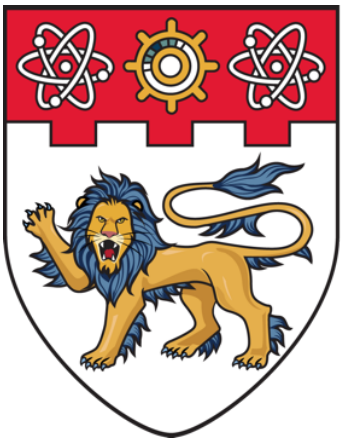


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4003 Lab 1

Report

Hans Tananda U1720251K

Experiments

2.1 Contrast Stretching

a. Load Original image

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

Output:

```
 Name         Size                Bytes  Class    Attributes

  Pc         320x443x3            425280  uint8
```

Change Image to grayscale

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

Output:

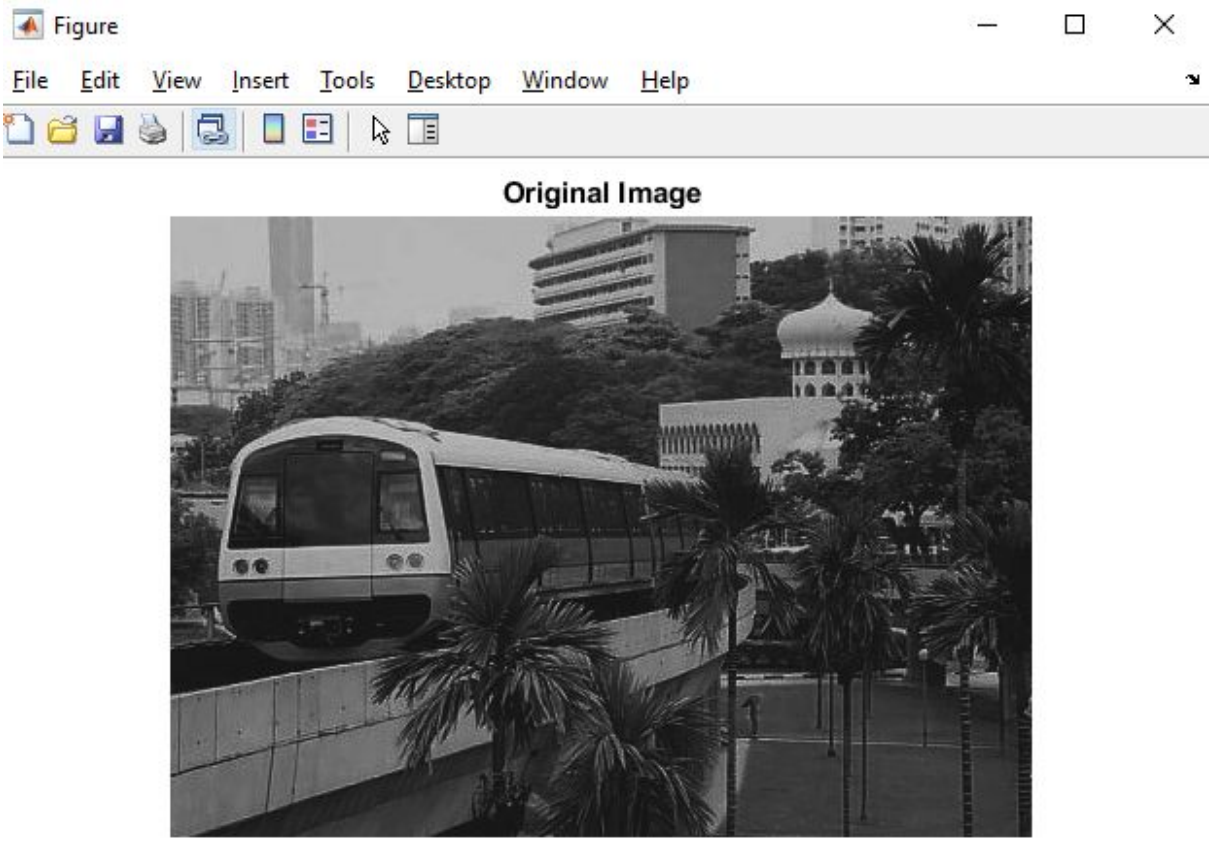
```
 Name         Size              Bytes  Class    Attributes

  P          320x443            141760  uint8
```

This step confirms that the image has been changed to grayscale.

b. Show the image

```
figure
imshow(P);
title('Original Image')
```

Result:



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

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

Output:

```
ans =

  uint8

   13


ans =

  uint8

   204
```

d. *two lines* of MATLAB code to do contrast stretching

```
P2 = imsubtract(P,13);
P2 = immultiply(P2, 255/(204-13));
```

Code Explanation:

To do the contrast stretching, we subtract every single pixel to the minimum pixel value, so that the lowest pixel value for the image is 0. Then, we rescale the pixel values by multiplying it by 255(max pixel value) and dividing it by the current scale (max pixel value-min pixel value = 204-13).

Check the resulting transformed image class (make sure it's still `uint8`).

```
% check image class
whos P2
```

Output:

```
 Name        Size              Bytes  Class    Attributes

  P2        320x443            141760  uint8
```

Check the final image P2 has the correct minimum and maximum intensities of 0 and 255

```
% check min and max of P2
min(P2(:)), max(P2(:))
```

Output:

```
ans =

  uint8

   0


ans =

  uint8
```

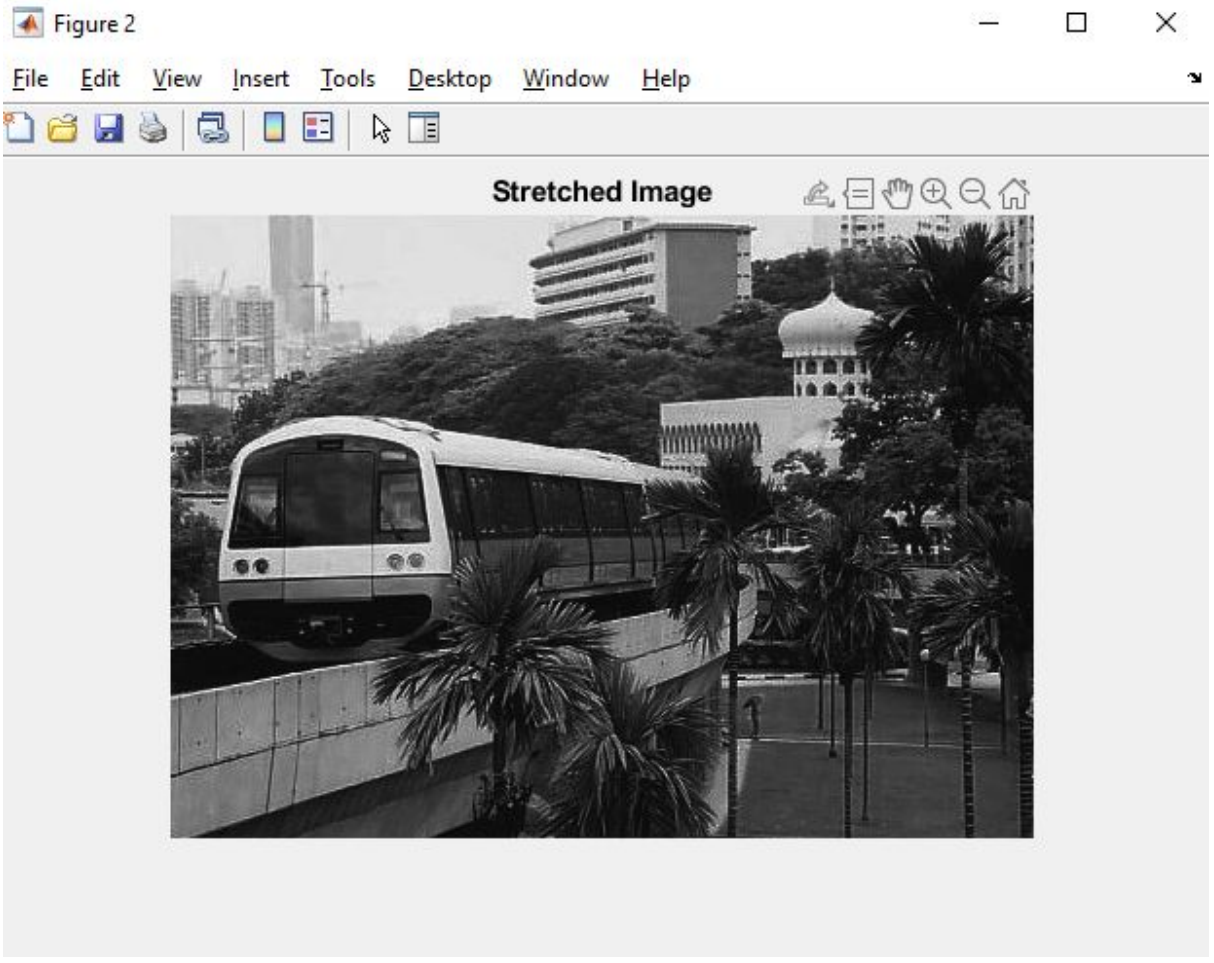
```
255
```

It is shown from the output above that the final image P2 has the correct minimum and maximum intensities of 0 and 255 respectively.

e. display stretched image

```
figure
imshow(P2)
title('Stretched Image')
```

Result:



2.2 Histogram Equalization

Reload Image from 2.1

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

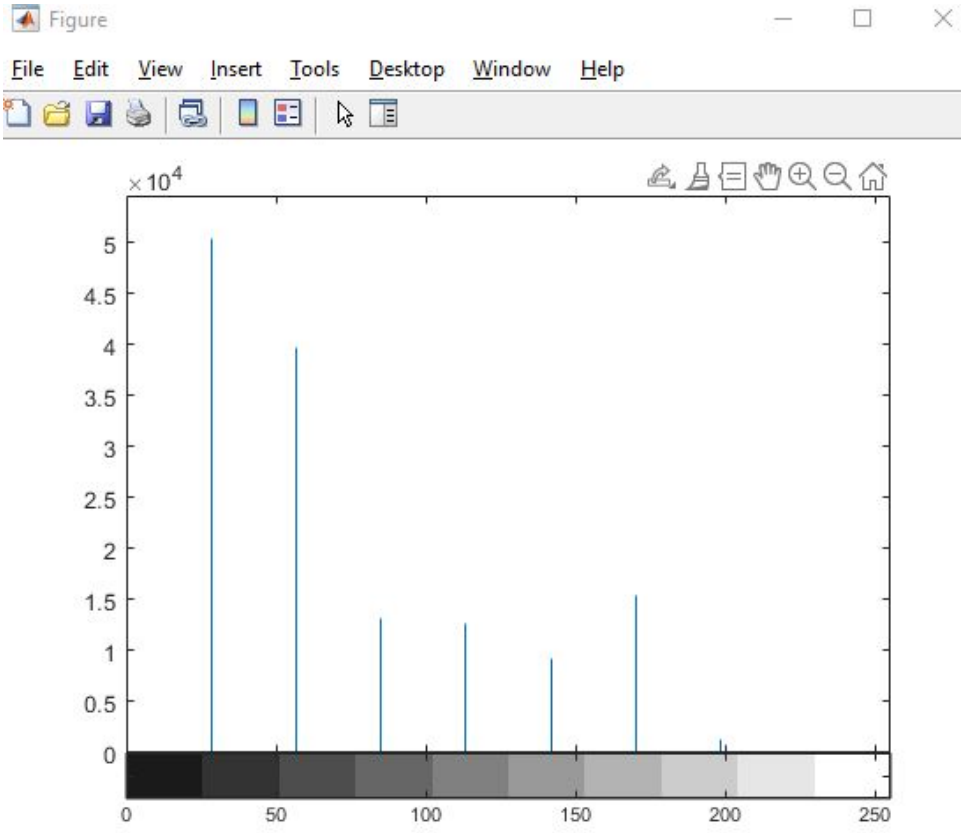
Output:

Name	Size	Bytes	Class	Attributes
P	320x443	141760	uint8	

a. Display the image intensity histogram of P using 10 bins

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

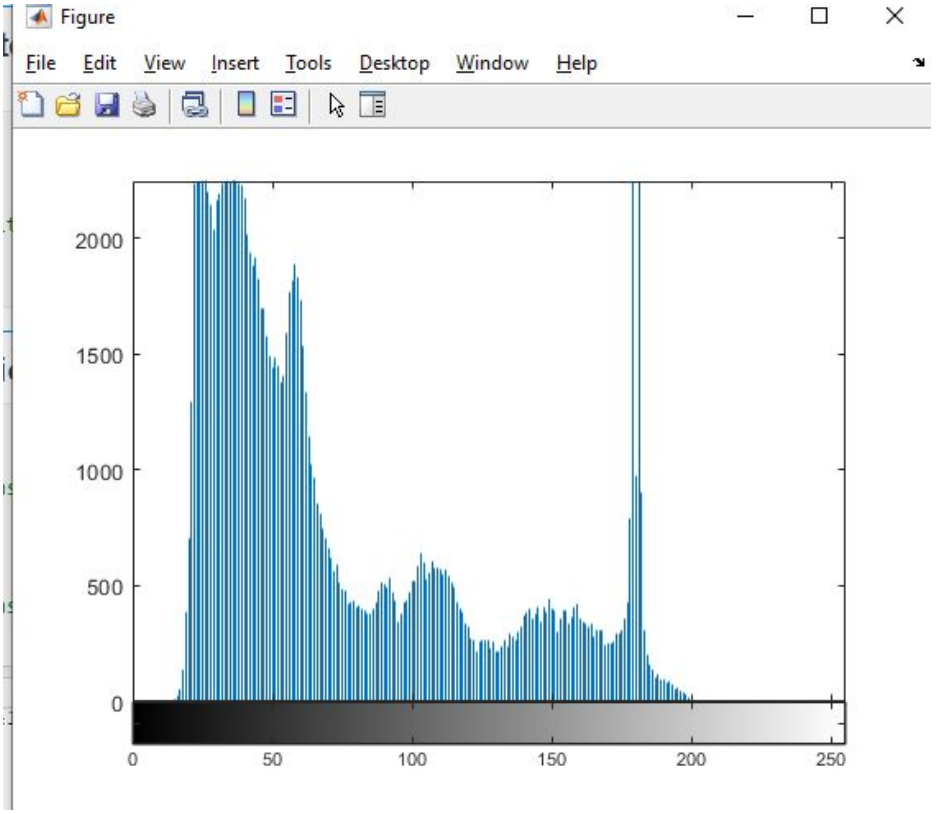
Result:



Display the image intensity histogram of P using 256 bins

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

Result:



The differences between histogram with 10 bins and 256 bins:

From the resulting histograms, it can be seen that the second histogram is much more detailed (has more 'curves') than the first bin. This happens since the second histogram has a lot more bins (25x) compared to the first histogram. Moreover, even though the overall histogram is rather similar, it can also be observed that the 2nd histogram displayed several peaks (>2000 pixels) around intensity of 180, whereas the 1st histogram only showed a slight increase in number of pixels around intensity of 170.

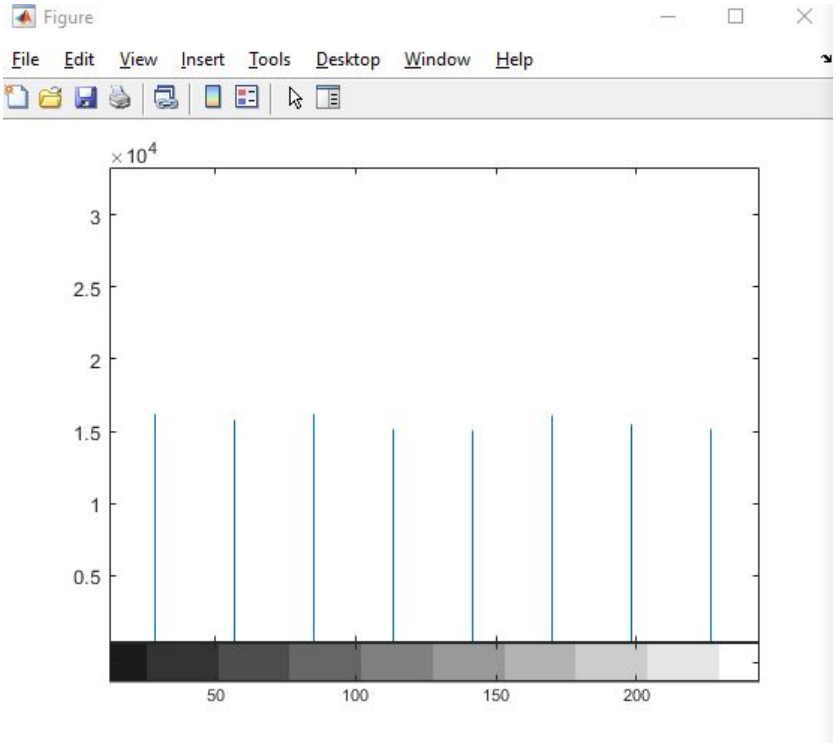
b. Histogram Equalization

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

Redisplay the image intensity histogram of P using 10 bins

```
figure
imhist(P3,10);
```

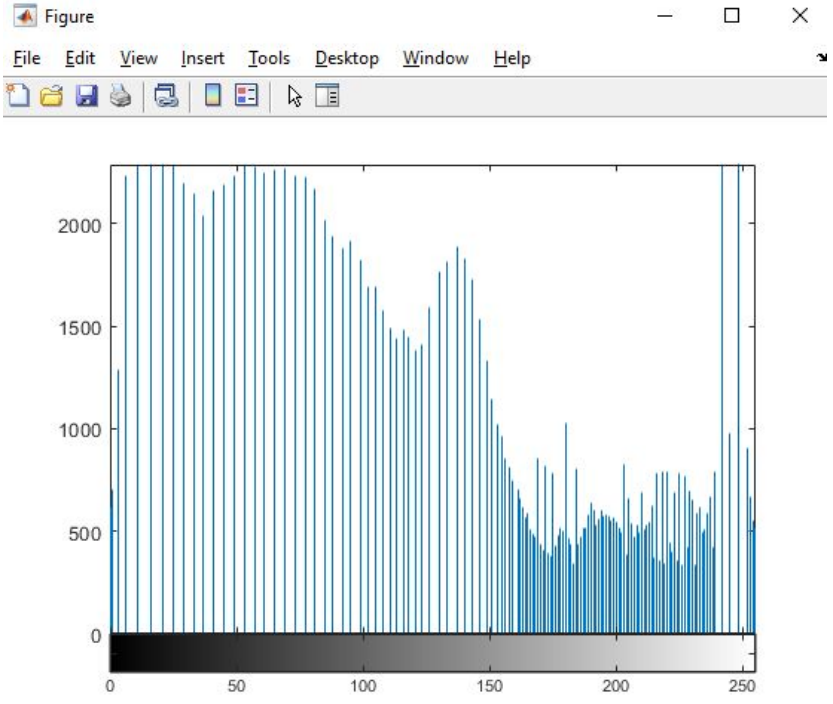
Result:



Redisplay the image intensity histogram of P using 256 bins

```
figure
imhist(P3,256);
```

Result:



Explanation

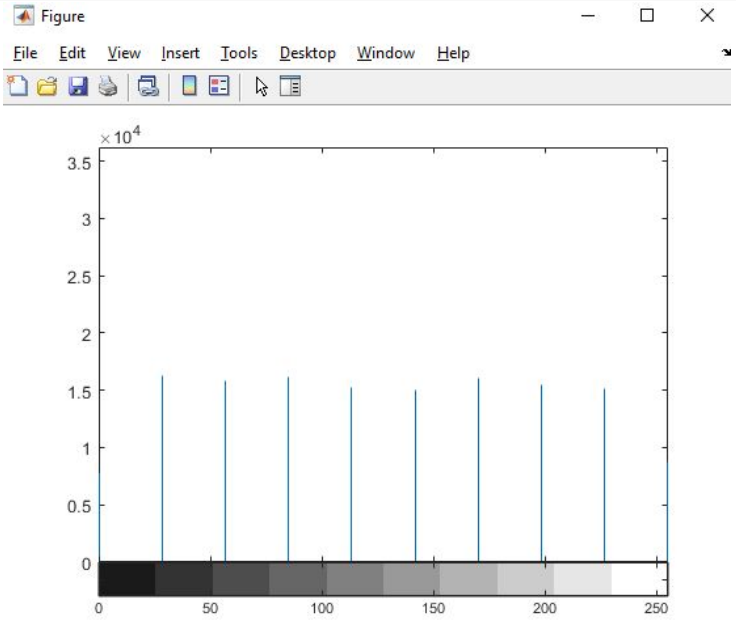
It can be seen that the two histograms are more equalized than those seen in part (a.). After the equalization, the histograms are more leveled and some bins that were previously empty are getting pixels from the other bins. However, it can be seen that in the histogram with 10 bins, each bin has relatively the same number of pixels (around 15000) except the first and the last bin (around 7000-8000 pixels). Meanwhile, the histogram with 256 bins still varies quite significantly (mostly around 500-2000). In addition, there are still quite a lot of bins in the second histogram that is empty. Meanwhile, all of the 10 bins in the first histogram are filled.

c. Rerun the histogram equalization on P3

```
P3_1 = histeq(P3,255);
```

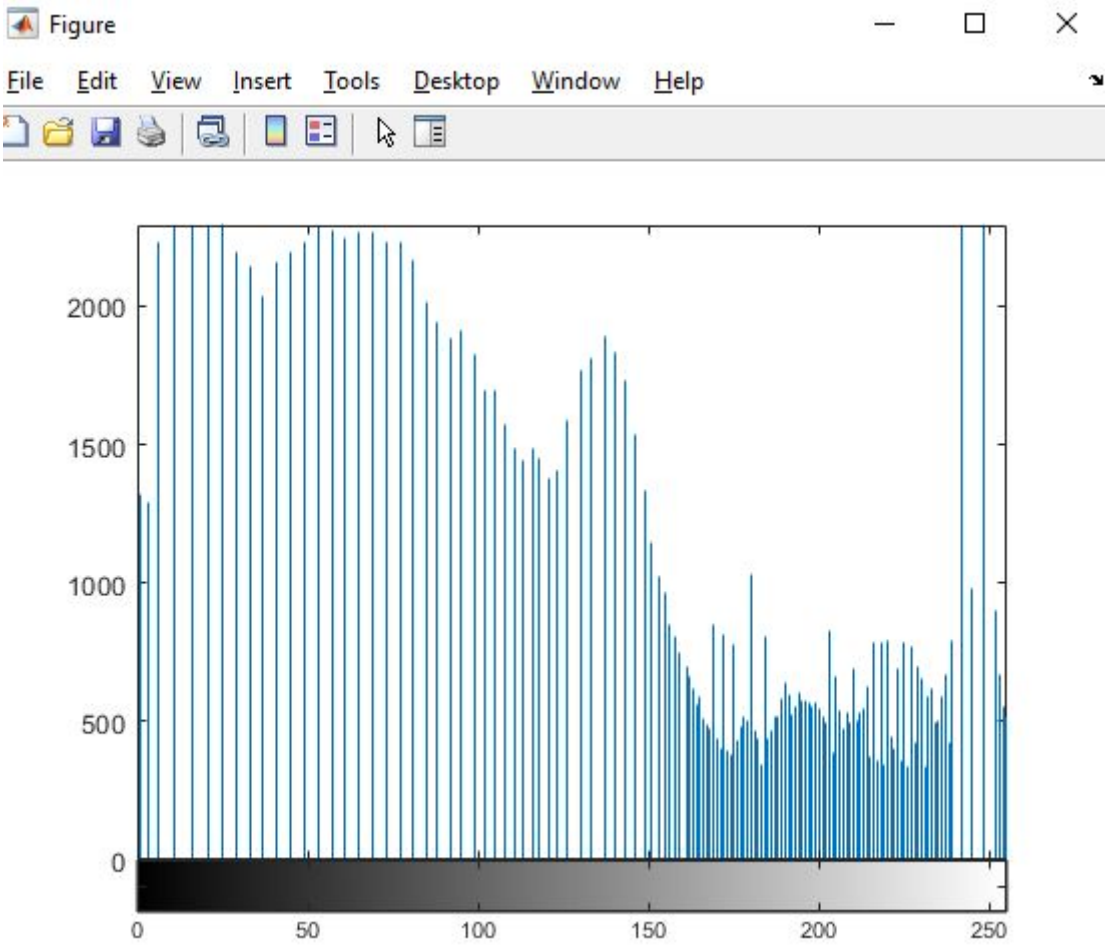
Redisplay the image intensity histogram of P using 10 bins

```
figure
imhist(P3_1,10);
```



Redisplay the image intensity histogram of P using 256 bins

```
figure
imhist(P3_1,256);
```



Explanation:

It can be observed that the histogram did not differ. To confirm whether it is exactly the same, I decided to test using the following code:

```
sum(sum(P3-P3_1))
```

Output:

```
ans = 0
```

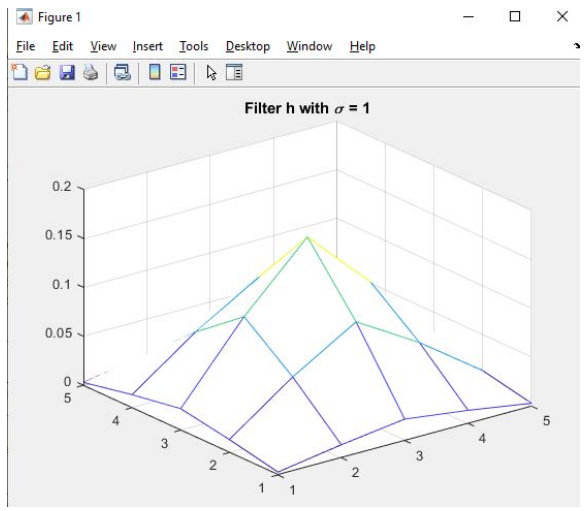
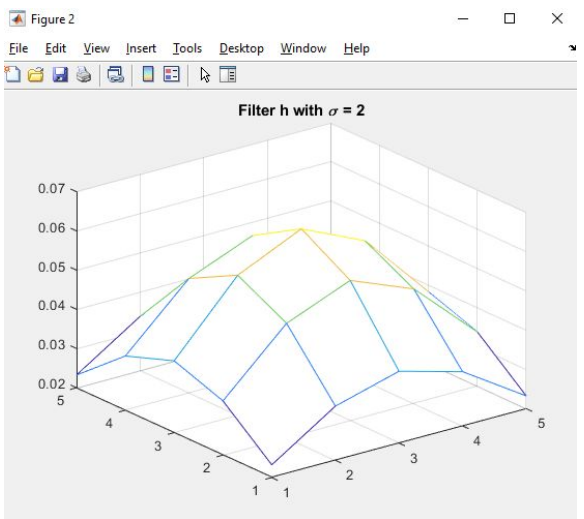
Therefore, this confirms that reapplying the algorithm produces the exact same image. It seems that this is because of the fact that the pixels are already mapped to the “designated” bins based on the cumulative distribution function(cdf) from the original image. Since the distribution does not change during the process (partly due to the fact that we can only merge and not split the bins from the histogram), we obtained the same mapping and thus the same image.

2.3 Linear Spatial Filtering

a. Generate the gaussian filters and view the filters as 3D graphs by using the mesh function

```
% Source:
https://www.mathworks.com/matlabcentral/answers/430031-how-to-i-apply-ga
ussian-filter-on-images-in-matlab
% First filter
filter_h1 = fspecial('gaussian', [5 5], 1);
% Display first filter as 3D graphs using mesh function
figure;
mesh(filter_h1);
title('Filter h with \sigma = 1');
% Second filter
filter_h2 = fspecial('gaussian', [5 5], 2);
% Display second filter as 3D graphs using mesh function
figure;
mesh(filter_h2);
title('Filter h with \sigma = 2');
```

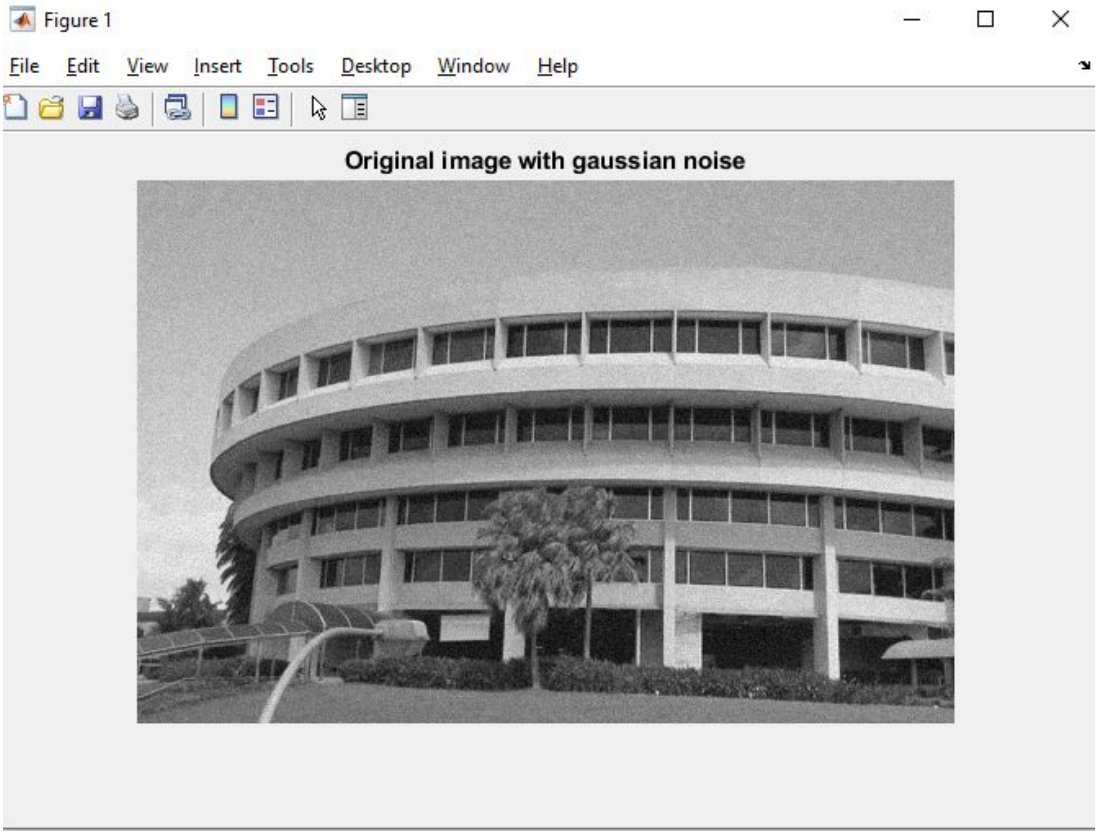
Result:

Gaussian filter with Y and X-dimensions are 5 and σ = 1.0	Gaussian filter with Y and X-dimensions are 5 and σ = 2.0
	

b. Load and display the image with gaussian noise

```
P_ntu_gn = imread('ntu-gn.jpg');
figure;
imshow(P_ntu_gn);
title('Original image with gaussian noise');
```

Result:

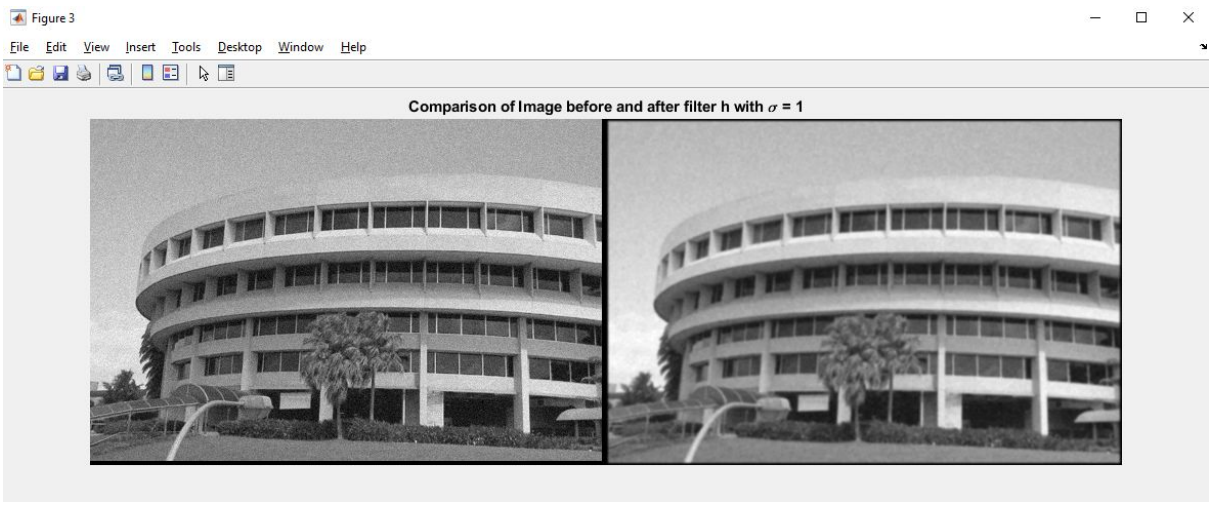


c. Filter the image using the conv2 function, and display the resultant images

Filter and display image after first filter is applied to the image

```
P_ntu_gn_after_filter_h1 = conv2(P_ntu_gn,filter_h1);
figure;
imshow(uint8(P_ntu_gn_after_filter_h1));
title('Image after filter h with \sigma = 1');
figure;
% Based on
https://www.mathworks.com/matlabcentral/answers/161457-how-can-i-display
-two-images-at-once
imshowpair(P_ntu_gn, uint8(P_ntu_gn_after_filter_h1), 'montage')
title('Comparison of Image before and after filter h with \sigma = 1');
```

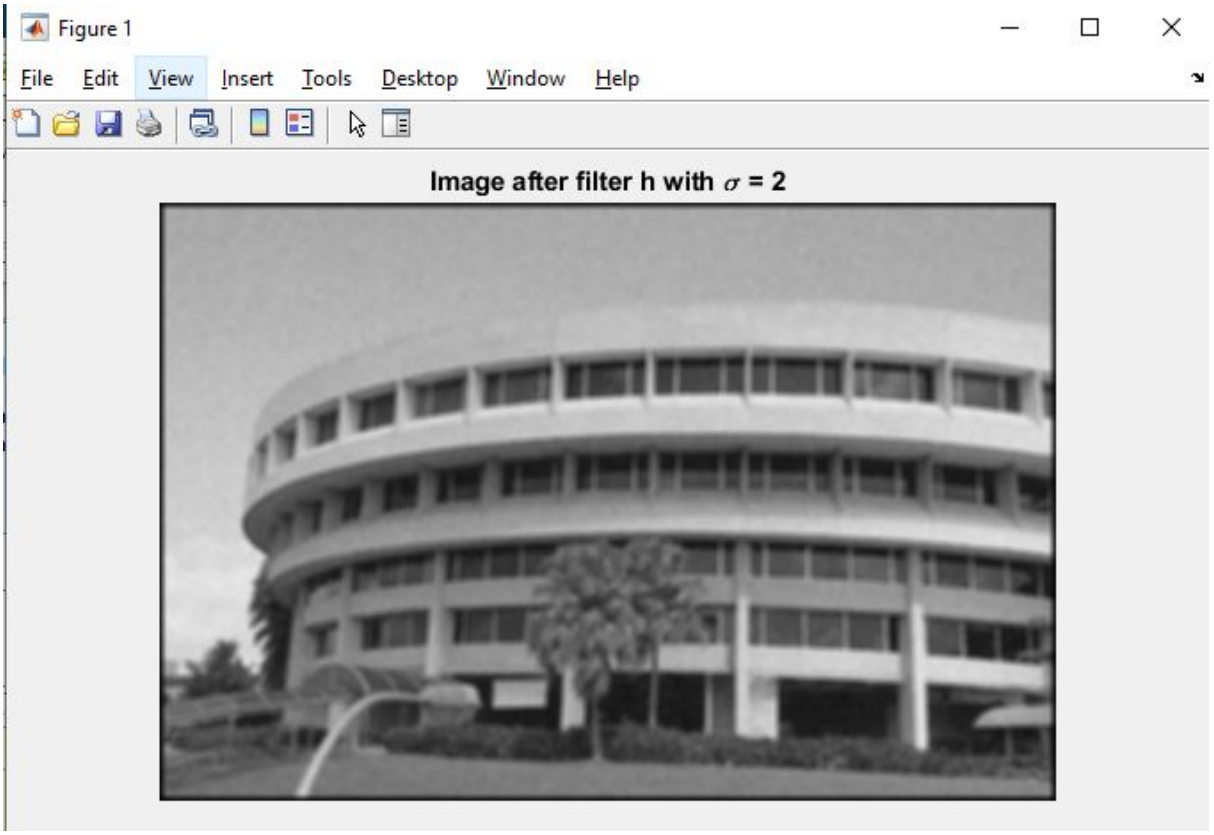
Result:

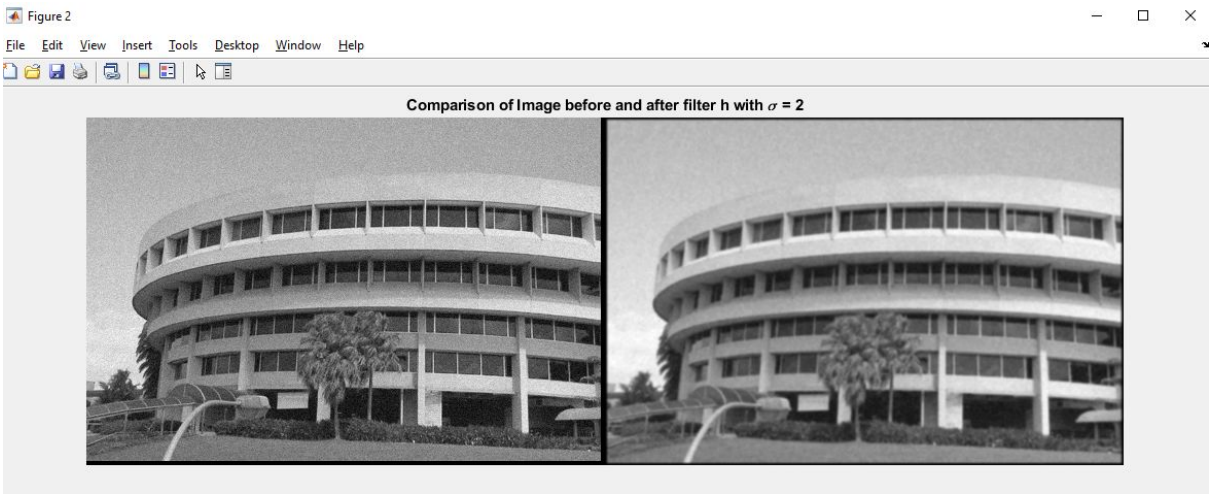


Filter and display image after second filter is applied to the image

```
P_ntu_gn_after_filter_h2 = conv2(P_ntu_gn,filter_h2);
figure;
imshow(uint8(P_ntu_gn_after_filter_h2));
title('Image after filter h with \sigma = 2');
figure;
imshowpair(P_ntu_gn, uint8(P_ntu_gn_after_filter_h1), 'montage')
title('Comparison of Image before and after filter h with \sigma = 2');
```

Result:



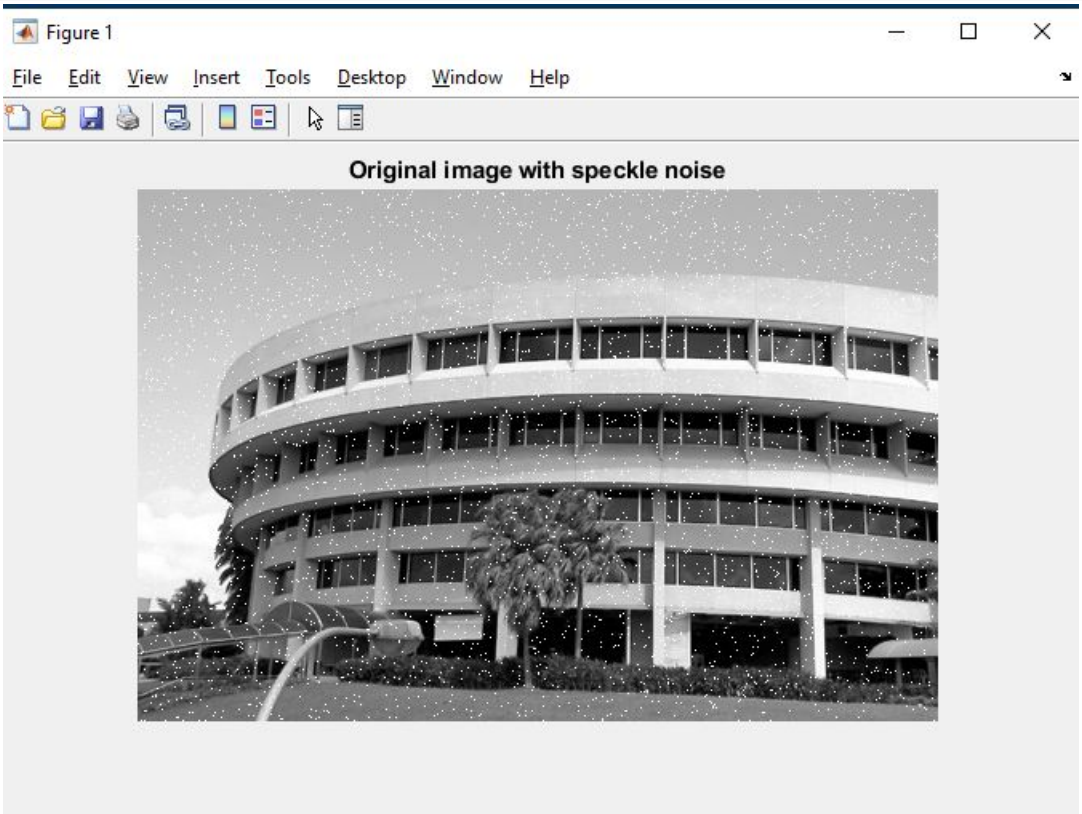


Explanation:

From the result above, it can be observed that the noise in the images has been suppressed. After taking a closer look, it can be seen that the noise in the second image, which has been filtered with a gaussian filter with σ=2, is far less visible than the first image which has been filtered with a gaussian filter with σ=1. However, it can be seen that the second image is more blurred than the first one. Hence, it can be said that the second filter is more effective than the fist one in removing noise. This comes with a tradeoff of reducing the sharpness of the image.

d. Load and display the image with speckle noise

```
P_ntu_sp = imread('ntu-sp.jpg');
figure;
imshow(P_ntu_sp);
title('Original image with speckle noise');
```

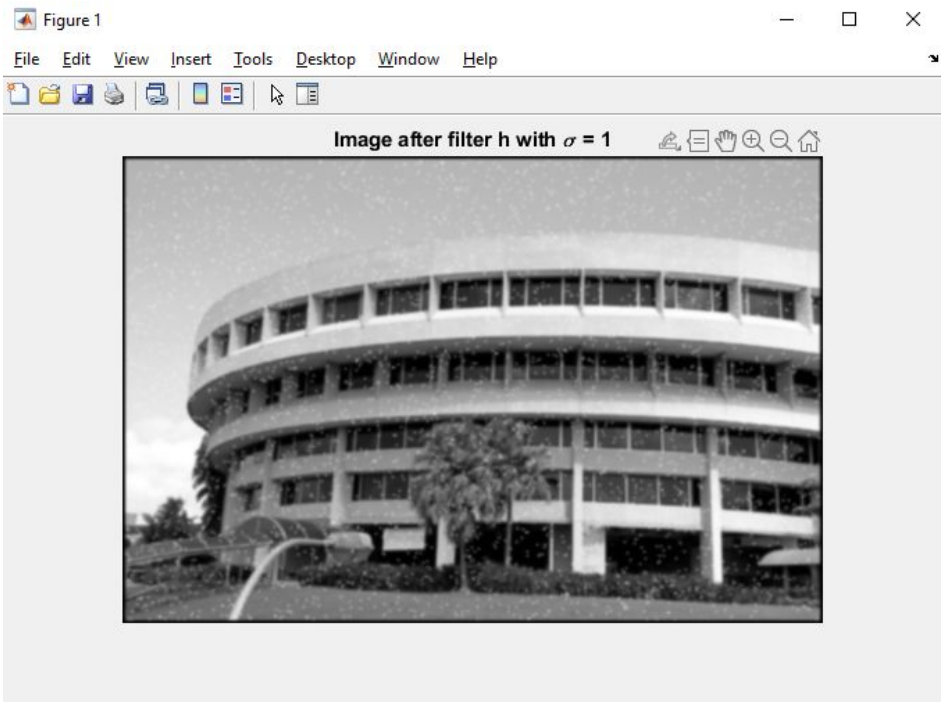


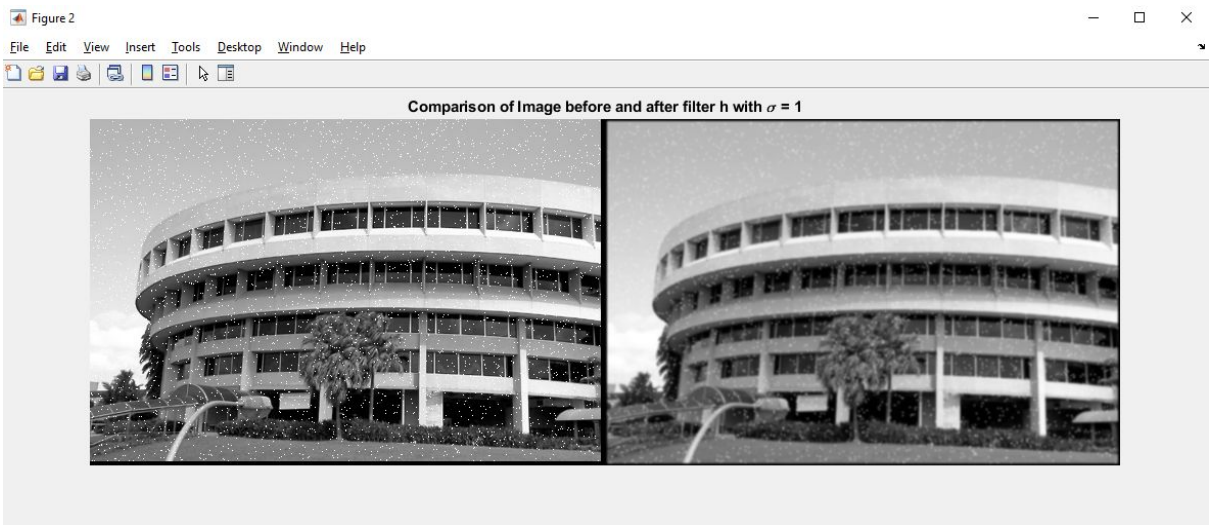
e. Repeat step (c) to the image with speckle noise

Filter and display image after first filter is applied to the image

```
P_ntu_sp_after_filter_h1 = conv2(P_ntu_sp,filter_h1);
figure;
imshow(uint8(P_ntu_sp_after_filter_h1));
title('Image after filter h with \sigma = 1');
figure;
imshowpair(P_ntu_sp, uint8(P_ntu_sp_after_filter_h1), 'montage')
title('Comparison of Image before and after filter h with \sigma = 1');
```

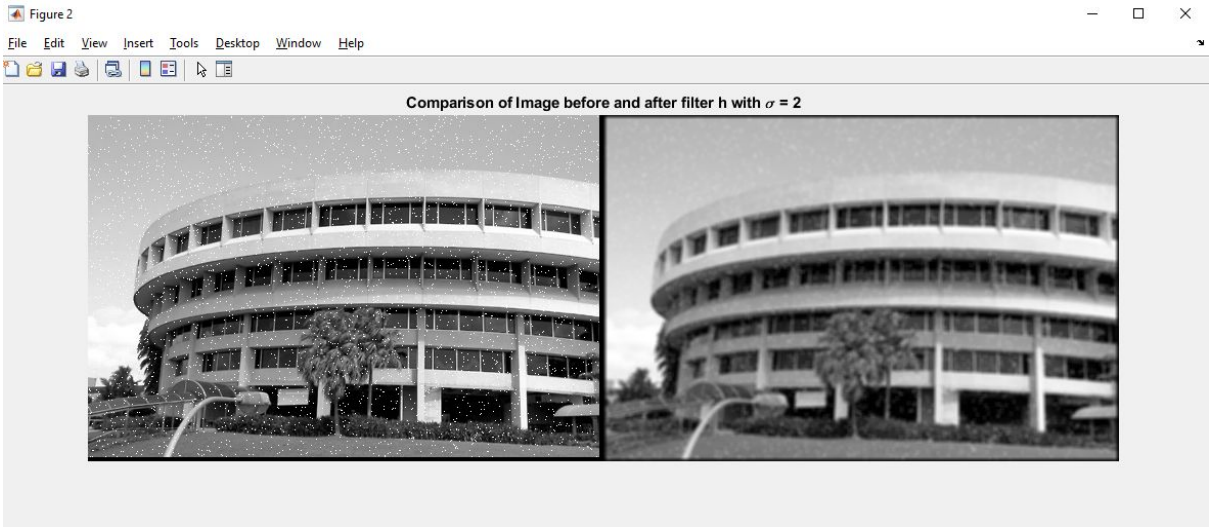
Result:





Filter and display image after second filter is applied to the image

```
P_ntu_sp_after_filter_h2 = conv2(P_ntu_sp,filter_h2);
figure;
imshow(uint8(P_ntu_sp_after_filter_h2));
title('Image after filter h with \sigma = 2');
figure;
imshowpair(P_ntu_sp, uint8(P_ntu_sp_after_filter_h2), 'montage')
title('Comparison of Image before and after filter h with \sigma = 2');
```



Explanation

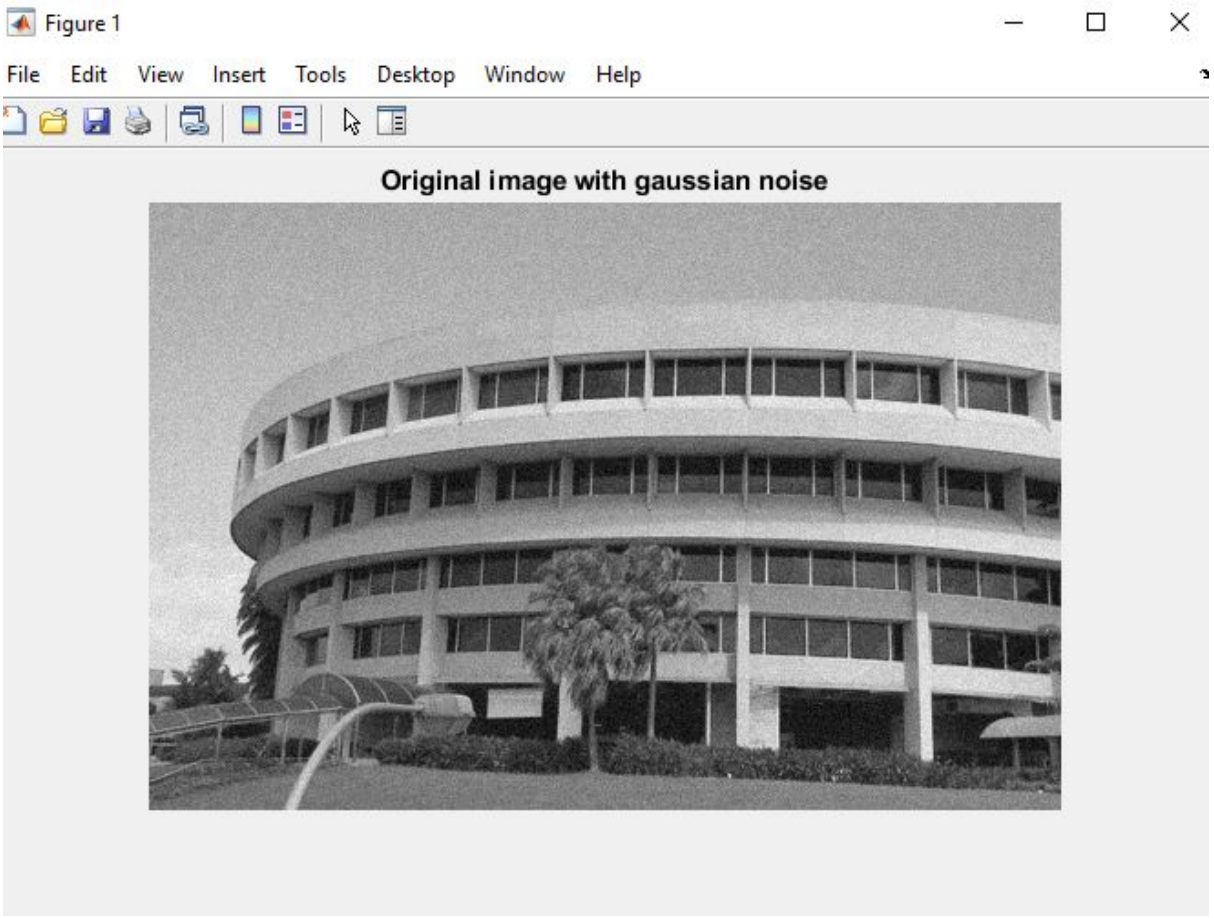
It can be observed from the two results above that the Gaussian averaging filter is unable to handle the speckle noise well. The noise in the image is less obvious, but it is still quite prominent. This may be caused by the fact that speckle noise are “outliers”, that stand out among the neighbouring pixels. Hence, when the gaussian filters are applied, the noise will be spread out to the neighbouring pixels. Thus, the noise will be spread out but is still there. It can be concluded that the Gaussian averaging filter, as its name suggests, performs better in correcting gaussian noise than speckle noise.

2.3 Median Filtering

a. Load and display the image with gaussian noise

```
P_ntu_gn = imread('ntu-gn.jpg');
figure;
imshow(P_ntu_gn);
title('Original image with gaussian noise');
```

Result:



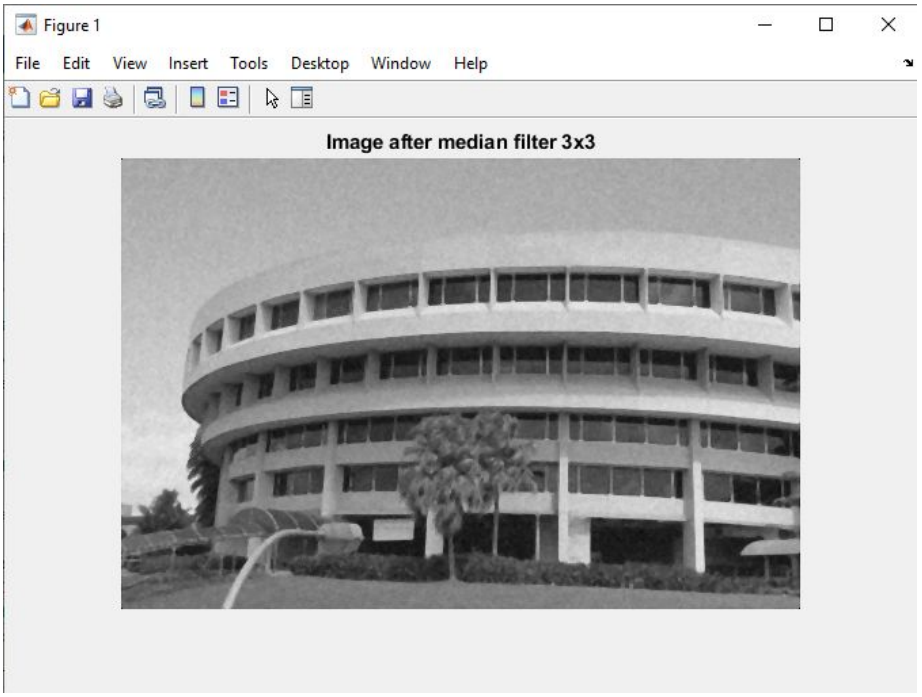
b. Filter the image using the conv2 function, and display the resultant images

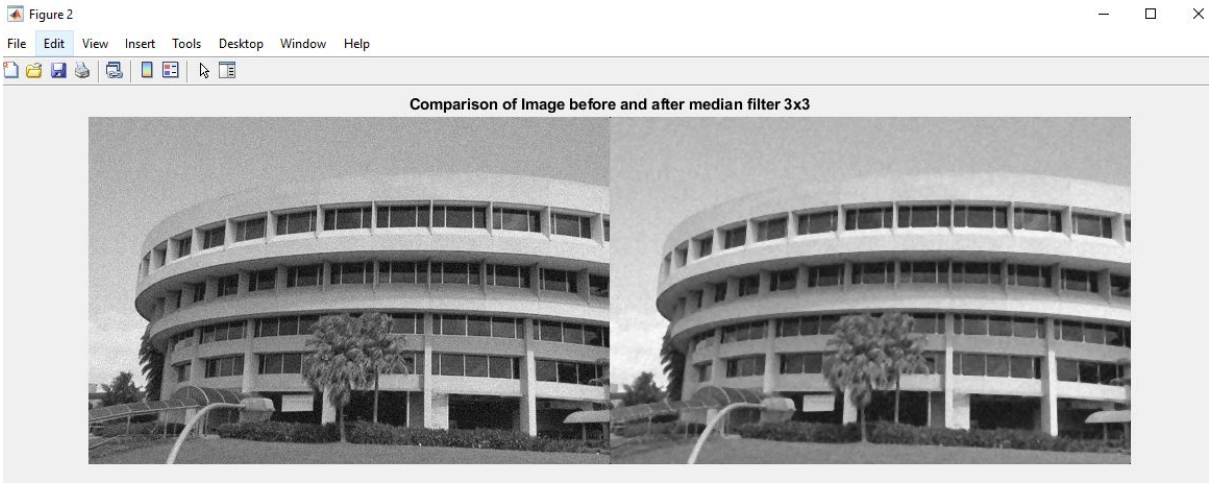
Filter and display image after first filter is applied to the image

```
P_ntu_gn_after_med3 = medfilt2(P_ntu_gn, [3 3]);
figure;
imshow(uint8(P_ntu_gn_after_med3));
title('Image after median filter 3x3');
figure;
```

```
imshowpair(P_ntu_gn, uint8(P_ntu_gn_after_med3), 'montage')
title('Comparison of Image before and after median filter 3x3');
```

Result:

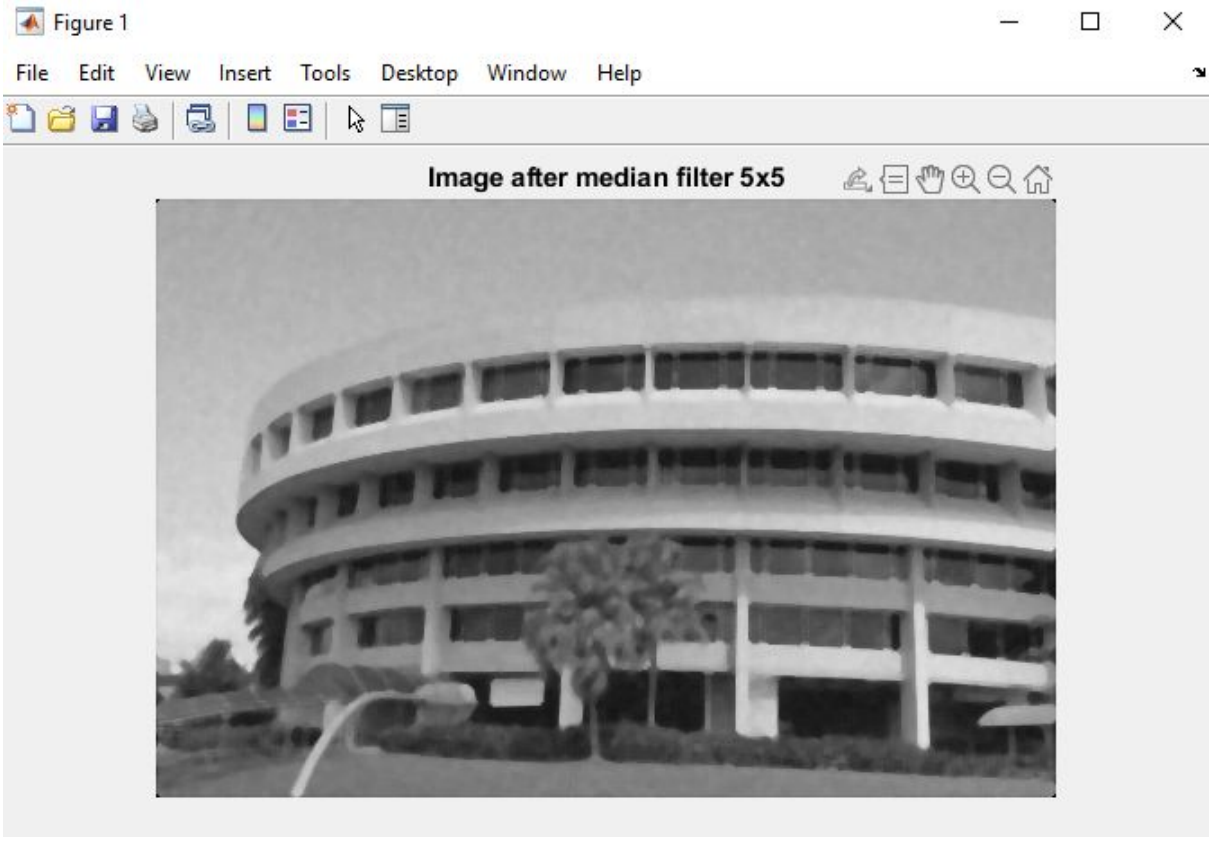


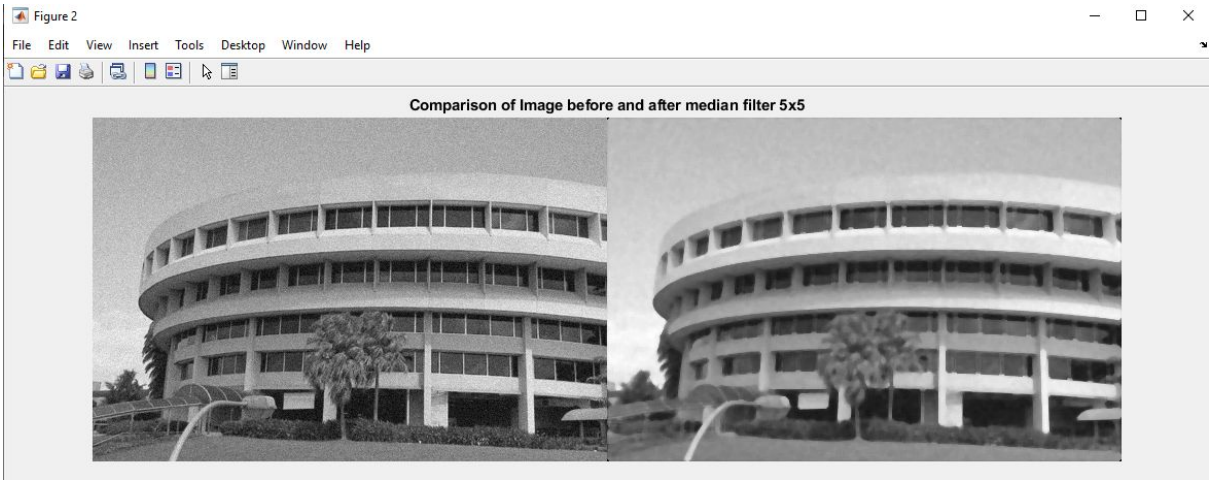


Filter and display image after second filter is applied to the image

```
P_ntu_gn_after_filter_h2 = conv2(P_ntu_gn,filter_h2);
figure;
imshow(uint8(P_ntu_gn_after_filter_h2));
title('Image after filter h with \sigma = 2');
figure;
imshowpair(P_ntu_gn, uint8(P_ntu_gn_after_filter_h1), 'montage')
title('Comparison of Image before and after filter h with \sigma = 2');
```

Result:

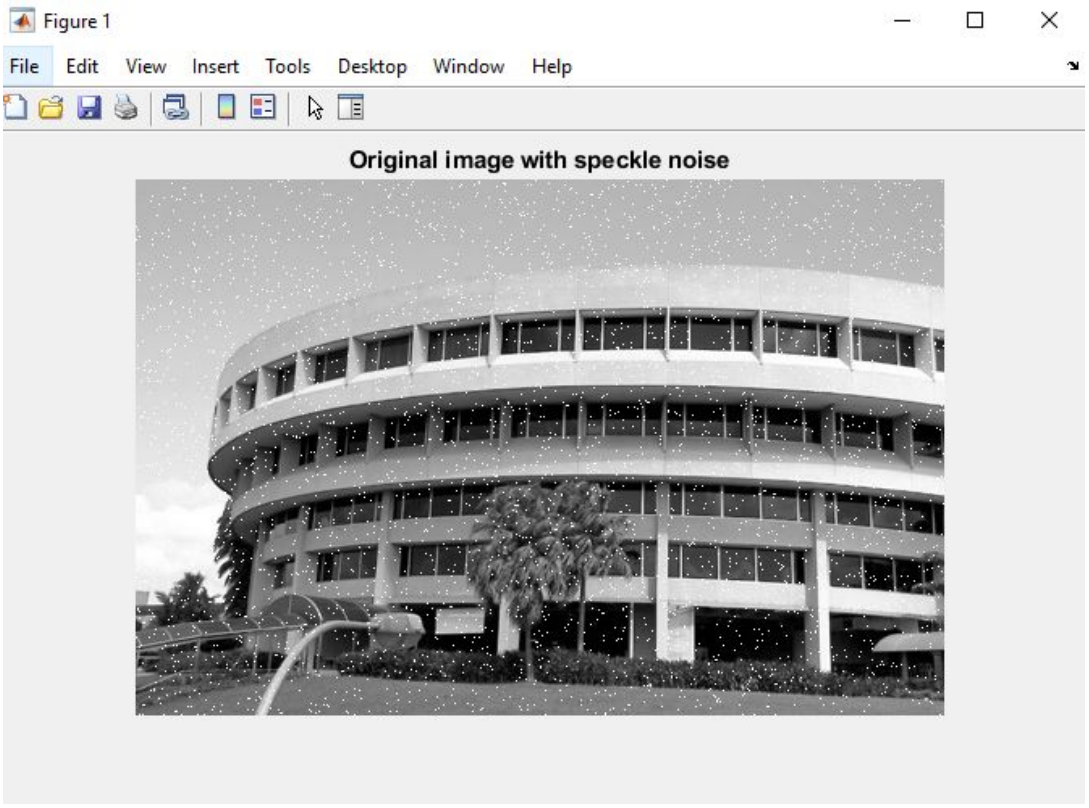




c. Load and display the image with speckle noise

```
P_ntu_sp = imread('ntu-sp.jpg');
figure;
imshow(P_ntu_sp);
title('Original image with speckle noise');
```

Result:

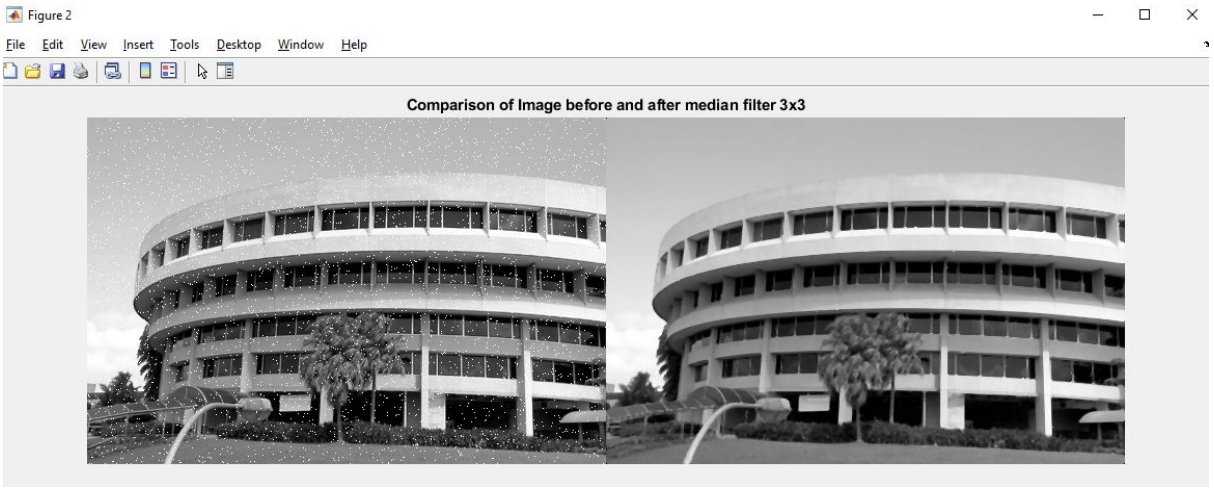


d. Repeat step (b) in the image with speckle noise

Filter and display image after first filter is applied to the image

```
P_ntu_sp_after_med3 = medfilt2(P_ntu_sp, [3 3]);
figure;
imshow(uint8(P_ntu_sp_after_med3));
title('Image after median filter 3x3');
figure;
imshowpair(P_ntu_sp, uint8(P_ntu_sp_after_med3), 'montage')
title('Comparison of Image before and after median filter 3x3');
```

Result:



Filter and display image after second filter is applied to the image

```
P_ntu_sp_after_med5 = medfilt2(P_ntu_sp, [5 5]);
figure;
imshow(uint8(P_ntu_sp_after_med5));
title('Image after median filter 5x5');
figure;
```

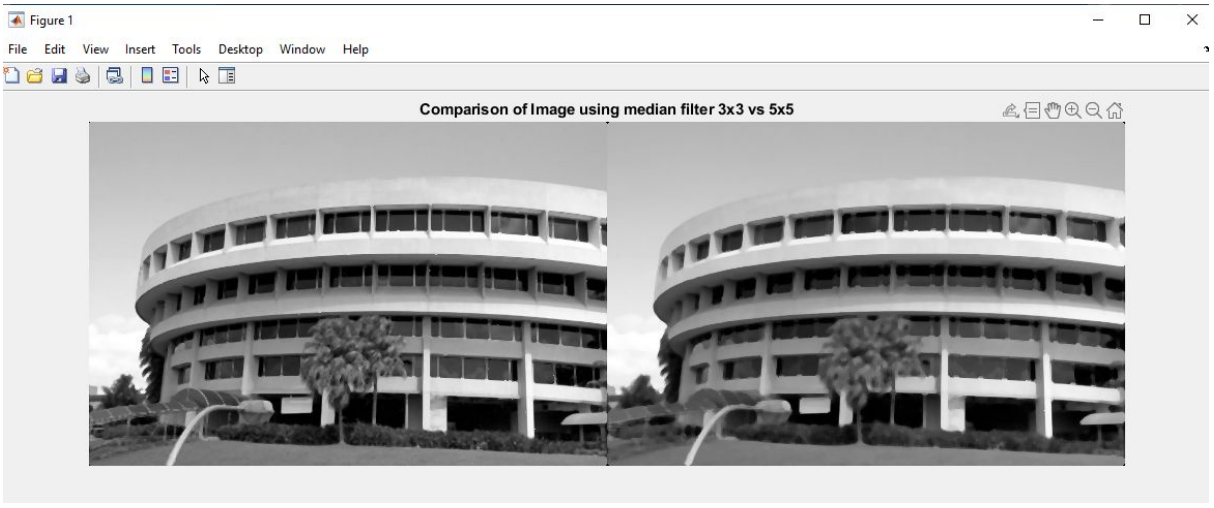
```
imshowpair(P_ntu_sp, uint8(P_ntu_sp_after_med5), 'montage')
title('Comparison of Image before and after median filter 5x5');
```

Result:

Comparison of median filters

```
figure;
imshowpair(uint8(P_ntu_sp_after_med3), uint8(P_ntu_sp_after_med5),
'montage')
title('Comparison of Image using median filter 3x3 vs 5x5');
```

Result:



Explanation:

It can be observed that the median filter is much more effective in handling speckle noise. By only using the median filter of size 3x3, the speckle noise is barely visible. Only some speckle noise near the white building can be seen. By increasing the kernel size to 5x5, it makes the filtering more effective. However, more details in the image are lost and thus making the edges in the image becomes blurry.
The median filter is still able to handle the gaussian noise to some extent. However, we can see that the gaussian averaging filter handles gaussian noise better than the median filter. We can also see that increasing the kernel size also makes the edges more blurred.

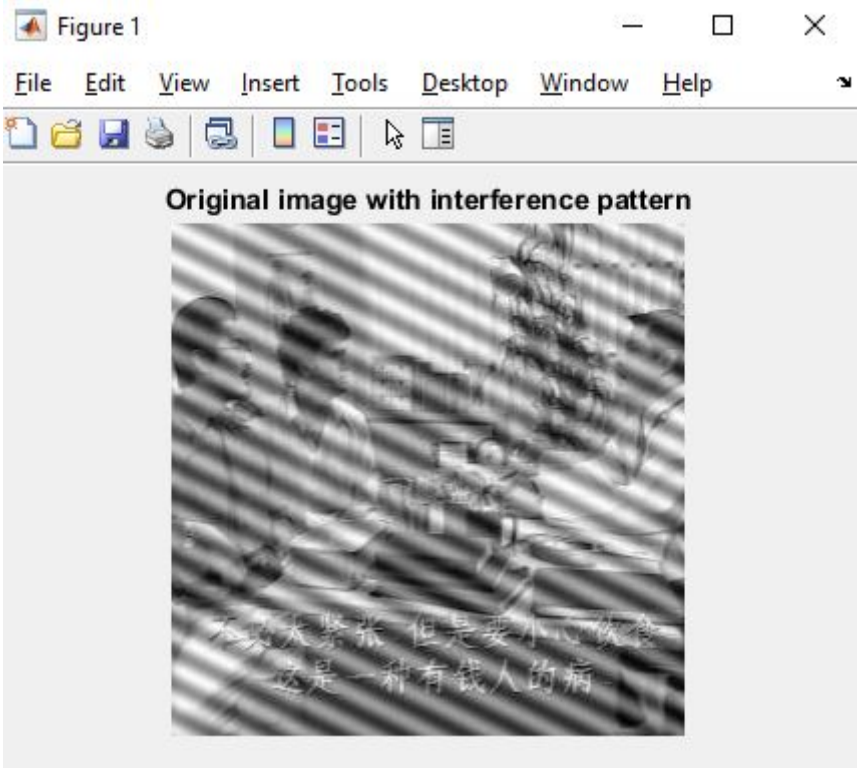
In conclusion, a gaussian averaging filter is more suitable to handle gaussian noise, and a median kernel is more suitable to handle speckle noise. Increasing the sigma value for the gaussian filter removes more noise yet it also makes the image more blurred. Similarly, increasing the kernel size for median filter removes more speckle noise yet it also makes the edges in the image more blurred.

2.5 Suppressing Noise Interference Patterns

a. Load and display the image

```
P_pck = imread('pck-int.jpg');
figure;
imshow(P_pck);
title('Original image with interference pattern');
```

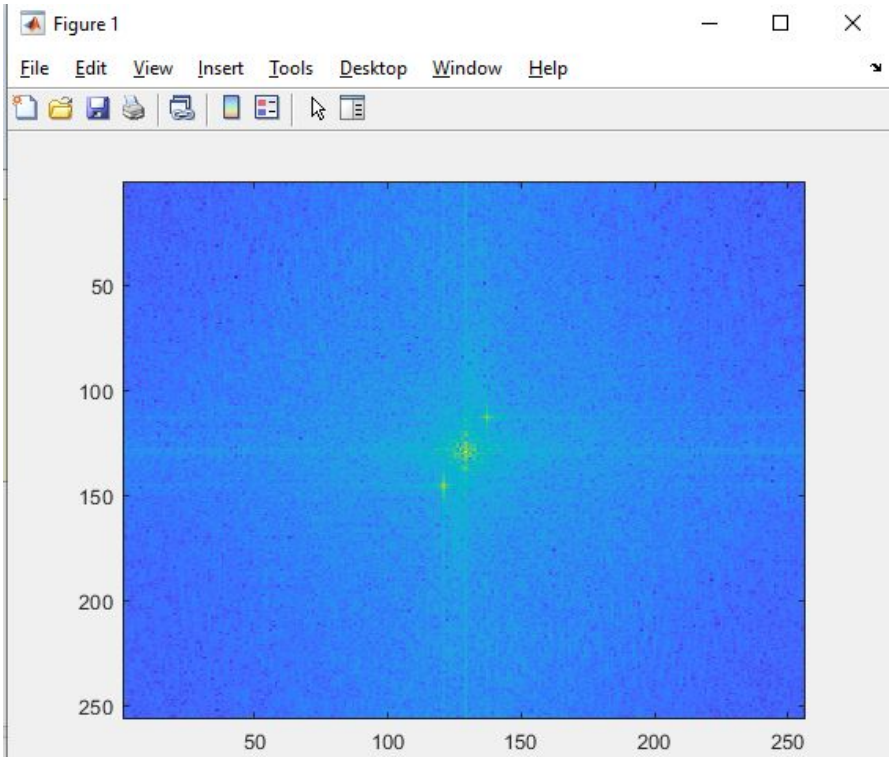
Result:



b. Fourier transform F of the image

```
F = fft2(P);

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

Result:

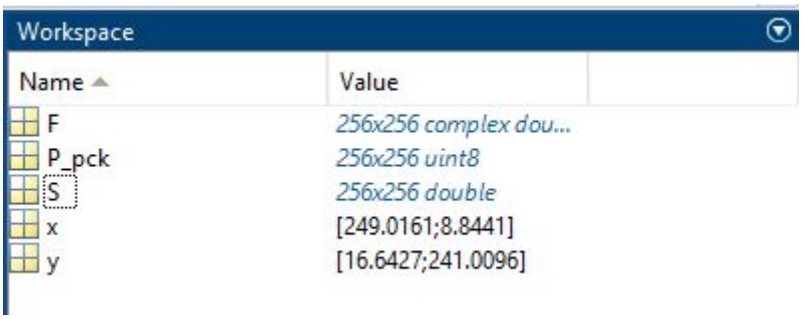


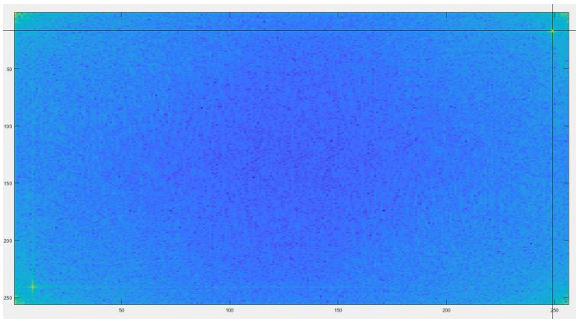
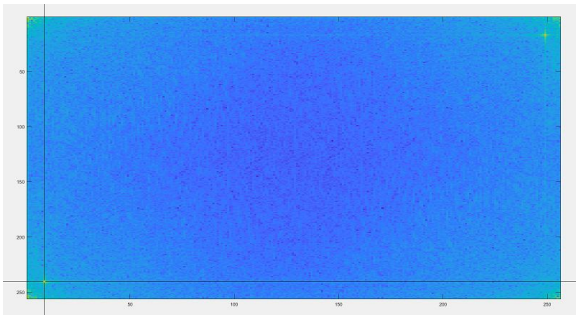
c. Redisplay the power spectrum without fftshift and measure actual location of the peaks

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

% Locate coordinate (x,y) using ginput() function
% https://www.mathworks.com/help/matlab/ref/ginput.html
[x,y] = ginput;
```

Result & Explanation:

After viewing the power spectrum, it can be confirmed that there are two peaks. Below are the coordinates obtained from the ginput:



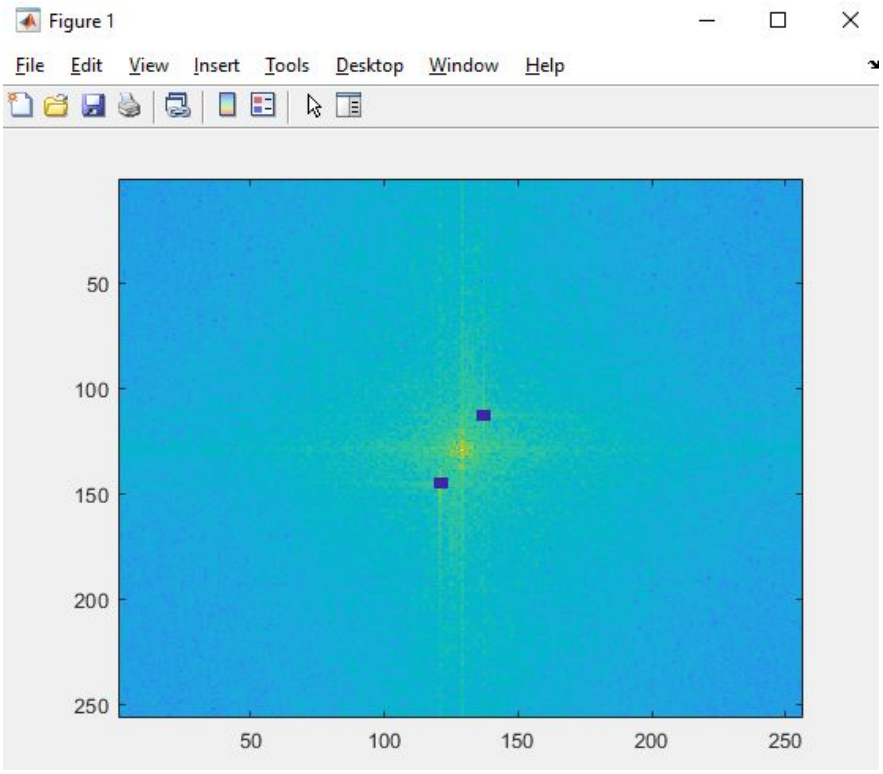
[249.0161, 16.6427]	[8.8441, 241.0096]
	

Since we know that the pixel location must be an integer, I decided to round the above obtained values to the nearest integer. Therefore, we have (249,17) and (9,241)

D. Set the 5x5 neighbourhood elements at the corresponding peaks to zero in the Fourier transform F.

```
xr = round(x);
yr = round(y);
F_1 = F;
%
https://www.mathworks.com/matlabcentral/answers/470450-how-can-i-change-a-particular-range-of-data
for i = 1: length(x)
      F_1(yr(i)-2:yr(i)+2, xr(i)-2:xr(i)+2) =0;
end
% Recompute the power spectrum
S_1= abs(F_1);
% Display it as in step (b)
figure
imagesc(fftshift(S_1.^0.1))
colormap('default')
```

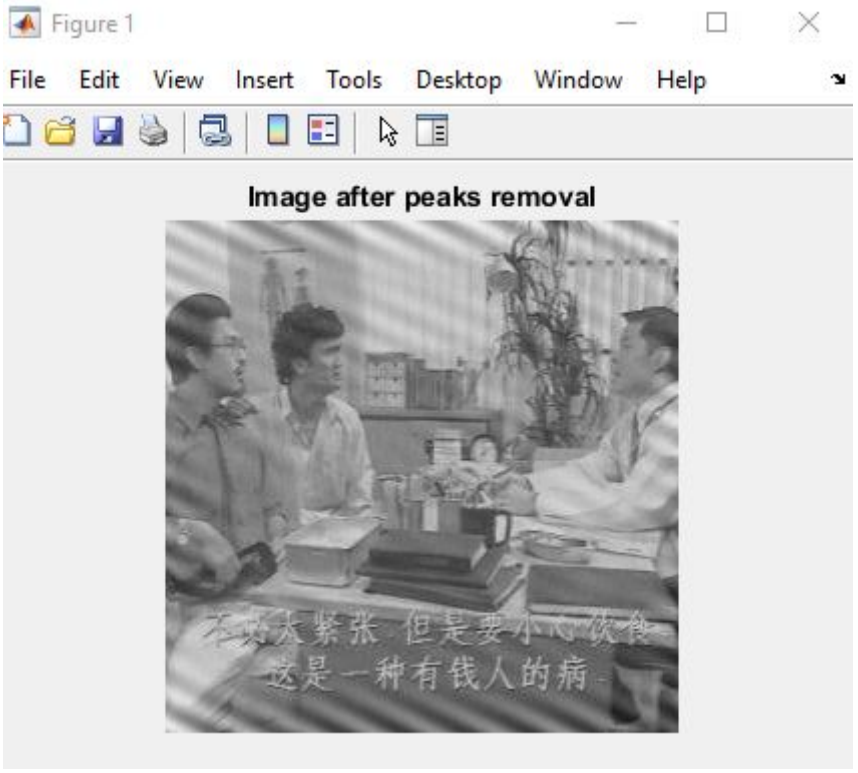
Result:



E. Compute the inverse FT and display the resultant image.

```
P_pck_1 = uint8(ifft2(F_1));
figure
imshow(real(P_pck_1));
title('Image after peaks removal');
figure;
imshowpair(P_pck, real(P_pck_1), 'montage')
title('Comparison of Image before and after peak removal');
```

Result:



Explanation:

It can be seen that most of the interference in the centre of the picture has been removed. However, it seems that the interference in the border of the image is still quite prominent. This is due to the fact that we have tried to remove the “peak” in the FFT that represents the most part of the interference in the image.

Upon further observation on the power spectrum shown in (c.), it can be observed that there are long straight lines along the x-axis and y-axis of the peak coordinates. Therefore, I tried to remove them by changing the value to 0. Interestingly, this reduces the interference quite

significantly. The interference in the border is minimized, only those that are really close to the border that are still quite prominent.

Note: To make the implementation simple, I decided to change the values that have the same x or y coordinates and are at most 50 pixels away from the peak coordinates to 0. I added additional checking with the array bounds (1 and 256) to prevent it from exceeding the array bounds.

Code:

```
%% Improvement
F_2 = F_1;
for i = 1: length(x)
      F_2(max(1,yr(i)-50):min(yr(i)+50,256), xr(i)) =0;
      F_2(yr(i), max(1,xr(i)-50):min(xr(i)+50,256)) =0;
end
% Recompute the power spectrum
S_2= abs(F_2);
% Display it as in step (b)
figure
imagesc(fftshift(S_2.^0.1))
colormap('default')
title('power spectrum after improvement')
%% Compute the inverse FT and display the resultant image.
P_pck_2 = uint8(ifft2(F_2));
figure
imshow(real(P_pck_2));
title('Image after further improvment');
figure;
imshowpair(real(P_pck_1), real(P_pck_2), 'montage')
title('Comparison of Image before and improvement');
```

Result:

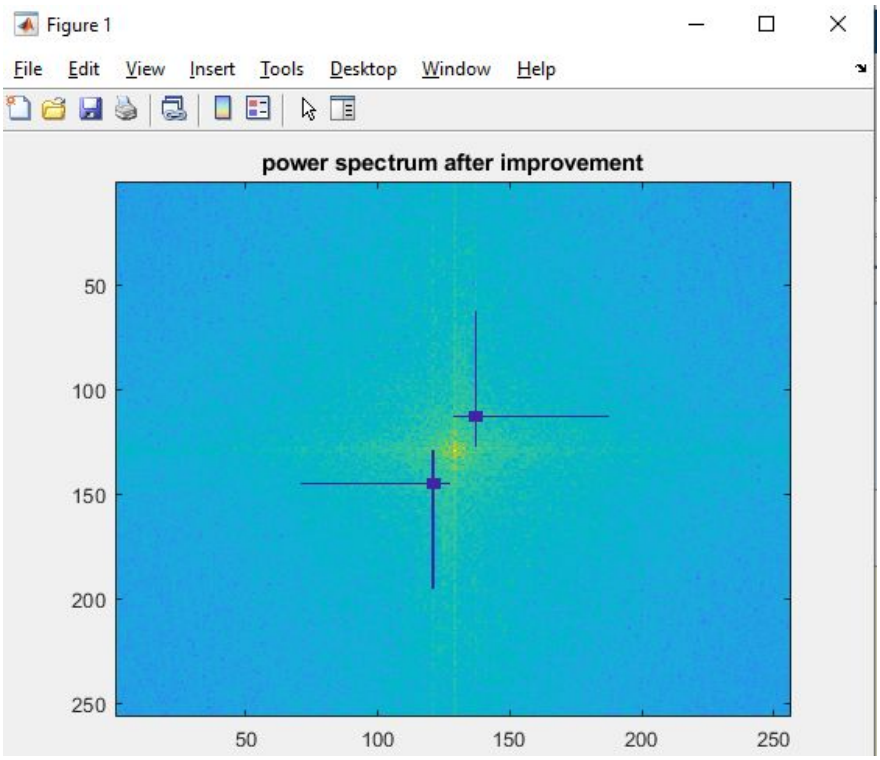


Figure 1
File Edit View Insert Tools Desktop Window Help
power spectrum after improvement
50
100
150
200
250
50
100
150
200
250

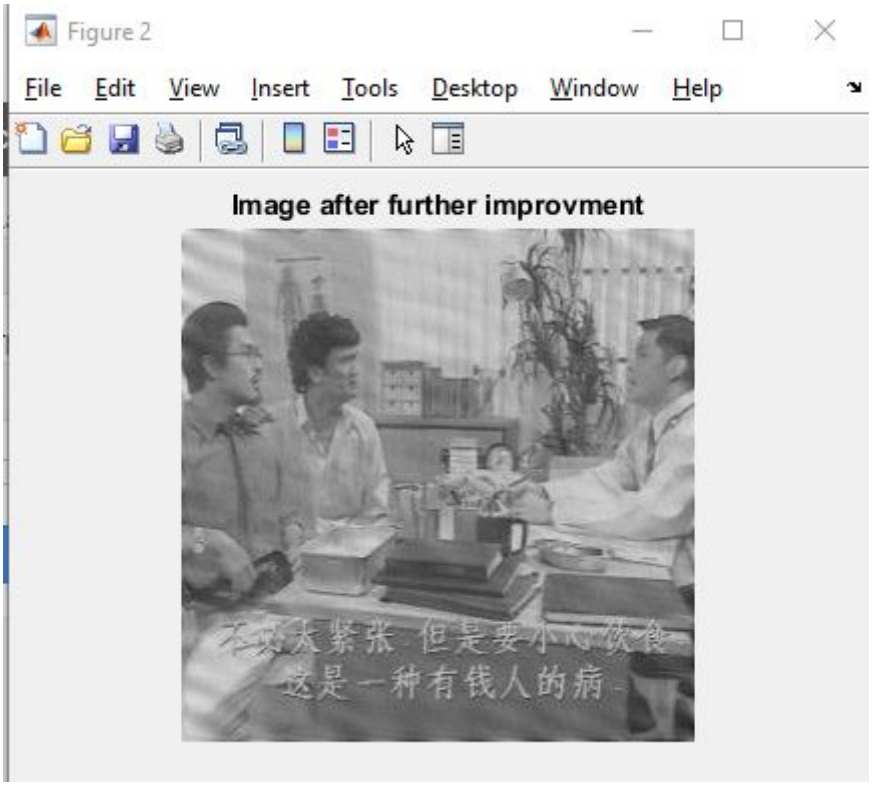


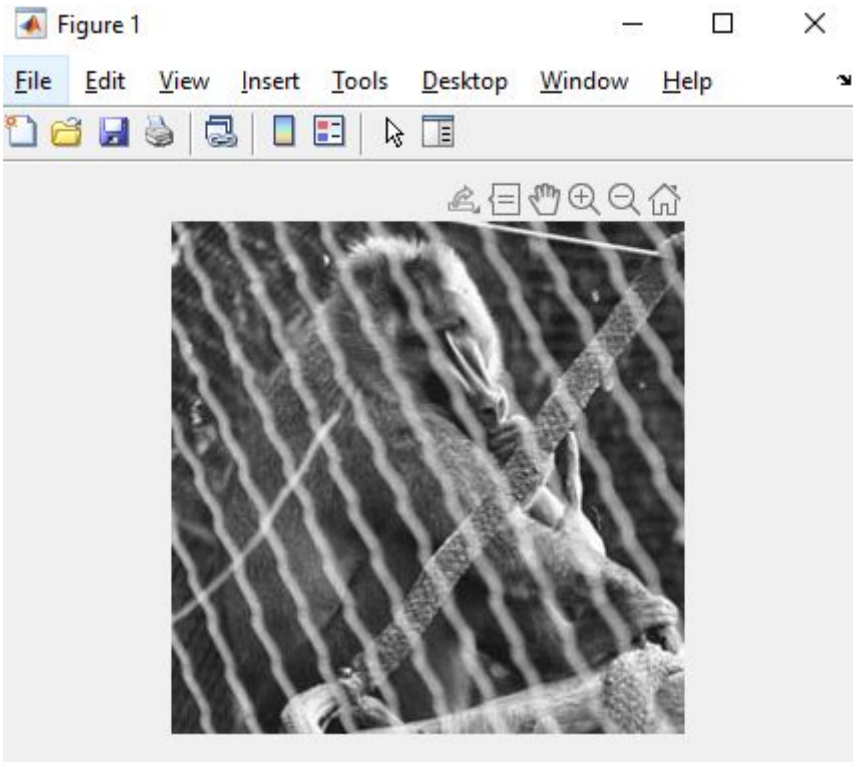
Figure 2
File Edit View Insert Tools Desktop Window Help
Image after further improvment
不要太紧张 但是要小心谨慎
这是一种有钱人的病

F. Free the “primate” by filtering out the fence

Load & display the Image

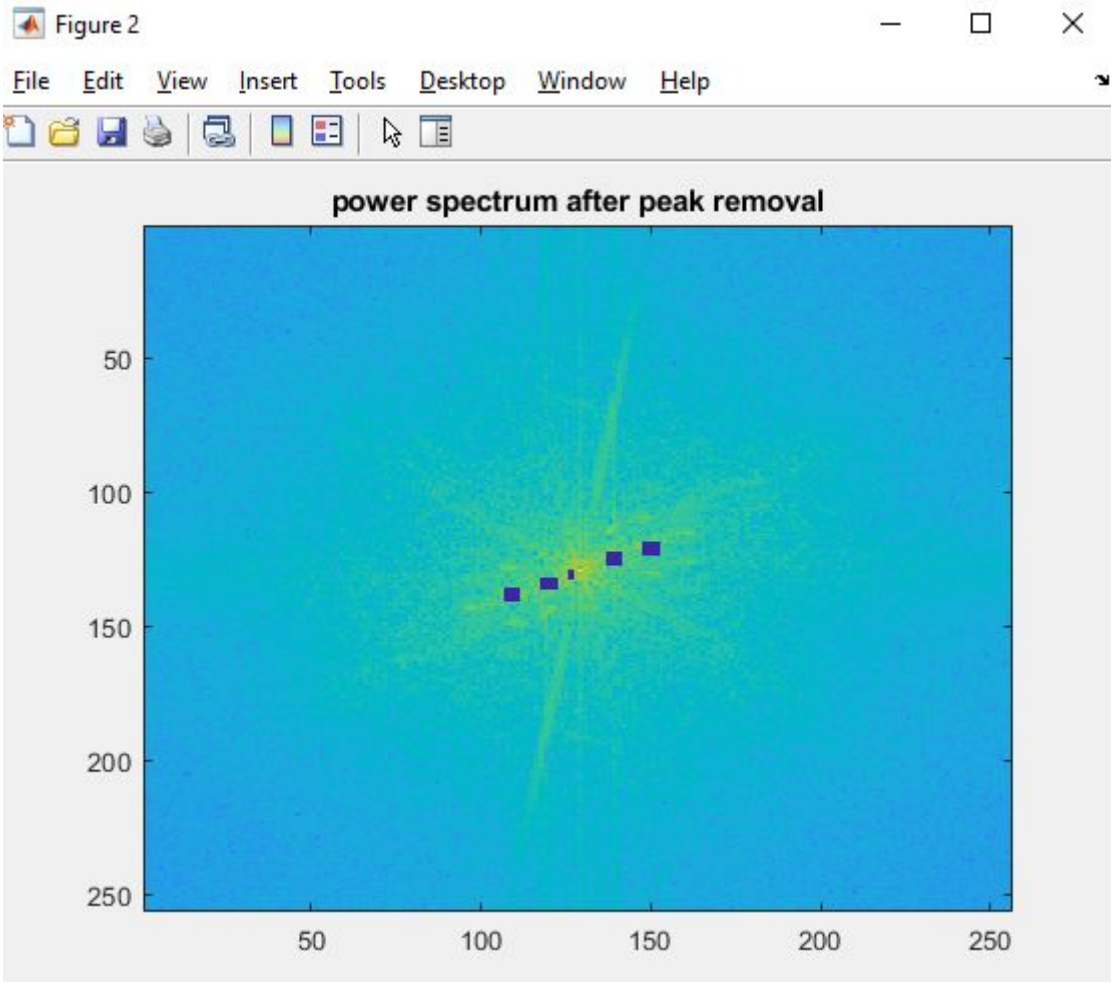
```
P_primate=imread('primate-caged.jpg');
P_primate = rgb2gray(P_primate);
figure
imshow(P_primate);
```

Result:



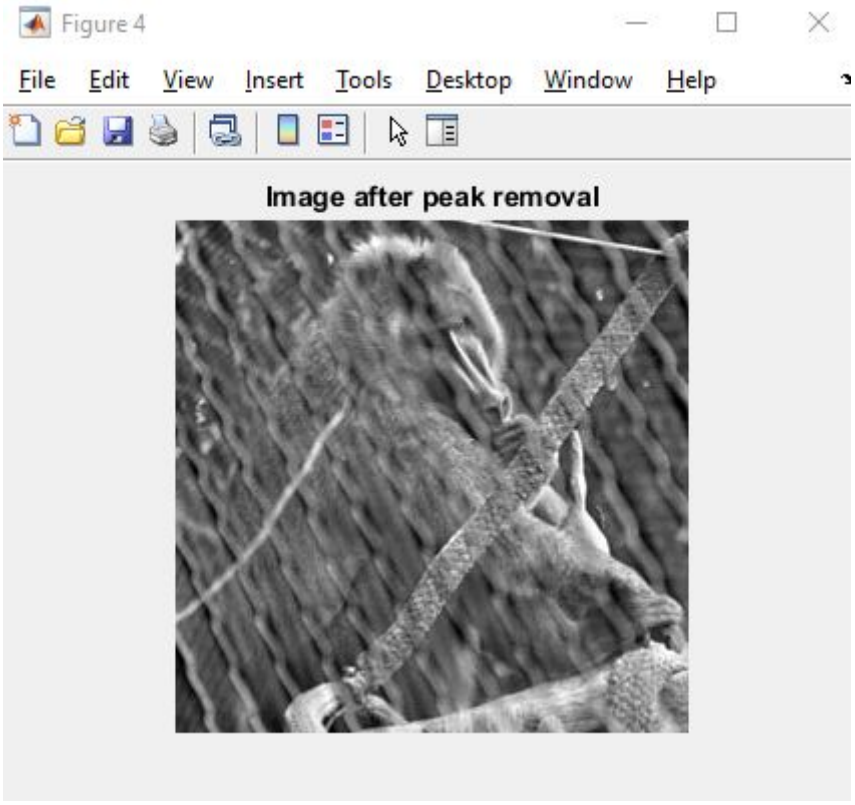

```
%% Display the power spectrum without fftshift and measure actual
location of the peaks
F_primate = fft2(P_primate);
S_primate = abs(F_primate);
figure
imagesc(S_primate.^0.1);
colormap('default');
[xp, yp] = ginput;
%% Set the 5x5 neighbourhood elements at the corresponding peaks to zero
in the Fourier transform F.
xpr = round(xp);
ypr = round(yp);
F_primate_1 = F_primate;
for i = 1: length(xp)
      F_primate_1(max(1,ypr(i)-2):min(ypr(i)+2,
256),max(1,xpr(i)-2):min(xpr(i)+2, 256))  =0;
end
% Recompute the power spectrum
S_primate_1= abs(F_primate_1);
% Display it as in step (b)
figure
imagesc(fftshift(S_primate_1.^0.1))
colormap('default')
title('power spectrum after peak removal')
```

Result:




```
%% Compute the inverse FT and display the resultant image.
P_primate_1 = uint8(ifft2(F_primate_1));
figure
imshow(real(P_primate_1));
title('Image after peak removal');

imshowpair(P_primate, real(P_primate_1), 'montage')
title('Comparison of Image before and after peak removal');
```

Result:



Explanation:

After some trial and error, I have identified 5 possible peaks which correspond to the fence. Those are: (11, 253), (22, 249), (248, 6), (256, 2), (237, 10).

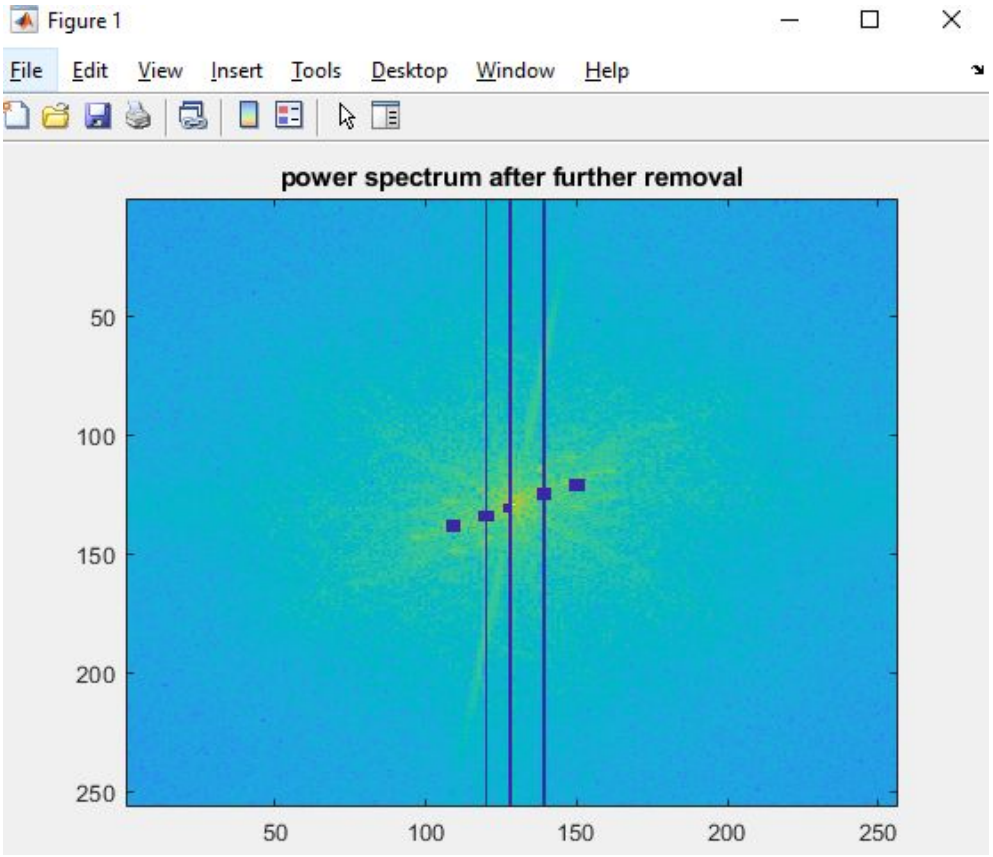
After that, I tried to filter them by doing the same step as (D). After transforming back using ifft2() and displaying the image, it seems that the fence cannot be filtered that well.

Moreover, the original image is also getting “blurred” due to the fact that we also might have filtered some parts that are indeed supposed to be part of the image.
I have tried to use the similar approach in step (E), but only in 1 direction, since I noticed that there is a straight line along the y-axis on the 1st, 3rd, and 4th identified peak.

Code:

```
%% Improvement test (It seems to have little to no effect to the
resulting image)
F_primate_2 = F_primate_1;
for i = [1,3,4]
      % F_primate_2(max(1,ypr(i)-50):min(ypr(i)+50,256), xpr(i)) =0;
      F_primate_2(:, xpr(i)) =0;
      % F_primate_2(yr(i), max(1,xr(i)-50):min(xr(i)+50,256)) =0;
end
% Recompute the power spectrum
S_primate_2= abs(F_primate_2);
% Display it as in step (b)
figure
imagesc(fftshift(S_primate_2.^0.1))
colormap('default')
title('power spectrum after further removal')
```

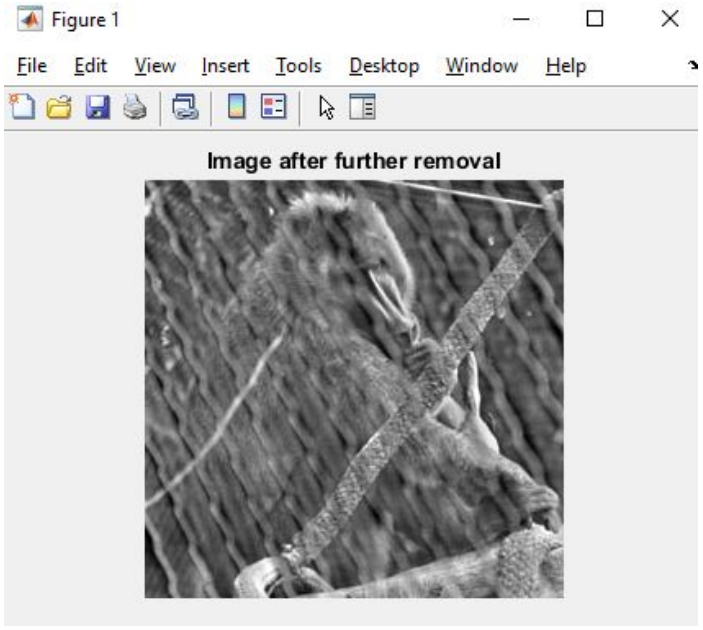
Result:



Compute the inverse FT and display the resultant image.

```
P_primate_2 = uint8(ifft2(F_primate_2));
figure
imshow(real(P_primate_2));
title('Image after further removal');

figure
imshowpair(real(P_primate_1), real(P_primate_2), 'montage')
title('Comparison of Image before and after peak removal');
```

Result:



Explanation:

As you can see in the comparison figure above, it seems that the improvement that I have tried to make did not differ that much. It indeed filters the cage slightly better, but it also makes the image itself a bit more blurry.

2.6 Undoing Perspective Distortion of Planar Surface

a. Load and display the image

```
P_book = imread('book.jpg');
figure;
imshow(P_book);
title('Original image');
```

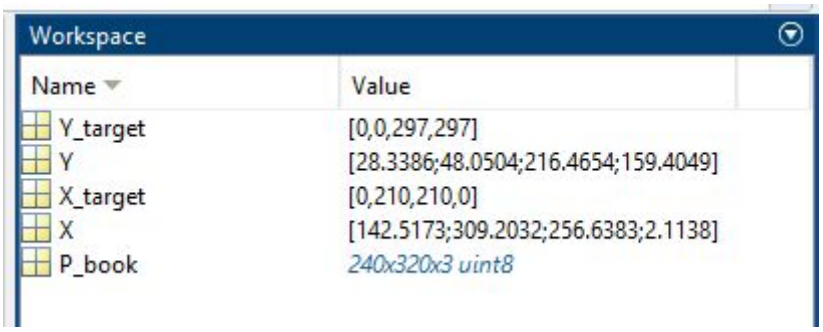
Result:

B. Obtain 4 corners of the book in the original image, and indicate four corners of desired image based on A4 dimensions (210mmx297mm)

```
% Specify vectors x and y to indicate for corners of the desired image
[X, Y] = ginput(4);

% Coord of the 4 corners based on A4 (210x297) in the same order as
above
X_target = [0;210;210;0];
```

```
Y_target = [0;0;297;297];
```

Result:



C. Set up matrices required to estimate the projective transformation based on the given equation

```
A = zeros(8);
v = zeros(8,1);
for i = 1: 8
      if(mod(i,2)==1)
      A(i,1)=X(idivide(i+1,int8(2)));
      A(i,2)=Y(idivide(i+1,int8(2)));
      A(i,3)=1;
      A(i,7)=-X_target(idivide(i+1,int8(2)))*X(idivide(i+1,int8(2)));
      A(i,8)=-X_target(idivide(i+1,int8(2)))*Y(idivide(i+1,int8(2)));
      v(i)= X_target(idivide(i+1,int8(2)));
      else
      A(i,4)=X(idivide(i,int8(2)));
      A(i,5)=Y(idivide(i,int8(2)));
      A(i,6)=1;
      A(i,7)=-Y_target(idivide(i,int8(2)))*X(idivide(i,int8(2)));
      A(i,8)=-Y_target(idivide(i,int8(2)))*Y(idivide(i,int8(2)));
      v(i)= Y_target(idivide(i,int8(2)));
      end

end
%% Computes u = A^-1 * v
u = A \ v;
```

Convert the projective transformation parameters to the normal matrix form

```
U = reshape([u;1], 3, 3)';
U
```

Result:

```
U =

    1.4495    1.5528 -250.5826
   -0.4382    3.7054  -42.5567
    0.0001    0.0053    1.0000
```

Verify that this is correct by transforming the original coordinates

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

Result:

```
w =

    0.0000  210.0000  210.0000         0
   -0.0000    0.0000  297.0000  297.0000
    1.0000    1.0000    1.0000    1.0000
```

Explanation:

These coordinates are indeed the 4 corners of the target image.

D. Warp the image

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

E. Display the image

```
figure;
imshow(P2);
title('Transformed image');
```

Result:

Explanation:

The resulting image is pretty much what I expected. The book in the original image has been transformed into 2d Image. It can be observed that the lower pixels of the transformed image are quite sharp and are getting blurry to the upper pixels of the image. This is expected since in the original picture, the bottom part of the book is bigger and more focused, and the top areas of the book are smaller and less focused. Thus, the bottom part is “squeezed” and the top part of the book is stretched to fit the desired image. Thus, this makes the resulting image becomes what we can see above.

Additional Information

Links to MATLAB code references that are not obtained from the Lab manual have been put as code comments above the referenced code where it is first used.

The original source codes will be uploaded alongside the submission of this report, and are also available at https://github.com/hanstananda/CZ4003-Lab1.