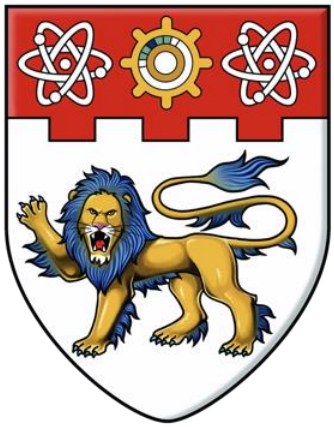


NANYANG TECHNOLOGICAL UNIVERSITY

CZ4003: Computer Vision

Lab 1 Report

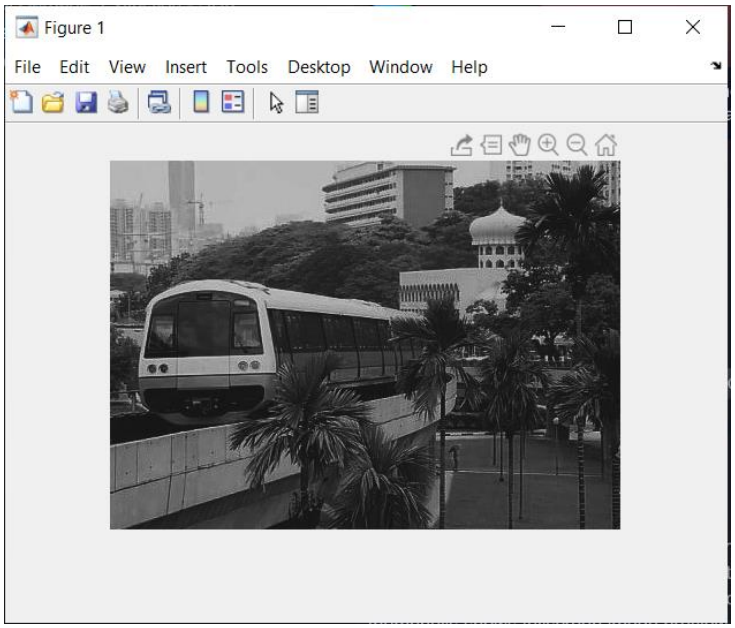
Nigel Ang Wei Jun

U1721087C

2. Experiments

2.1 Contrast Stretching

Original Image:

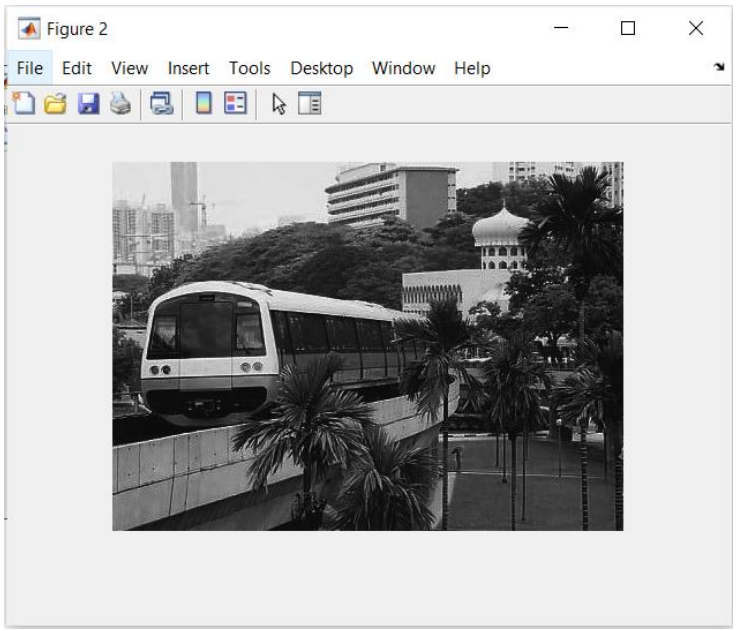


Contrast Stretching is applied to array of pixels in original image P via ‘imsubtract’ and ‘immultiply’

$$s = \frac{255\left(r - r_{\min}\right)}{r_{\max} - r_{\min}}$$

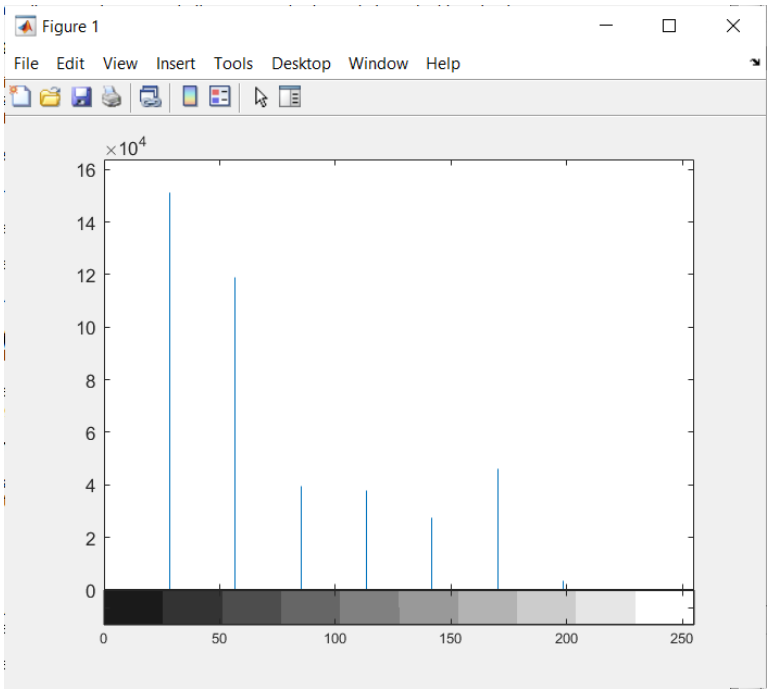
```
min(P(:)), max(P(:))

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

After Contrast Stretching:

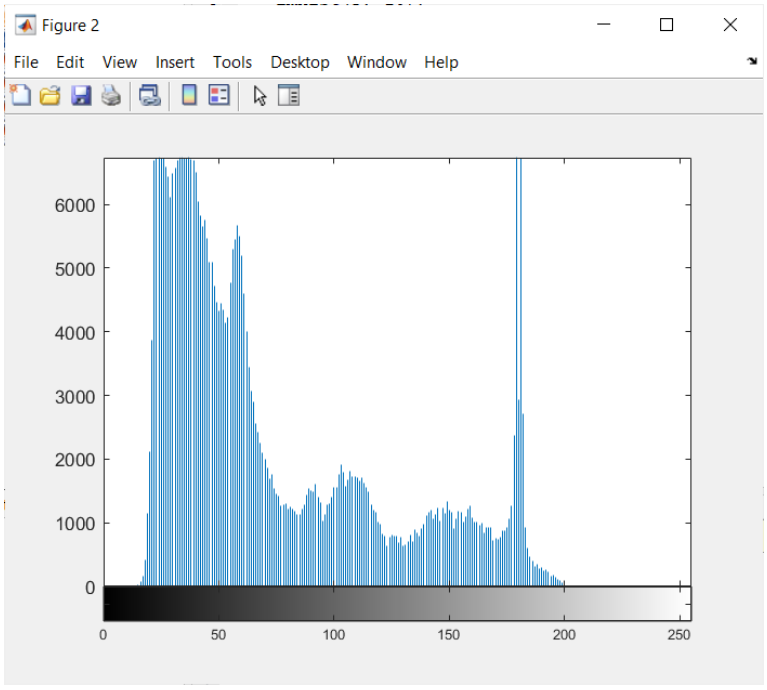


2.2 Histogram Equalisation

Histogram with 10 bins:



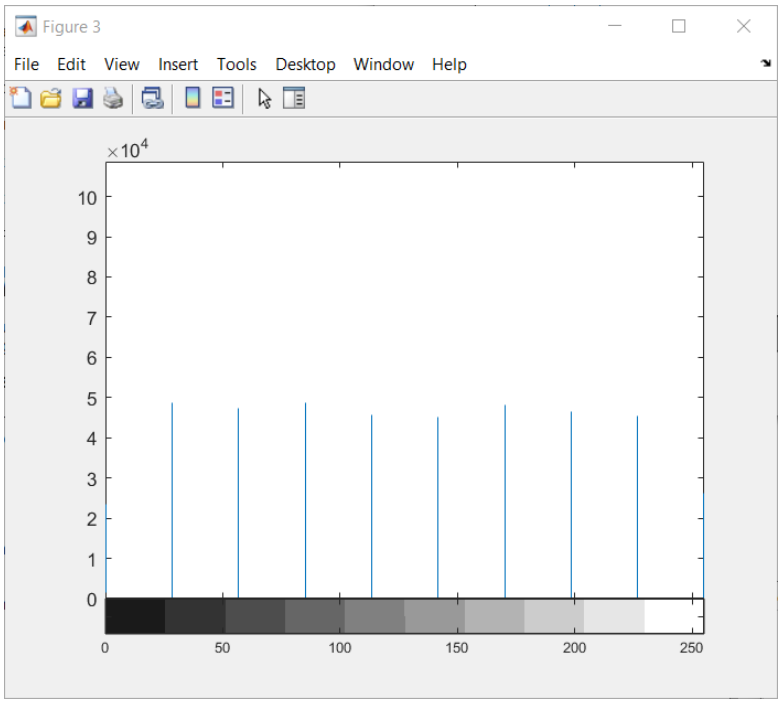
Histogram with 256 bins:



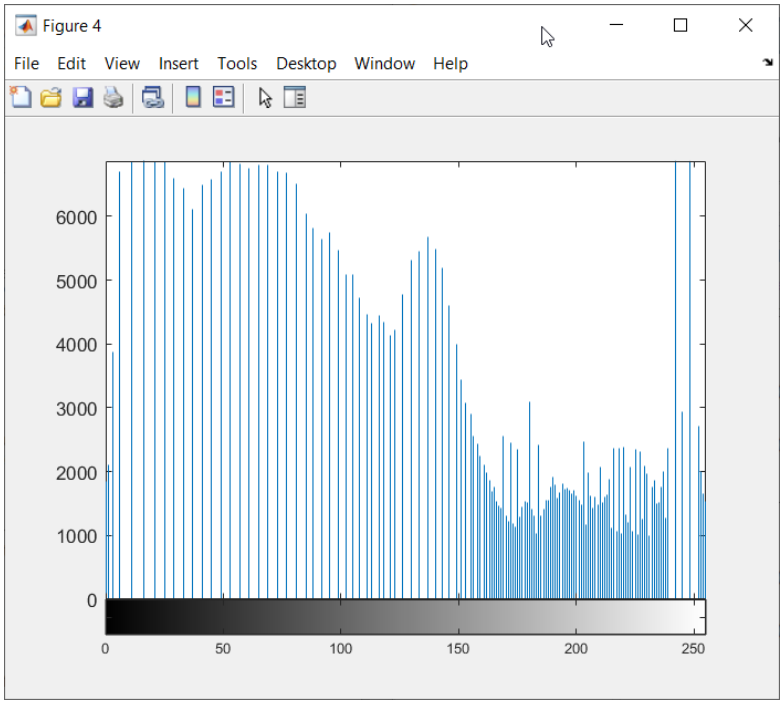
There are many more ‘lines’ on the graph, as more bins mean intensities can fall under a greater variety of bins; There is a greater distribution of intensities across the bins

After performing Histogram Equalisation:

Histogram with 10 bins:



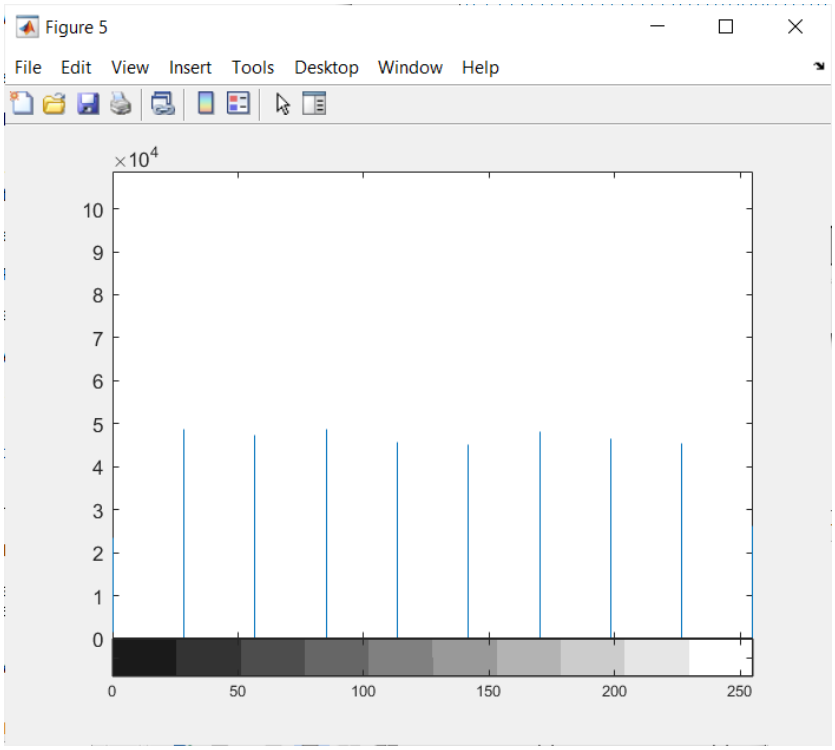
Histogram with 256 bins:

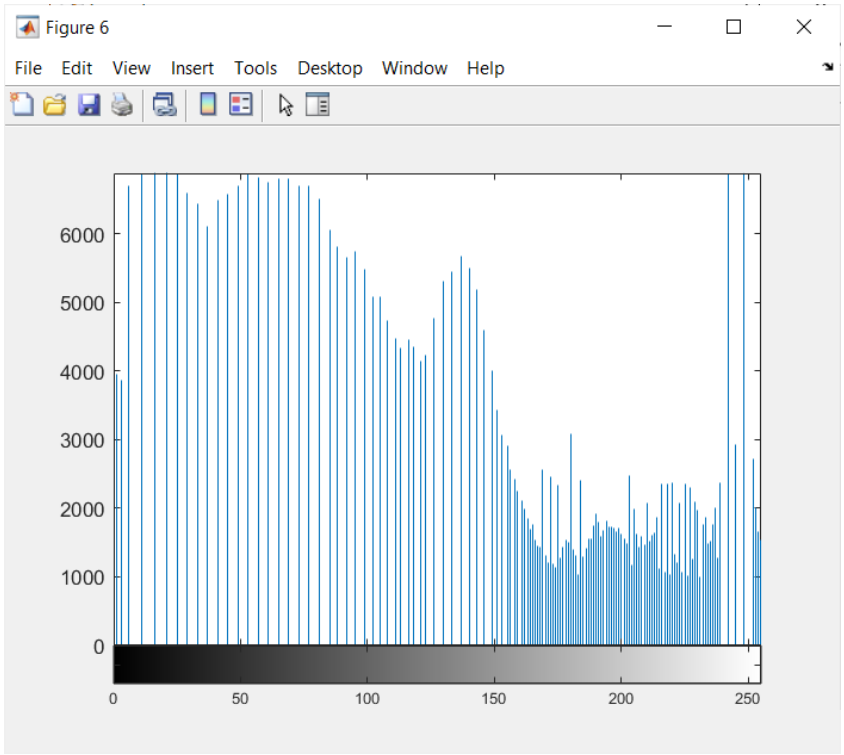


Histograms are equalised for 10 bins, but not for 256 bins. The number of bins before and after remains the same, but the intensities are more spread out into different bins.

After performing histeq() again, no further changes occur.

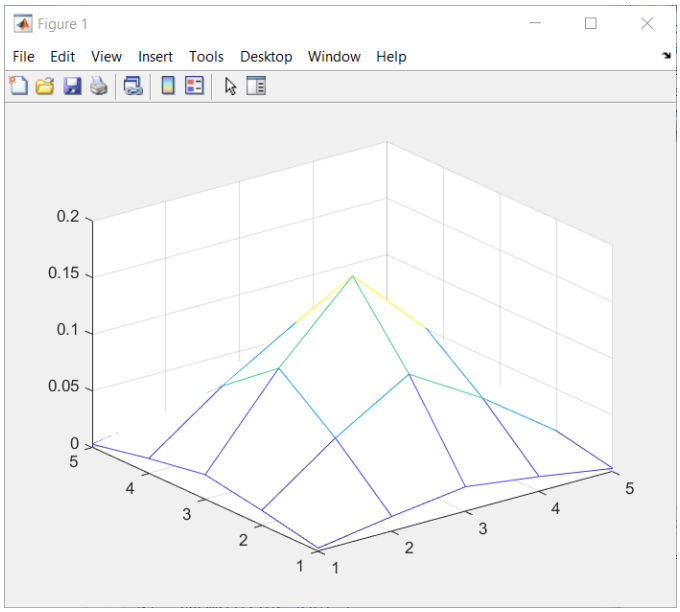
histeq() is an idempotent function; The value of the histogram equalization does not change when it is multiplied by itself.



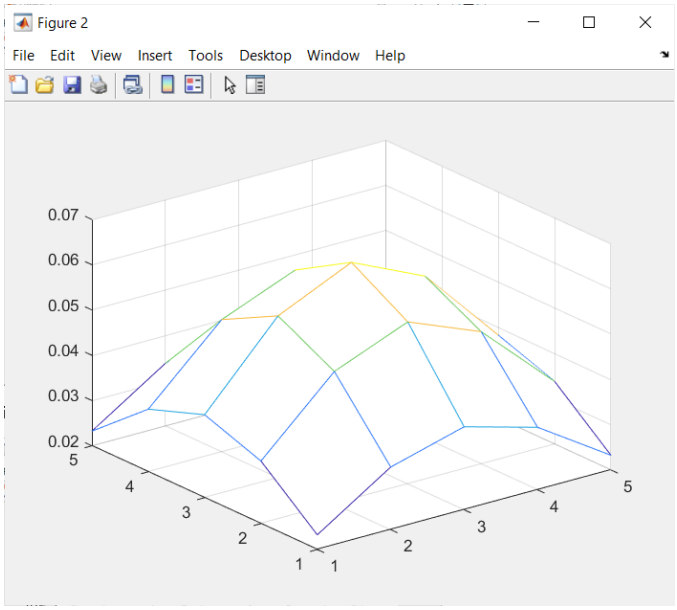


2.3 Linear Spatial Filtering

i) X, Y = 5, σ = 1.0

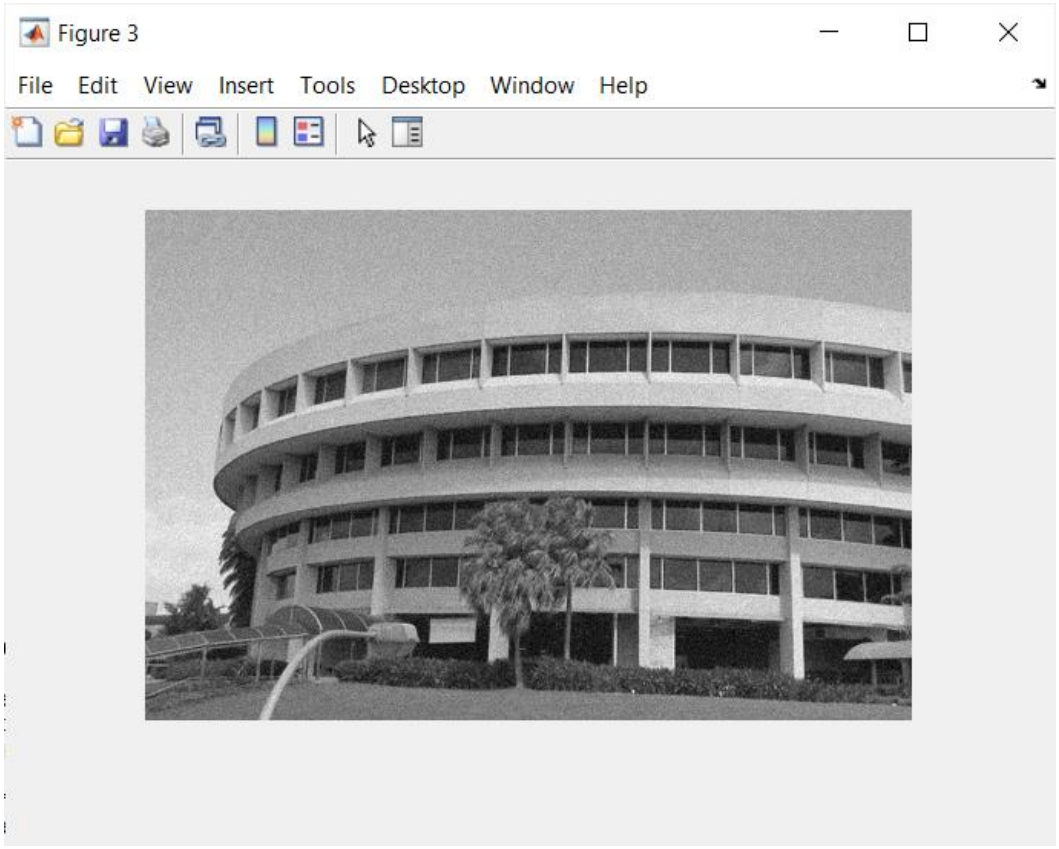


ii) X, Y = 5, σ = 2.0

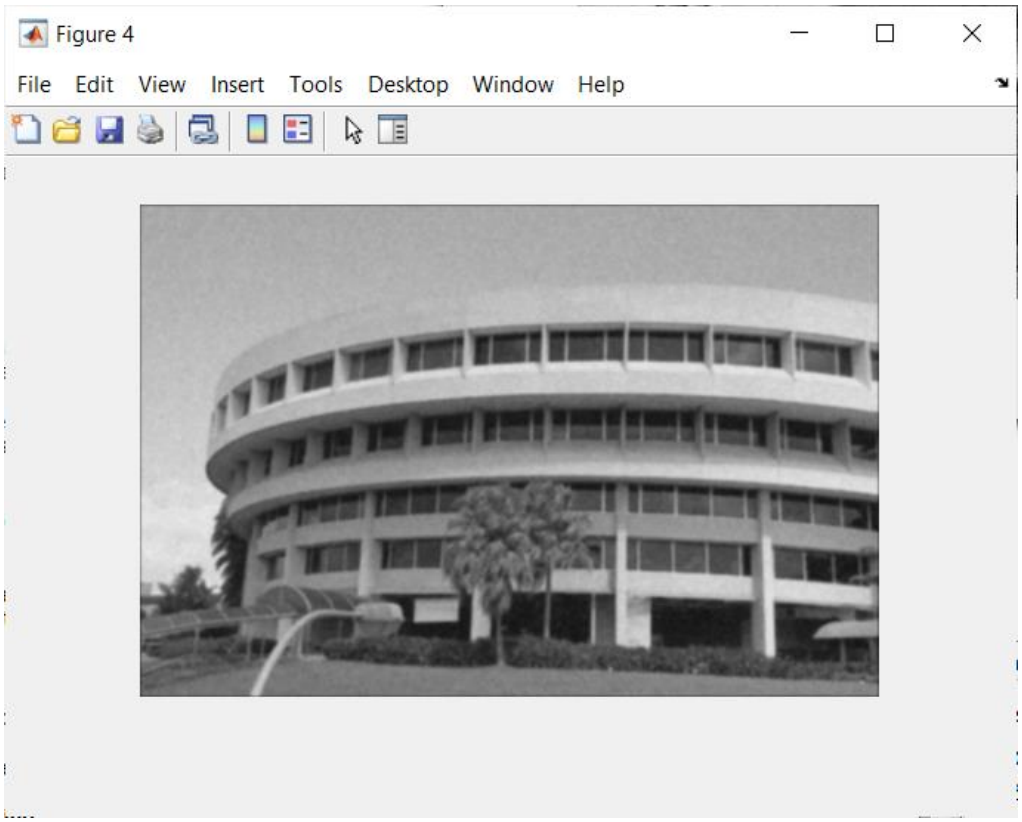



```
%% Gaussian Noise
% sigma = 1
h1 = fspecial('gaussian', 5, 1);
h1 = (h1/sum(h1(:)));
mesh(h1)

% sigma = 2
h2 = fspecial('gaussian', 5, 2);
h2 = (h2/sum(h2(:)));
figure
mesh(h2)
```

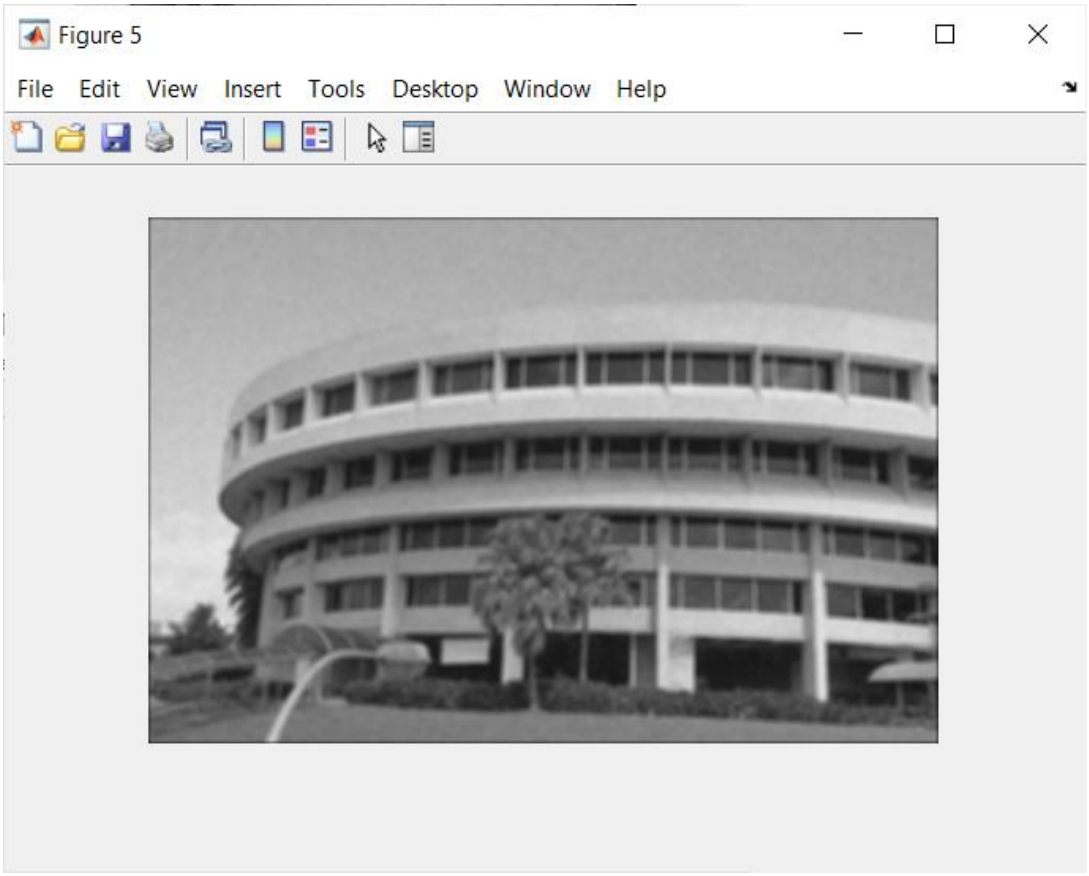
Original Image (Gaussian Noise):



Applying 3x3 Gaussian Filter



Applying 5x5 Gaussian Filter:

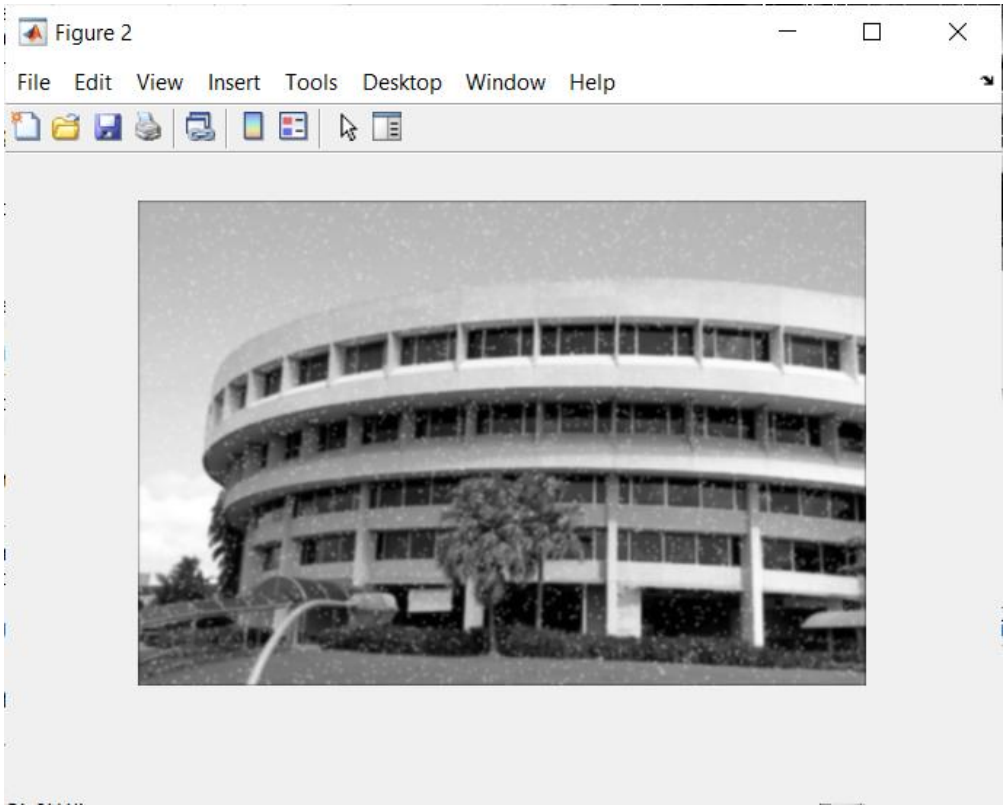


Filtering the image removes the noise but blurs the picture features as well.

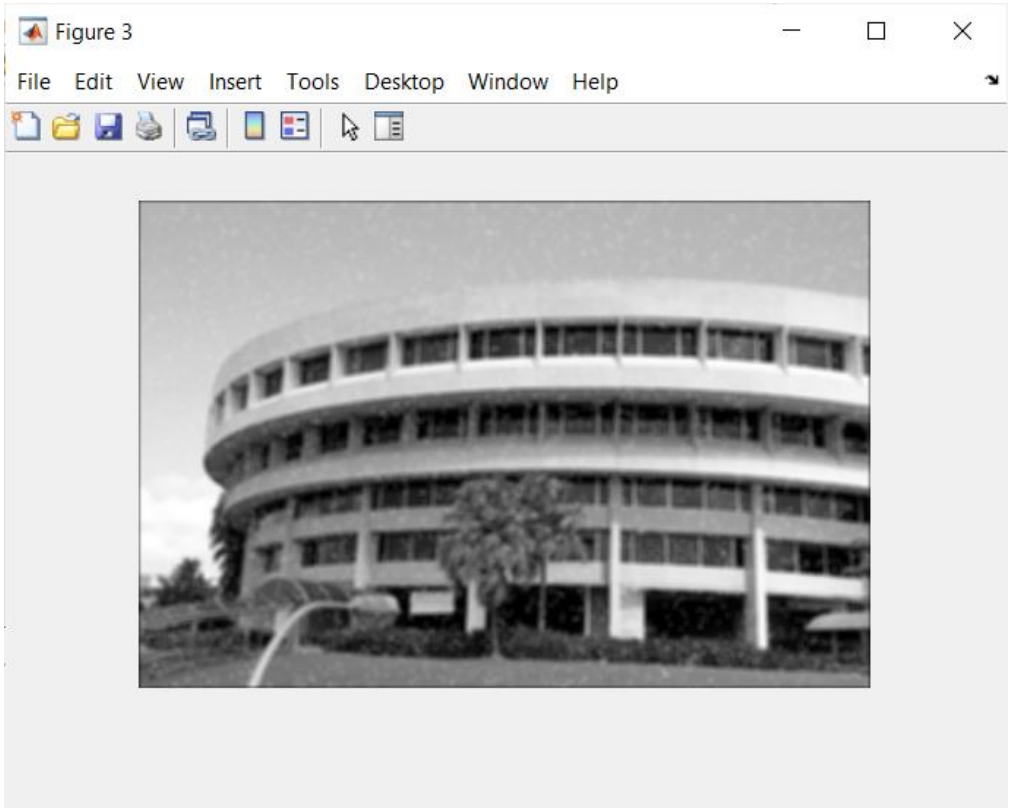
- 5x5 filter causes more smoothing/blurring than 3x3 filter.
- Trade-off between picture clarity vs. removal of noise

Original Image (Speckle Noise):

Applying 3x3 Gaussian Filter:



Applying 5x5 Gaussian Filter:

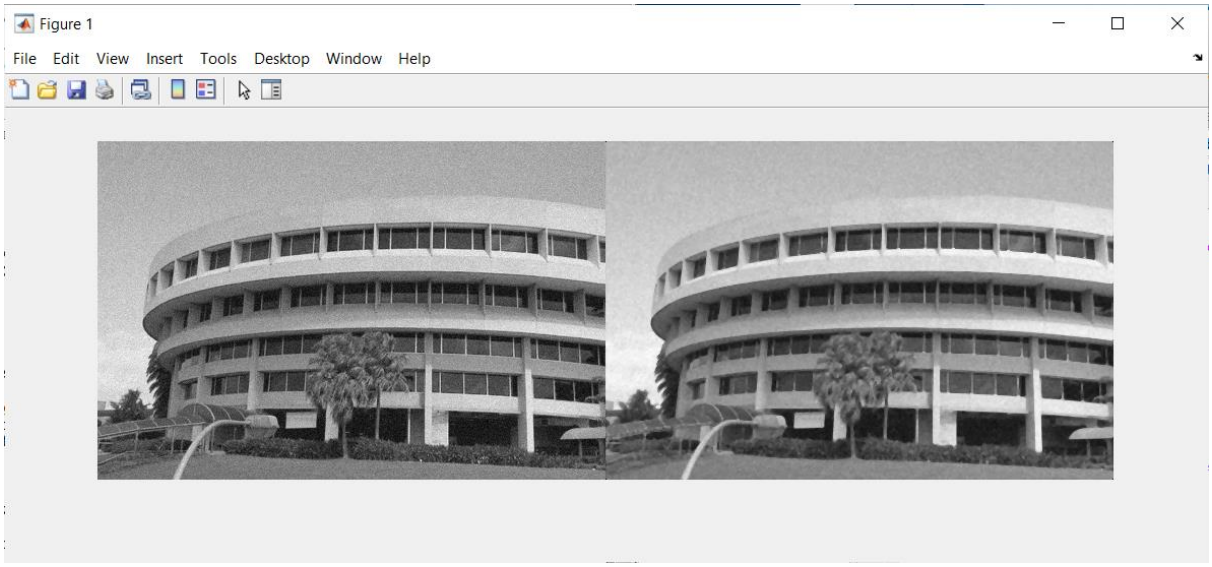


Speckle noise can still be observed in both pictures; Filters are better at handling Gaussian Noise.

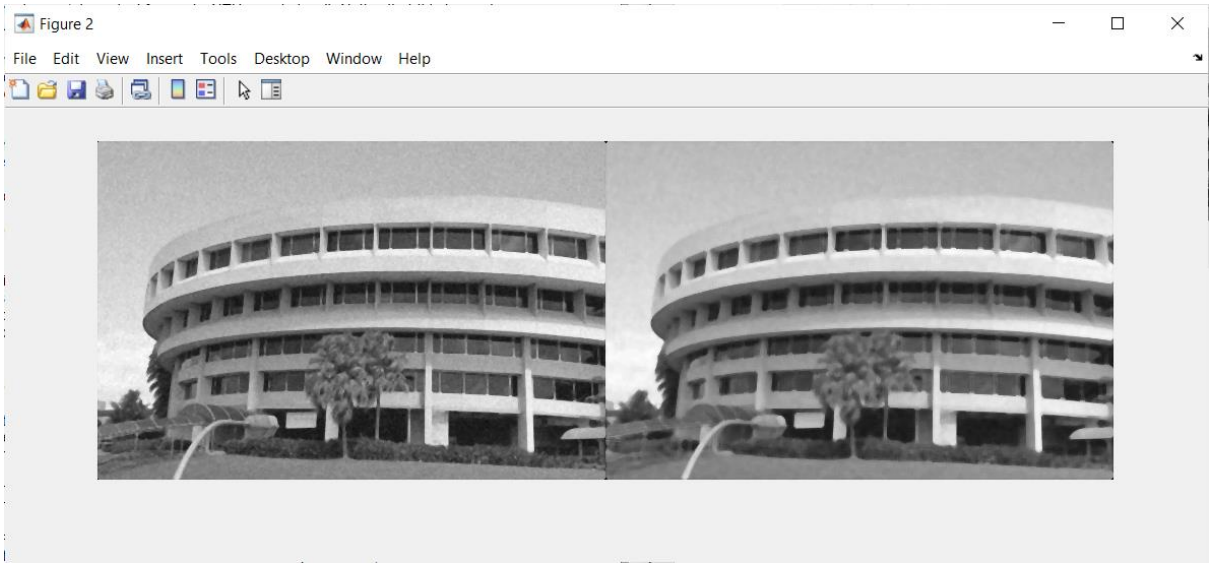
2.4 Median Filtering

Using medfilt2 (Gaussian Noise):

- Left: Original; Right: 3x3 Median Filter

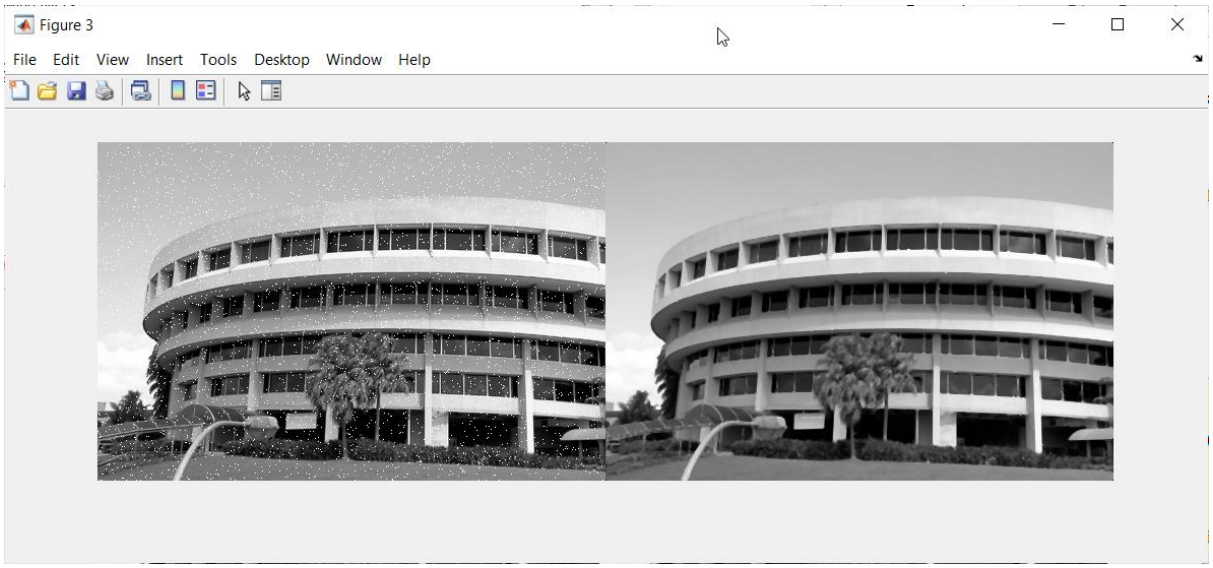


- Left: 3x3 Median Filter; Right: 5x5 Median Filter

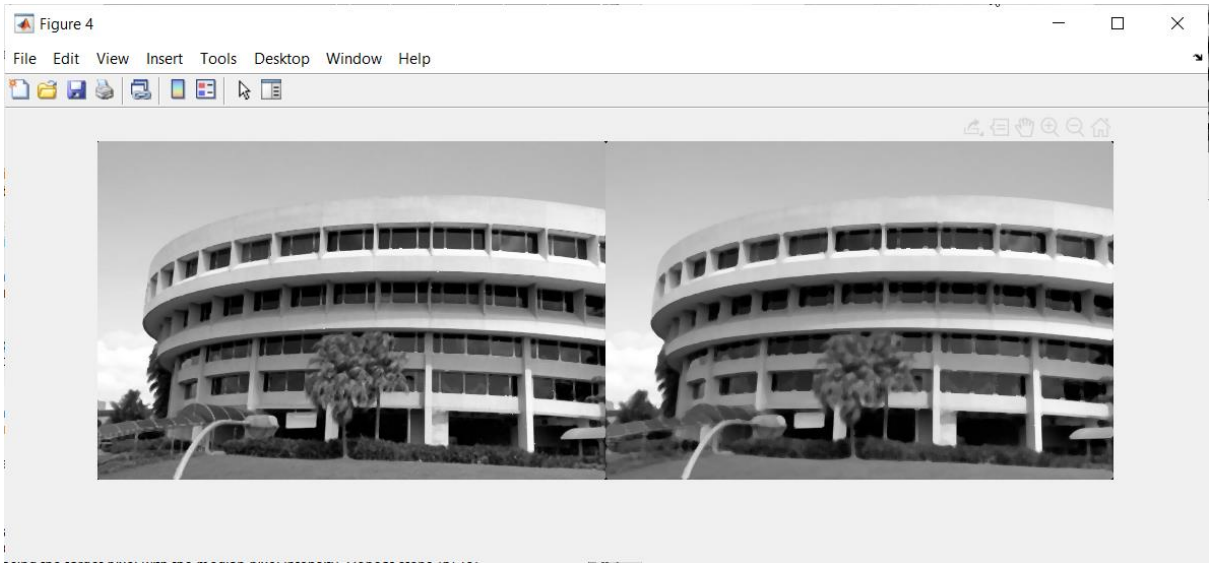


Using medfilt2 (Speckle Noise):

- Left: Original; Right: 3x3 Median Filter



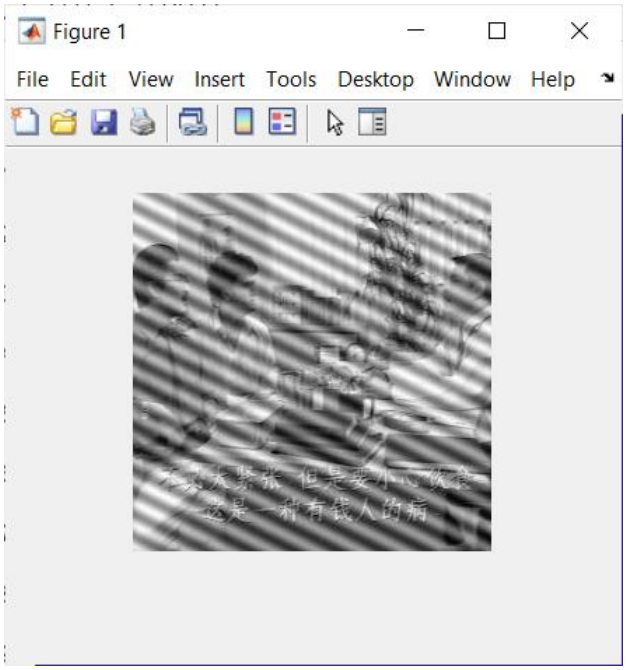
- Left: 3x3 Median Filter; Right: 5x5 Median Filter



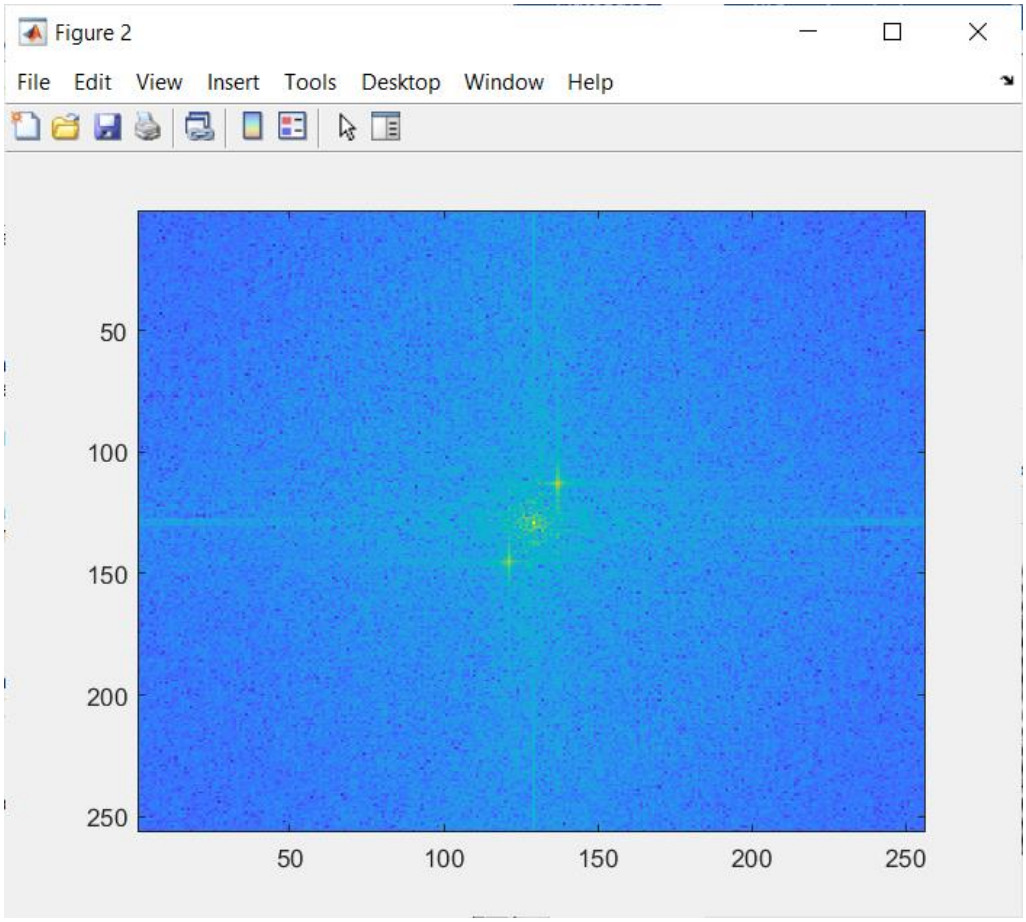
- Median filters are capable of handling both Gaussian and Speckle noise
- 5x5 median filter causes significant smoothing of picture
- Median filter preserves edges, but is slower to compute result

2.5 Suppressing Noise Interference Patterns

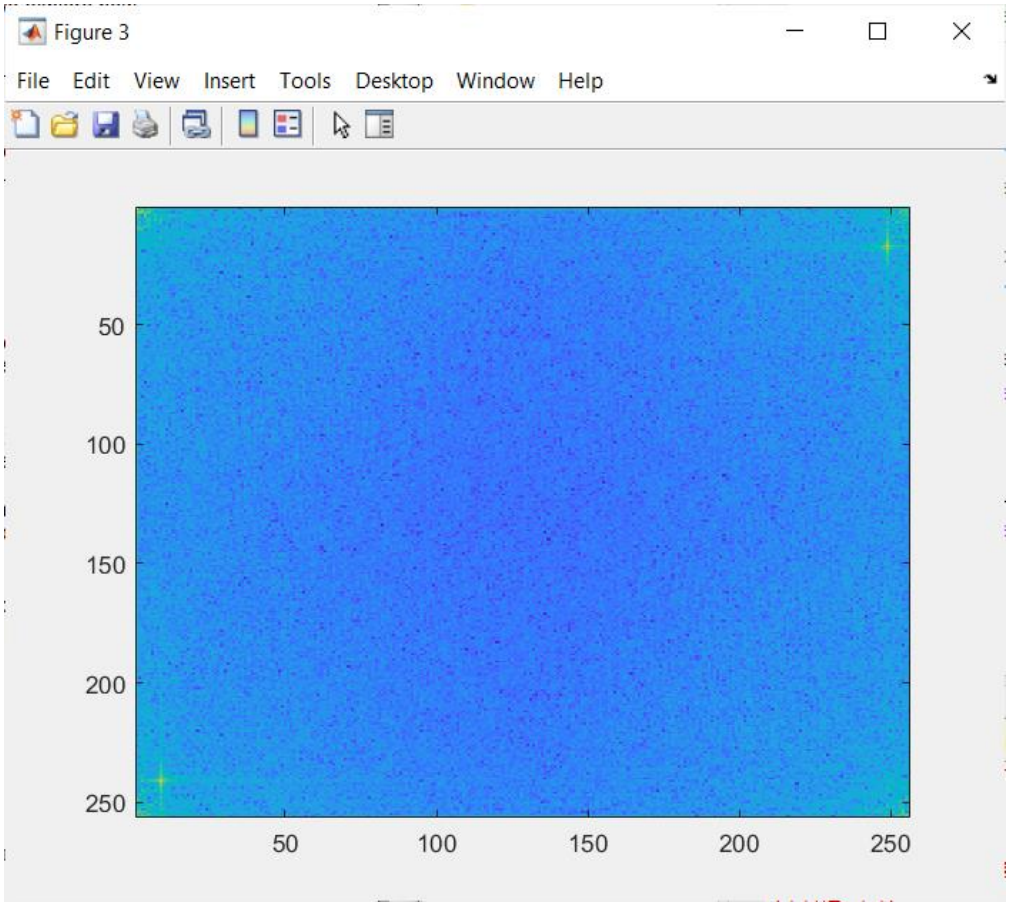
Original Image:



Power Spectrum S with fftshift:



Power Spectrum S without fftshift:

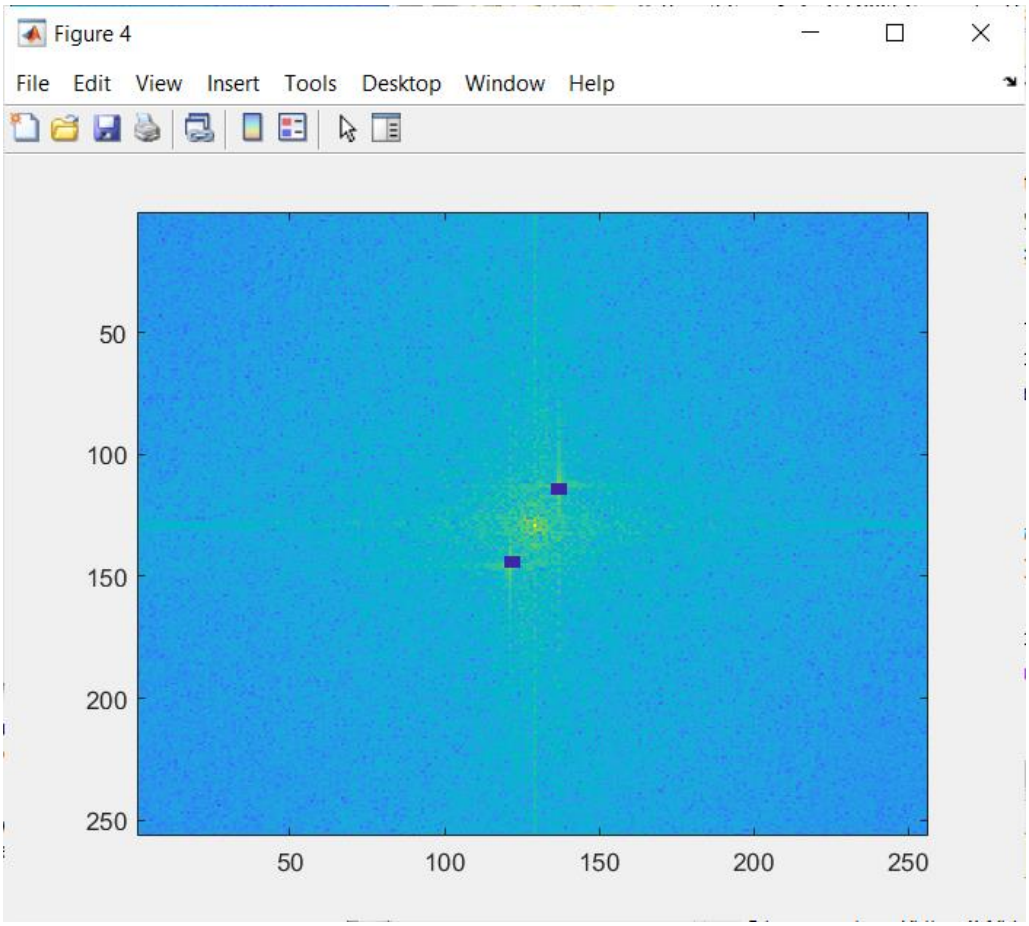


Peaks: (249.6639, 16.9486), (9.2219, 241.8458)

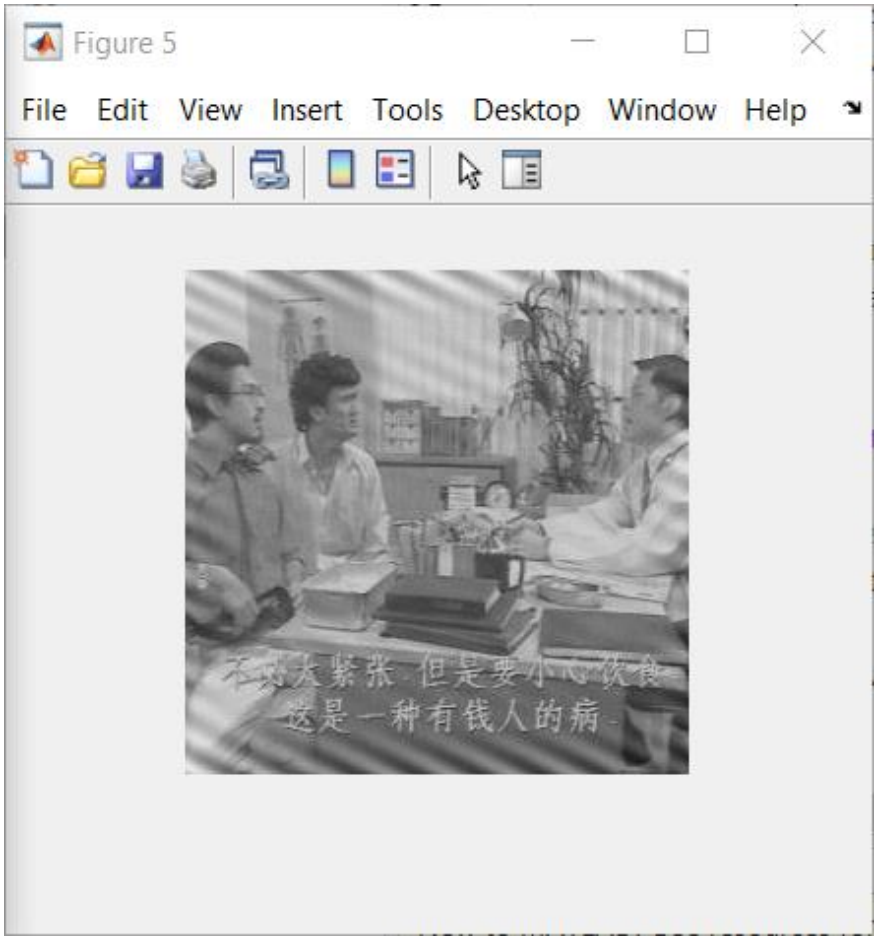
```
x =

  249.6639
    9.2219


y =

   16.9486
  241.8458
```

```
% d.
F_new = F;
[x, y] = ginput(2);
% (x, y) => F(y, x)
for i = 1: length(x)
    for j=-2:2
        for k=-2:2
            fprintf('At pixel: (%d, %d)\n', round(x(i))+j, round(y(i))+k);
            F_new(round(y(i))+k, round(x(i))+j) = 0;
        end
    end
end
% Recompute power spectrum
S_new = real(F_new);
figure
imagesc(fftshift(real(S_new.^0.1)));
colormap('default');
```

Upon setting the 5x5 neighbourhood elements of the selected (x, y) values to 0:



Resultant picture:

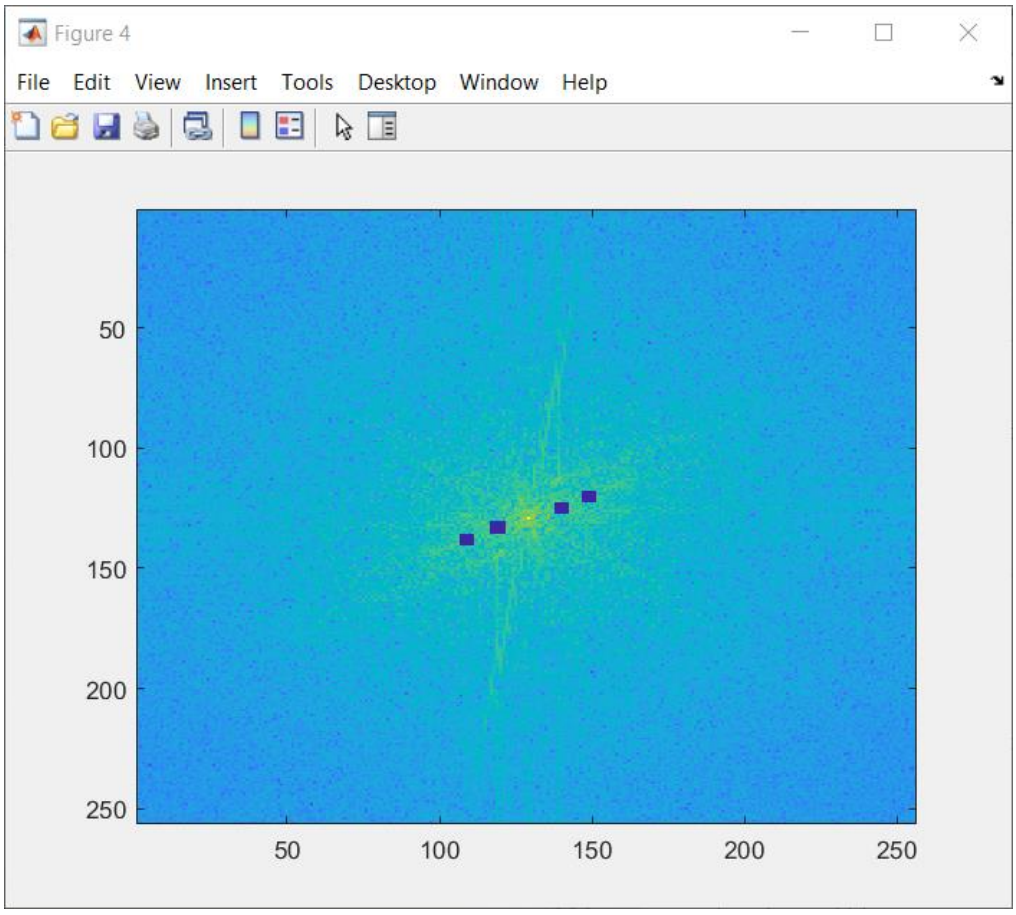


The noise is not completely removed, but we can see improvement!

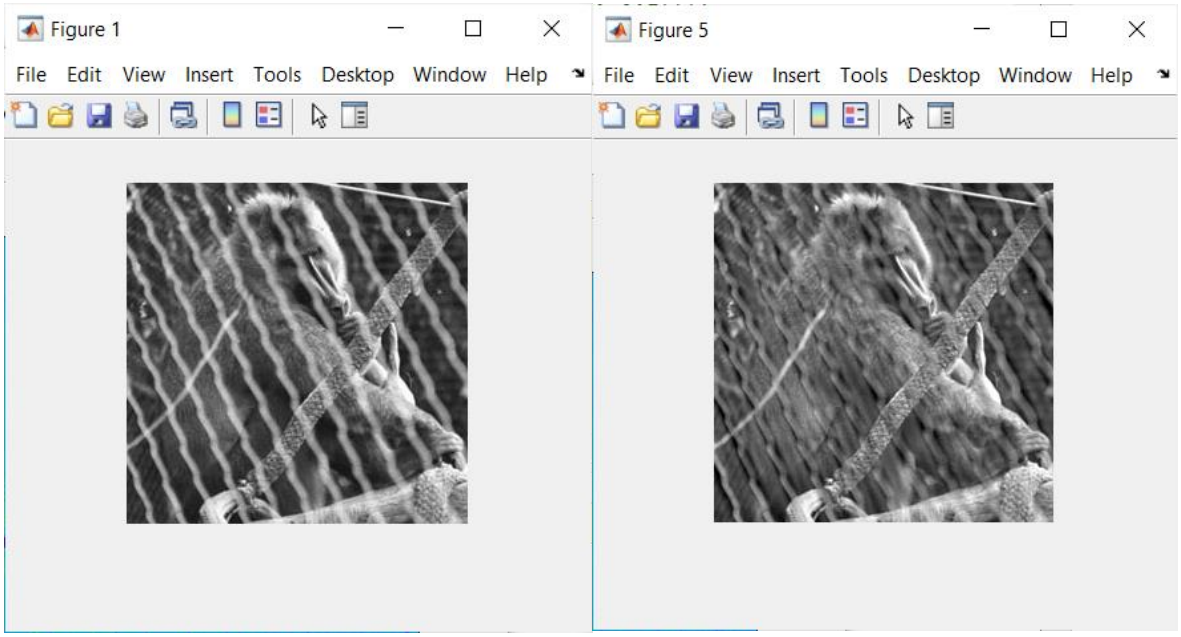
Now we can see it is picture of an episode from local TV series ‘Phua Chu Kang’.

2.5f Freeing the Primate

After attempting to remove the 4 bright spots

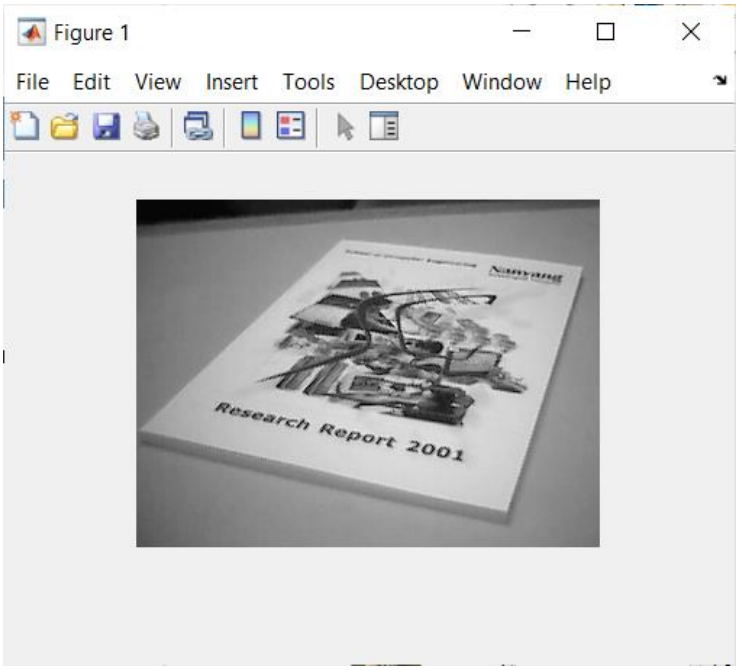


Original Image vs Result:



2.6 Undoing Perspective Distortion of Planar Surface

Original Image: Book dimensions are 210x297; Scale – 1px : 1mm



We select the corners in a clockwise manner, starting from the top left corner

We obtain the matrices A and v:

```
% Au = v
A = [x(1) y(1) 1    0    0    0 -x_im(1)*x(1) -x_im(1)*y(1);
     0    0    0    x(1) y(1) 1 -y_im(1)*x(1) -y_im(1)*y(1);
     x(2) y(2) 1    0    0    0 -x_im(2)*x(2) -x_im(2)*y(2);
     0    0    0    x(2) y(2) 1 -y_im(2)*x(2) -y_im(2)*y(2);
     x(3) y(3) 1    0    0    0 -x_im(3)*x(3) -x_im(3)*y(3);
     0    0    0    x(3) y(3) 1 -y_im(3)*x(3) -y_im(3)*y(3);
     x(4) y(4) 1    0    0    0 -x_im(4)*x(4) -x_im(4)*y(4);
     0    0    0    x(4) y(4) 1 -y_im(4)*x(4) -y_im(4)*y(4);];
v = [x_im(1);
     y_im(1);
     x_im(2);
     y_im(2);
     x_im(3);
     y_im(3);
     x_im(4);
     y_im(4);];
```

We obtain U:

```
U =

    1.5132    1.5710 -261.9512
   -0.4269    3.7294  -47.1123
    0.0003    0.0052    1.0000
```

Checking w:

```
w =

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

We obtain back our 4 corners: (0, 0), (210, 0), (210, 297) and (0, 297), corresponding to the 4 corners in clockwise order.

Result:

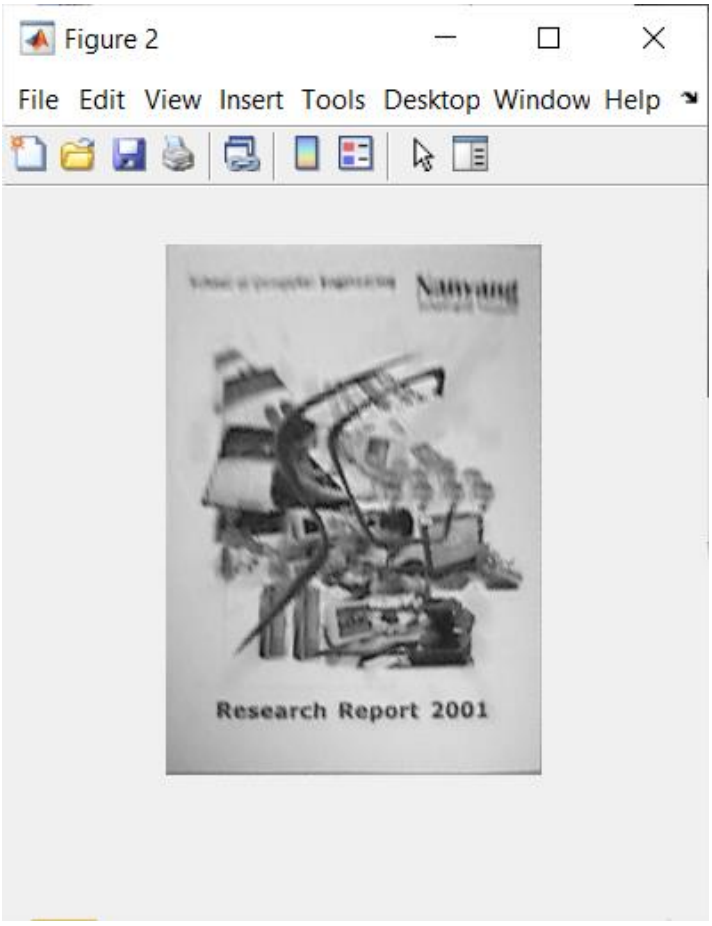


Image is a little blur on the top, as the top parts in the original image are being rescaled, hence loss to be expected in the result image.