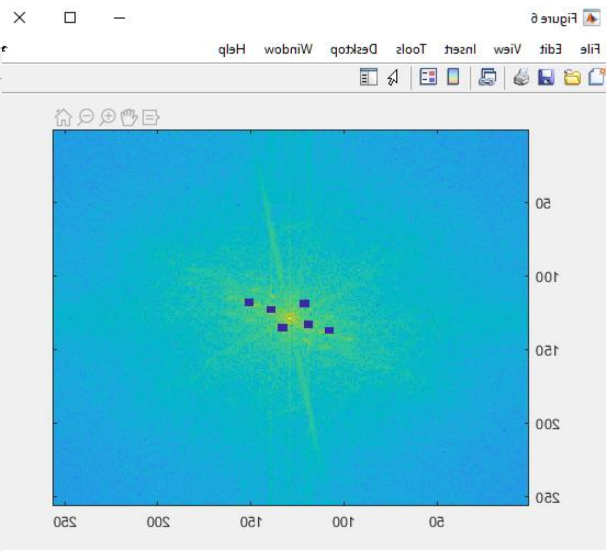


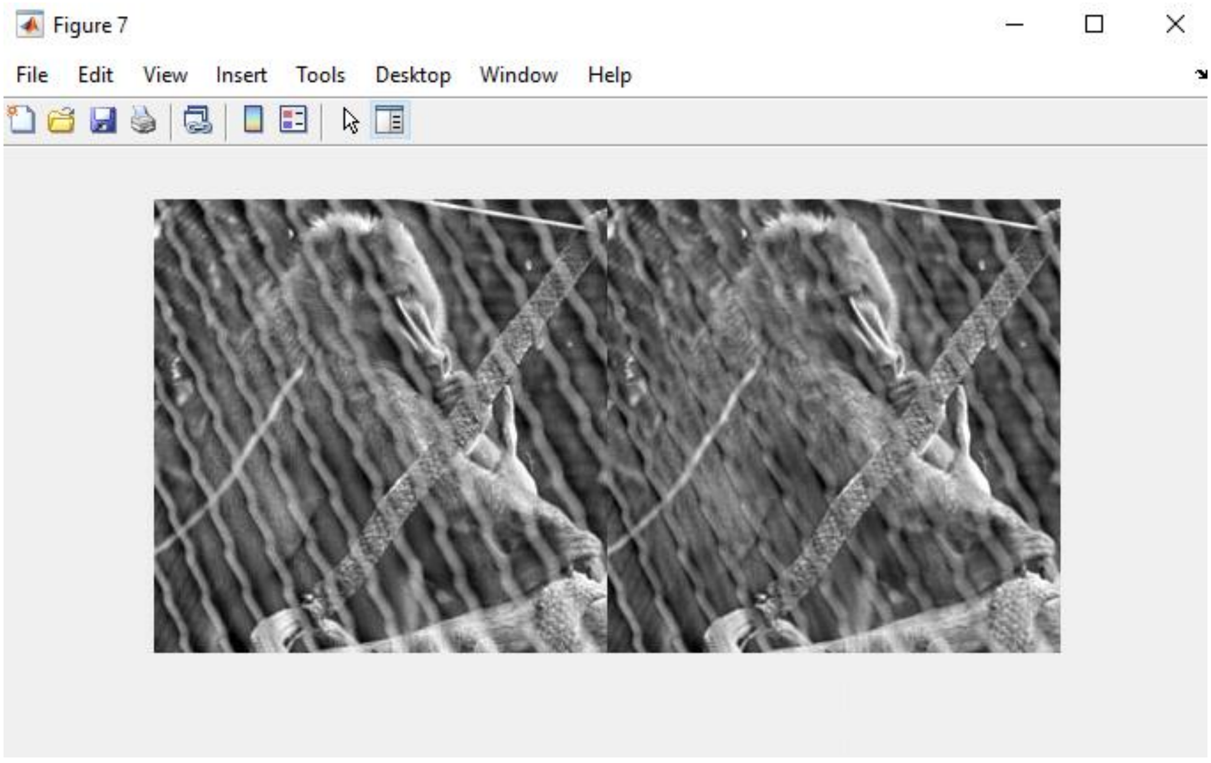
```
>> F2 = fft2(primate);
>> S2 = abs(F2);
>> figure
>> imagesc(S2.^0.1)
>> colormap('default')
>> F2_new = F2;
>> [x, y] = ginput(4);
>> for i = 1: length(x)
     fprintf('Peaks: (%d, %d)\n', round(x(i)), round(y(i)));
     for j=-2:2
          for m=-2:2
               fprintf('At pixel: (%d, %d)\n', round(x(i))+j, round(y(i))+m);
               F2_new(round(y(i))+m, round(x(i))+j) = 0;
               end
          end
     end
>> S2_new = real(F2_new);
>> figure
>> imagesc(fftshift(real(S2_new.^0.1)));
>> colormap('default');
```



```
>> F2_new_better=F2;
>> S2 = abs(F2_new_better);
>> figure
>> imagesc(S2.^0.1)
vcolormap('default')

>> [x2, y2] = ginput(6);
for i = 1: length(x2)
        fprintf('Peaks: (%d, %d)\n', round(x2(i)), round(y2(i)));
        for j=-2:2
              for m=-2:2
                    fprintf('At pixel: (%d, %d)\n', round(x2(i))+j, round(y2(i))+m);
                    F2_new_better(round(y2(i))+m, round(x2(i))+j) = 0;
            end
      end
end
>> S2_new_better= abs(F2_new_better);
>> figure
>> imagesc(fftshift(S2_new_better.^0.1))
>> colormap('default')

>> P2_inverse_better = uint8(ifft2(F2_new_better));
>> figure
>>imshowpair(real(P2_inverse),real(P2_inverse_better),'montage');
```





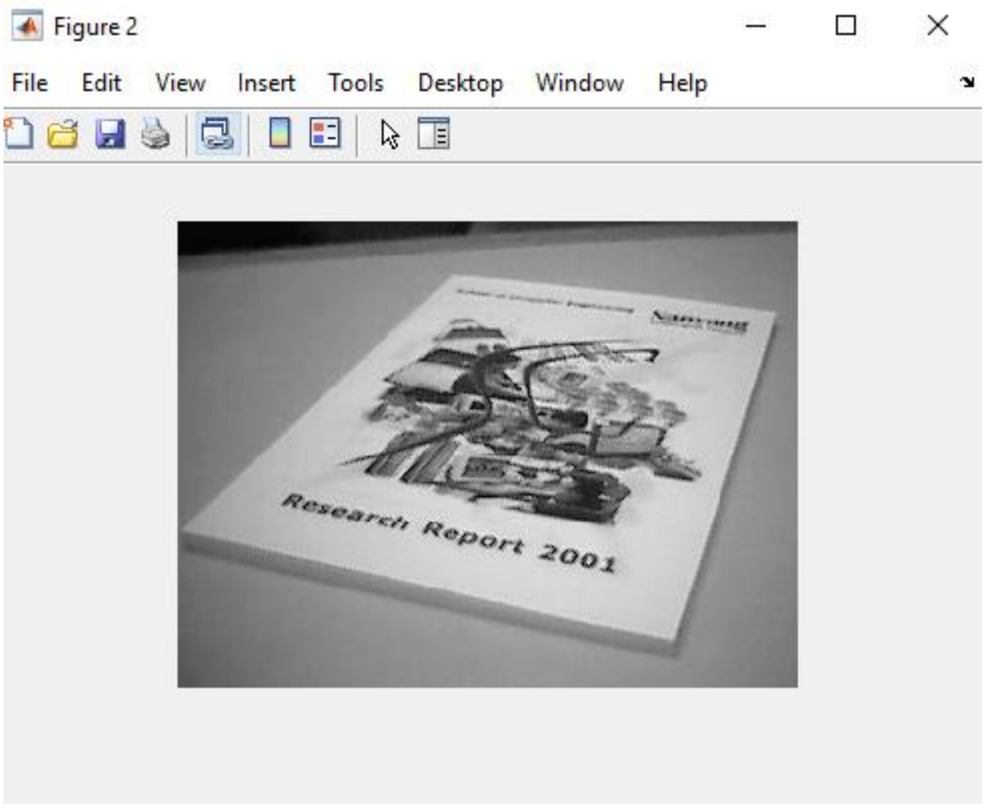


First, I have tried to move the fence by setting to zero the 5x5 neighbourhood elements at four locations corresponding to the above peaks in the Fourier transform. Then we can get the fftshift image as Figure3. Then after I compute the inverse Fourier transform using ifft2 and display the image, I found that the fence is not cleared much. Then I go to move the fence by setting to zero the 5x5 neighbourhood elements at six locations to get the better result.

2.6 Undoing Perspective Distortion of Planar Surface

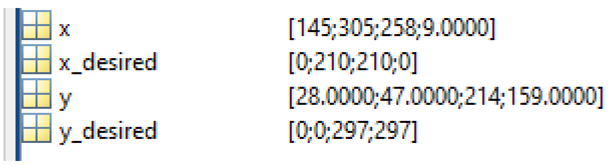
a. Read the image

```
>> P = imread('Images/book.jpg');
>> imshow(P);
>> P = rgb2gray(P);
>> figure
>> imshow(P);
```



b. 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, y] = ginput(4);
>> x_desired = [0;210;210;0];
>> y_desired = [0;0;297;297];
```



```
x              [145;305;258;9.0000]
x_desired      [0;210;210;0]
y              [28.0000;47.0000;214;159.0000]
y_desired      [0;0;297;297]
```

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

```
>> A =[[x(1),y(1),1,0,0,0,-x_desired(1)*x(1),-x_desired(1)*y(1)];
      [0,0,0,x(1),y(1),1,-y_desired(1)*x(1),-y_desired(1)*y(1)];
      [x(2),y(2),1,0,0,0,-x_desired(2)*x(2),-x_desired(2)*y(2)];
      [0,0,0,x(2),y(2),1,-y_desired(2)*x(2),-y_desired(2)*y(2)];
```

```
            [x(3),y(3),1,0,0,0,-x_desired(3)*x(3),-x_desired(3)*y(3)];
            [0,0,0,x(3),y(3),1,-y_desired(3)*x(3),-y_desired(3)*y(3)];
            [x(4),y(4),1,0,0,0,-x_desired(4)*x(4),-x_desired(4)*y(4)];
            [0,0,0,x(4),y(4),1,-y_desired(4)*x(4),-y_desired(4)*y(4)];];
      >> v = [x_desired(1);y_desired(1);x_desired(2);y_desired(2);x_desired(3);y_desired(3);
      >> x_desired(4);y_desired(4)];
      >> u =A\v;
      >> U = reshape([u;1], 3, 3)';
      >> w = U*[x'; y'; ones(1,4)];
      >> w = w ./ (ones(3,1) * w(3,:));
```

The output is as below. This is the correct coordinates. The transformation gives me back the 4 corners of the desired image.

Output:

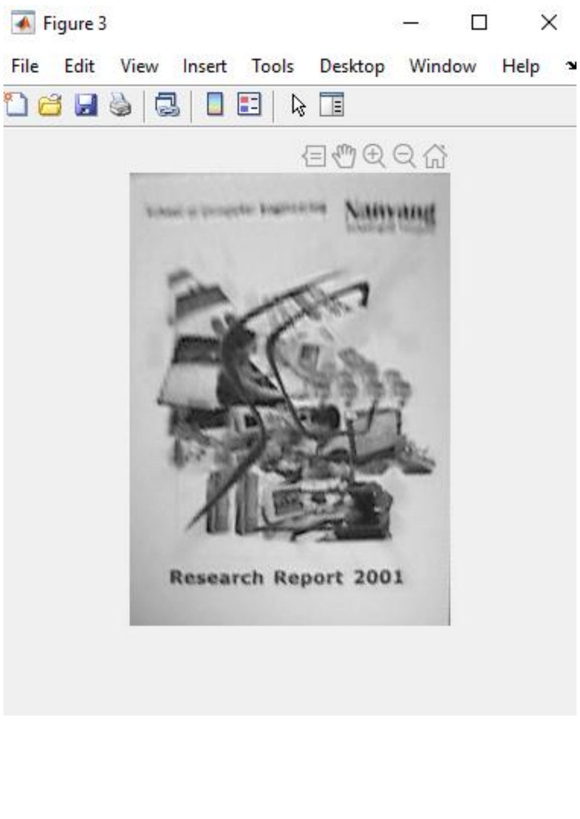
```
U =
     1.5004      1.5576 -261.1695
    -0.4469      3.7636   -40.5767
     0.0001      0.0054     1.0000
w =
     0.0000   210.0000   210.0000          0
     0.0000    -0.0000   297.0000   297.0000
     1.0000     1.0000     1.0000     1.0000
```

d. Warp the image

```
      >>T = maketform('projective', U');
      >> P6_new = imtransform(P6, T, 'XData', [0 210], 'YData', [0 297]);
```

e. Display the image.

```
      >>figure();
      >>imshow(P2);
```



Yes, this is what I expected, as the book in the picture is transformed into frontage. The bottom part has been transformed clearly but the top part is a little bit blurred. This is because that in the original picture, the top part is smaller and it needs to be rescaled to get the result, which will cause lower quality of that part of picture.