CE4003 Computer Vision Lab 1 Assignment Report

2.1 Contrast Stretching

2.1a. Input image into MATLAB matrix variable

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

Properties of Pc:

```
whos Pc
```

```
Name         Size                 Bytes  Class    Attributes

Pc         320x443x3             425280  uint8
```

Convert to grayscale image

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

2.1b. Viewing the image

```
figure;imshow(P);
```

2.1c. Checking minimum and maximum intensities present in image

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

```
ans =

  uint8

   13


ans =

  uint8

   204
```

Minimum = 13

Maximum = 204

2.1d. Contrast Stretching

```
x = double(P);
y = x - 13;
z = (255/(204-13))*y;
P2 = uint8(z);

min(P2(:)),max(P2(:))
```

```
ans =

  uint8

   0


ans =

  uint8

   255
```

Final minimum = 0

Final maximum = 255

Contrast stretching involves scaling the image of intensities from 13 to 204, into intensities from 0 to 255.

2.1e. Redisplay image P2

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

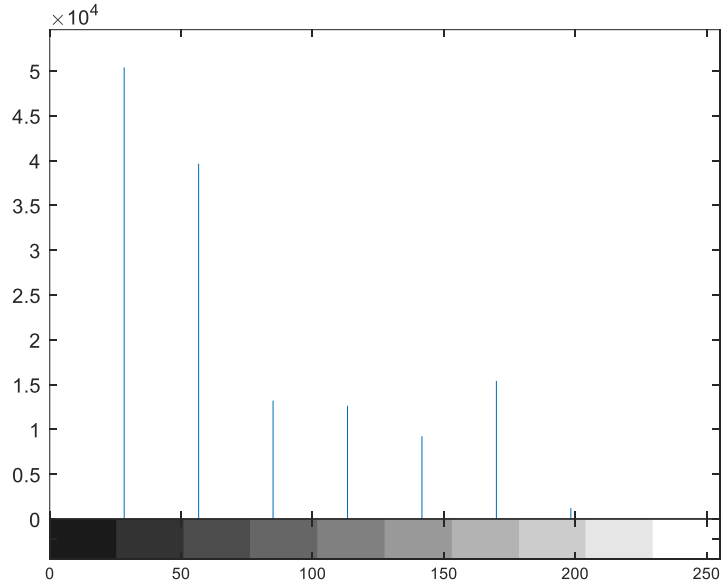
Contrast stretching is used to increase the contrast between light and dark areas. Comparing images P and P2, we have a clearer differentiation between the light and dark areas in image P2.

2.2 Histogram Equalization

2.2a Display image intensity of histogram P

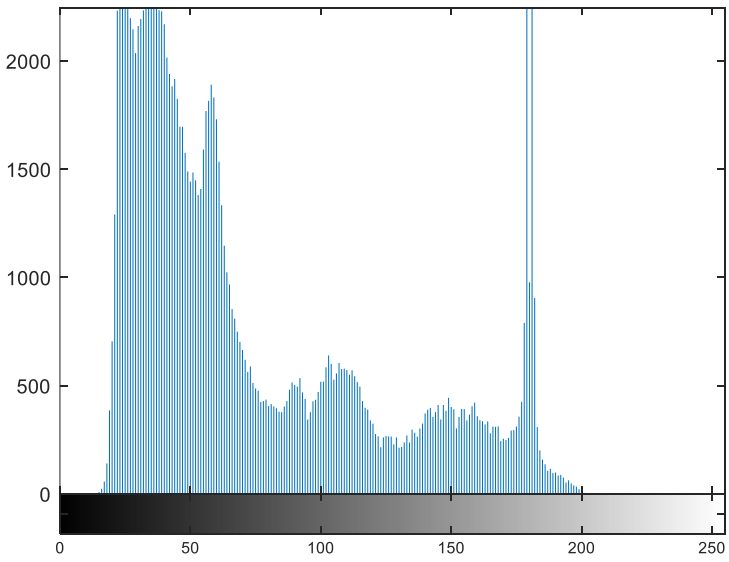
Using 10 bins:

```
%using 10 bins
figure;imhist(P,10);
```



Using 256 bins:

```
%using 256bins
figure;imhist(P,256);
```



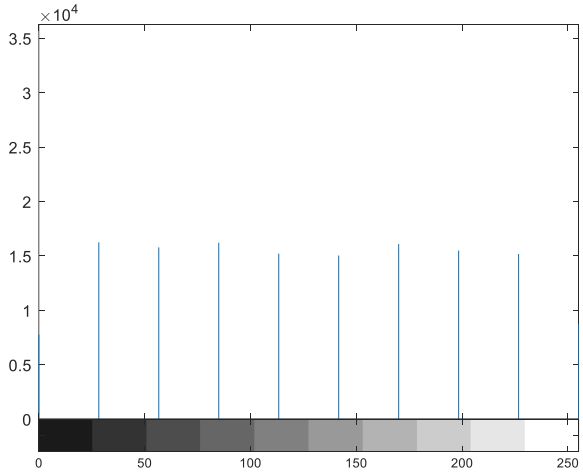
Differences:

There are more details shown in figure with 256 bins as compared to the figure with 10 bins. Since there are 256 levels in a grayscale image, we can see the intensity clearly for levels from 0 to 255. With only 10 bins, we miss the detail of the intensity between certain gray levels, such as the gray level area between 25 to 50, when comparing the 2 figures.

2.2b. Implement histogram equalization

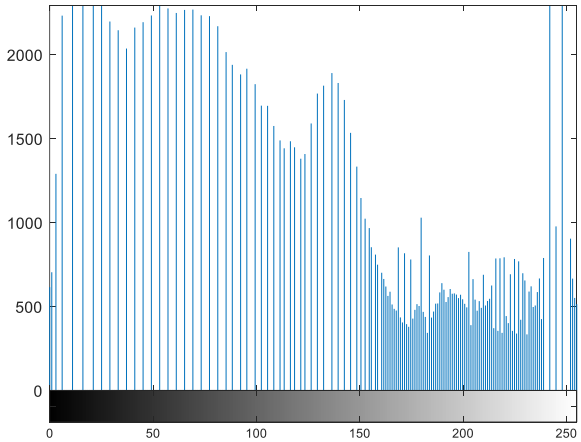
Histogram of P3 with 10 bins

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



Histogram of P3 with 255 bins

```
figure;imhist(P3,255);
```



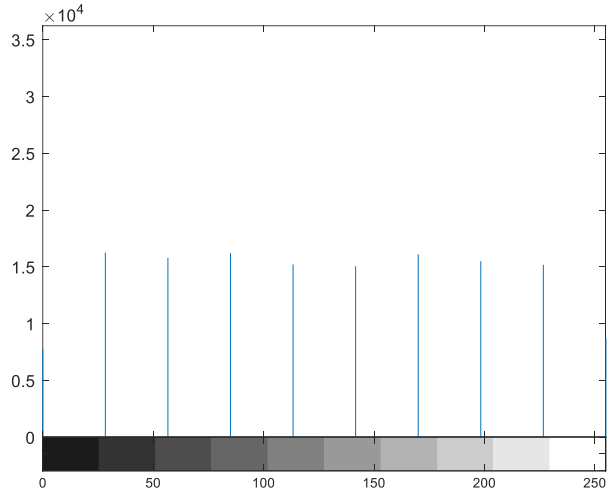
Are the histograms equalized? Similarities and differences:

Figure P3 with 10 bins are more equalized as compared to figure with 256 bins. Both figures show how histogram equalized are trying to flatten the gray-level histogram through a gray-level transformation, combining the bins with high intensity to bins of lower intensities. The figure with 256 bins might not still be flat as it is difficult to combine exactly to match the same intensity. Also, bins cannot be split, thus making the process difficult.

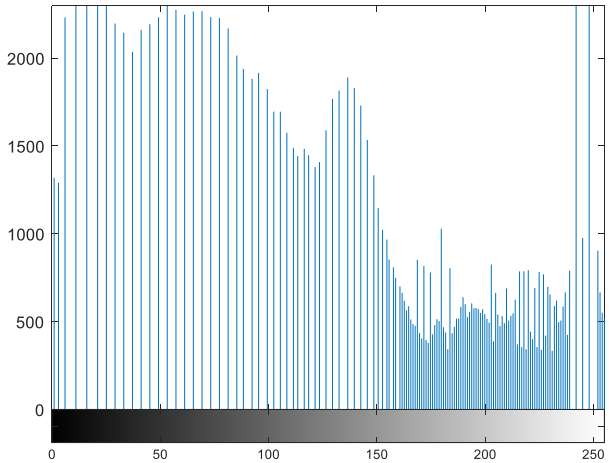
2.2c. Rerun histogram equalization on P3

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

Histogram with 10bins:



Histogram with 256 bins:



The histogram does not become more uniform and remains the almost the same. Reason is that most of the bins have gone to the correct place after the first equalization, thus there might be only a small change in bins. Also, as mentioned combining bins to flatten the histogram can be a difficult task.

2.3 Linear Spatial Filtering

2.3a. Generating filters

Generate filters with sigma = 1.0 and sigma = 2.0, both Y and X-dimensions = 5

Normalized the filters such that sum of all elements = 1.0

```
norm = -2:1:2;
[X Y] = meshgrid(norm, norm)
h1 = (1/(2*pi*1*1))*exp(-((X.^2 + Y.^2)/(2*1.^2)));
h2 = (1/(2*pi*2*2))*exp(-((X.^2 + Y.^2)/(2*2.^2)));
h1 = h1/sum(h1(:))
h2 = h2/sum(h2(:))
```

Check sum of all elements = 1.0:

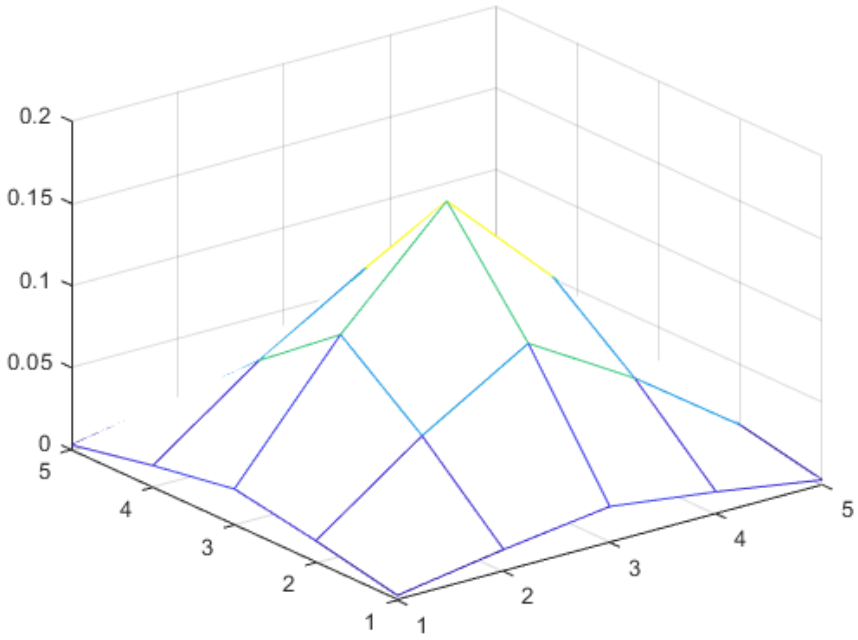
```
%check sum of all elements=1
sumh1 = sum(h1(:))
sumh2 = sum(h2(:))
```

```
sumh1 =

     1


sumh2 =

    1.0000
```

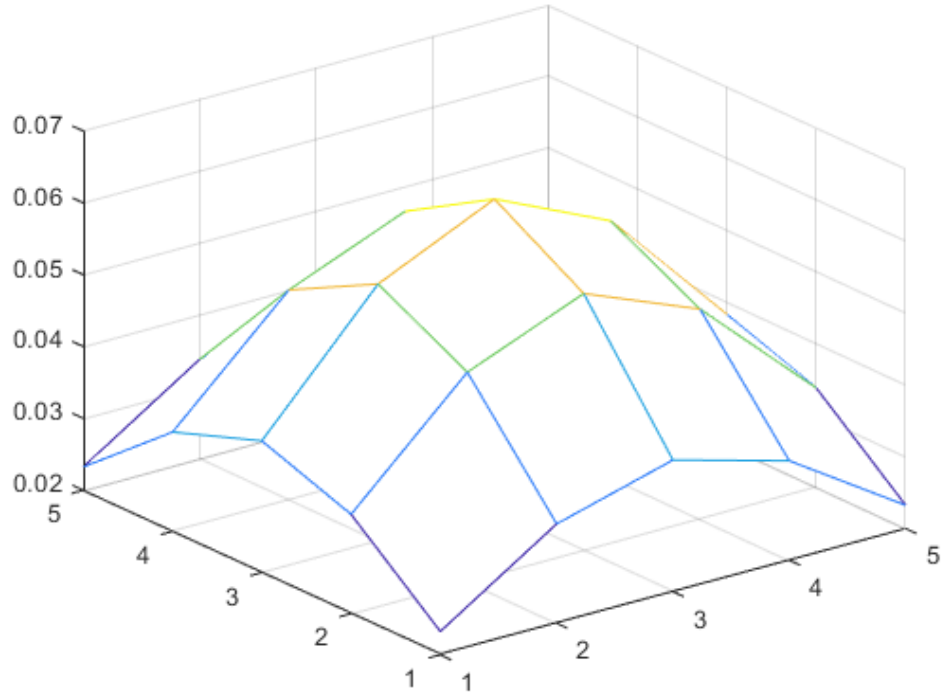
View filters as 3D graphs:

```
figure;mesh(double(h1));
figure;mesh(double(h2));
```

For sigma = 1.0:



For sigma = 2.0:



With higher sigma means higher standard deviation. Thus, the graph with sigma = 2.0, has as flat peak, as compared to the graph with sigma = 1.0.

2.3b. View ntu-gn.jpg

```
P4 = imread('ntu-gn.jpg');
figure;imshow(P4);
```

The image has additive Gaussian noise.

2.3d. Filtering the image and display the resultant images

Filtering the image with filter of sigma = 1.0:

```
%h1 filter sigma=1
P4_double = double(P4);
P4_h1_filtered = conv2(double(h1), P4_double);
P4_h1 = uint8(P4_h1_filtered);
figure;imshow(P4_h1);
```

Resultant image using filter of sigma = 1.0:

Filtering the image with filter of sigma = 2.0:

```
%h2 filter sigma=2
P4_h2_filtered = conv2(double(h2), P4_double);
P4_h2 = uint8(P4_h2_filtered);
figure;imshow(P4_h2);
```

Resultant image using filter of sigma = 2.0:

Effectiveness of Gaussian averaging filter and trade-offs:

After filtering, the additive Gaussian noise are effectively removed from the original image. However, the trade-off is that the resultant images became blurred. When sigma = 2.0, the image is more blur as compared to sigma = 1.0.

2.3d. Viewing ntu-sp.jpg

```
P5 = imread('ntu-sp.jpg');
figure;imshow(P5);
```

This image has additive speckle noise.

2.3e. Repeat 2.3c.

```
%h1 filter sigma=1
P5_double = double(P5);
P5_h1_filtered = conv2(double(h1), P5_double);
P5_h1 = uint8(P5_h1_filtered);
figure;imshow(P5_h1);

%h2 filter sigma=2
P5_h2_filtered = conv2(double(h2), P5_double);
P5_h2 = uint8(P5_h2_filtered);
figure;imshow(P5_h2);
```

Resultant image using filter of sigma = 1.0:

Resultant image using filter of sigma = 2.0:

Handling Gaussian noise vs speckle noise:

The Gaussian averaging filters are better in filtering the Gaussian noise. While the filters caused the resultant images to be blurred, the speckle noise is still visible, as compared to Gaussian noise which are effectively removed.

2.4 Median Filtering

Repeat steps 2.3b. to 2.3e. Using median filter of sizes 3x3 and 5x5

3x3 median filter on Gaussian noise:

```
%3x3 median filter on gaussian noise
P4_median = medfilt2(P4_double, [3 3]);
P4_median = uint8(P4_median);
figure;imshow(P4_median);
```

5x5 median filter on Gaussian noise:

```
%5x5 median filter on gaussian noise
P4_median = medfilt2(P4_double, [5 5]);
P4_median = uint8(P4_median);
figure;imshow(P4_median);
```

3x3 median filter on speckle noise:

```
% 3x3 median filter on speckle noise
P5_median = medfilt2(P5_double, [3 3]);
P5_median = uint8(P5_median);
figure;imshow(P5_median);
```

5x5 median filter on speckle noise:

```
% 5x5 median filter on speckle noise
P5_median = medfilt2(P5_double, [5 5]);
P5_median = uint8(P5_median);
figure;imshow(P5_median);
```

Comparing Gaussian filtering with median filtering:

For Gaussian noise, the resultant images from median filtering are clearer as compared to Gaussian filtering, however it is less effective in removing Gaussian noise.

For speckle noise, the resultant images from median filtering are clearer as compared to Gaussian filtering, also it is more effective in removing speckle noise.

Thus, Gaussian filtering is better for removing Gaussian noise, and median filtering is better for removing speckle noise.

2.5 Suppressing Noise Interference Patterns

2.5a. View pck-int.jpg

```
P6 = imread('pck-int.jpg');
figure;imshow(P6);
```

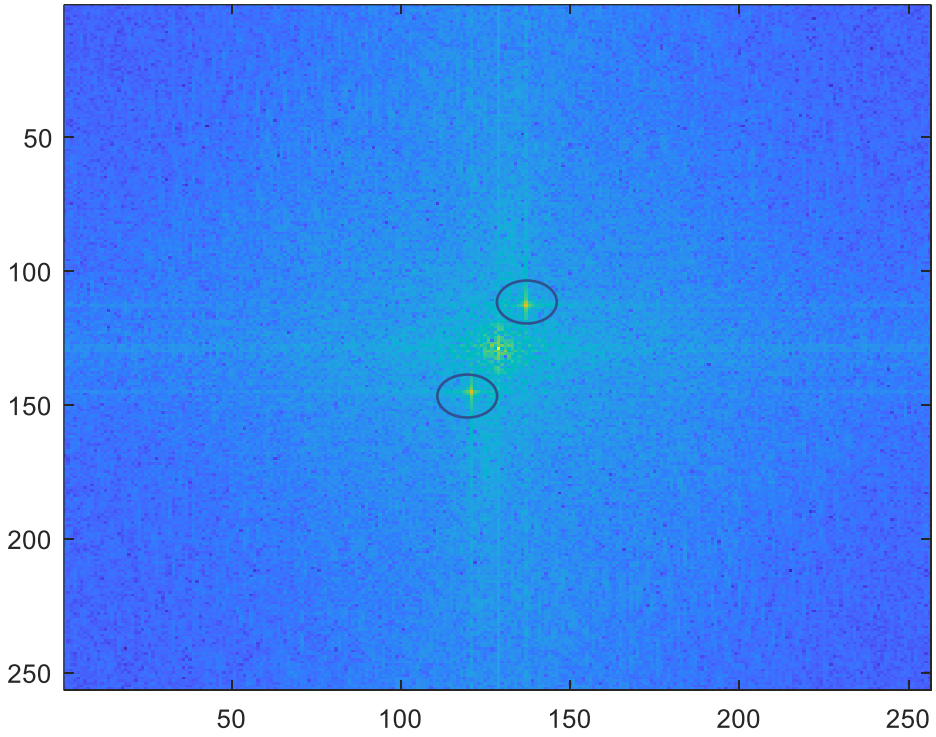
The image has diagonal lines destroying the quality of the image.

2.5b. Obtain four transform F and power spectrum S

```
%Obtain Fourier transform and power spectrum
F = fft2(P6);
S = abs(F);
```

Displaying power spectrum S

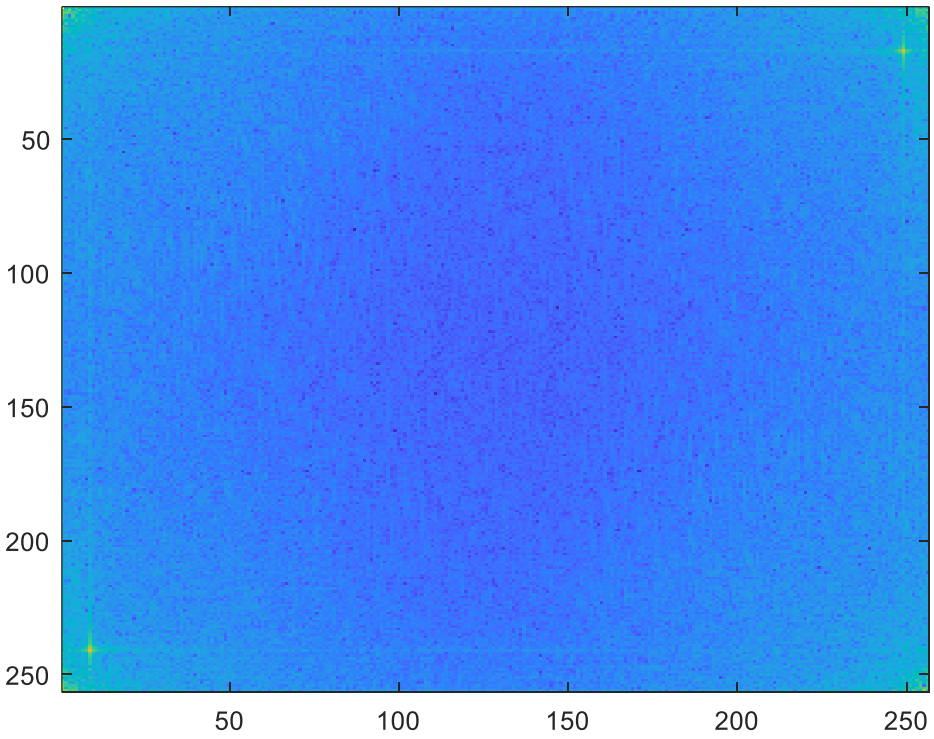
```
%Display power spectrum
imagesc(fftshift(S.^0.1));
colormap('default');
```



There are 2 distinct, symmetric frequency peaks, which corresponds to the interference pattern.

2.5c. Redisplay power spectrum without fftshift

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



Using ginput, actual locations of peaks:

```
%locations of peaks
[x y]=ginput(2)
x1 = 9;
y1 = 241;
x2 = 249;
y2 = 17;
```

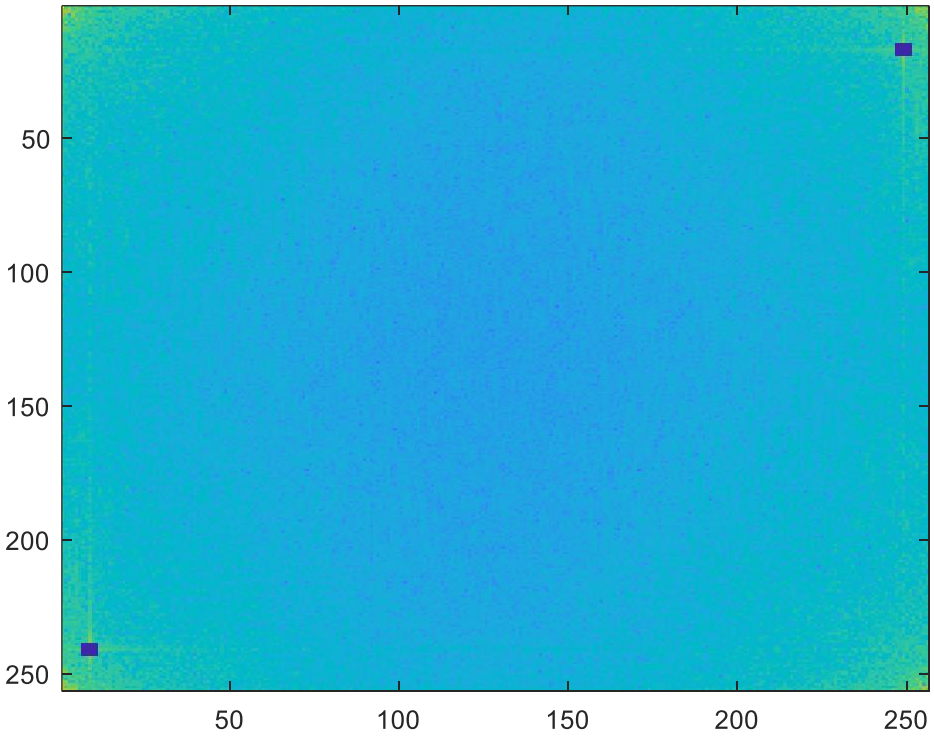
Coordinates are (9,241) and (249,17).

2.5d. Set to zero the 5x5 neighborhood elements at location corresponding to the peaks.

```
%coordinates of peaks
x1 = 9;
y1 = 241;
x2 = 249;
y2 = 17;

%set 5x5 neighbourhood of peaks to zero
F(y1-2:y1+2, x1-2:x1+2) = zeros(5);
F(y2-2:y2+2, x2-2:x2+2) = zeros(5);
```

Recompute power spectrum and display:

```
%recompute power spectrum and display
S = abs(F);
imagesc(S.^0.1);
colormap('default');
```



2.5e. Compute inverse Fourier transform and display resultant image.

```
P7 = uint8(ifft2(F));
figure;imshow(P7);
```

Comment on result:

The diagonal lines are less obvious. The peaks in the power spectrum means that there are high frequencies which causes the interference to zero. By setting the surroundings of the peaks to zero, it removes the interference patterns.

Ways to improve:

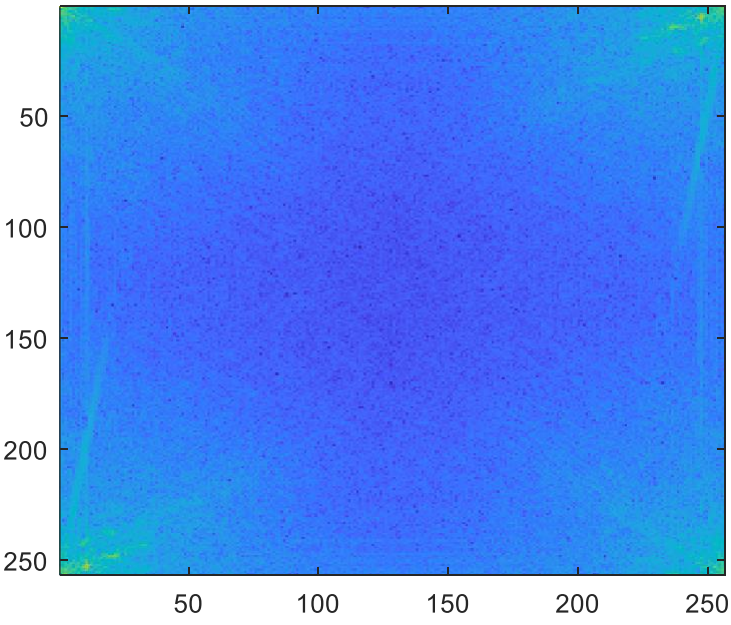
The diagonal lines are still visible. In order to improve on this, we can use a lowpass filter to remove the interference with high frequencies and pass the low frequencies. This allows a better result to be achieved.

2.5f. “Free” the primate.

Viewing the original image of primate-caged.jpg:

```
%display image
P8 = imread("primate-caged.jpg");
P8 = rgb2gray(P8);
imshow(P8);
```

Display power spectrum:



Find location of peaks:

```
%find location of peaks
[x y] = ginput(4)
x1 = 11;
y1 = 252;
x2 = 4;
y2 = 21;
x3 = 247;
y3 = 4;
x4 = 237;
y4 = 10;
```

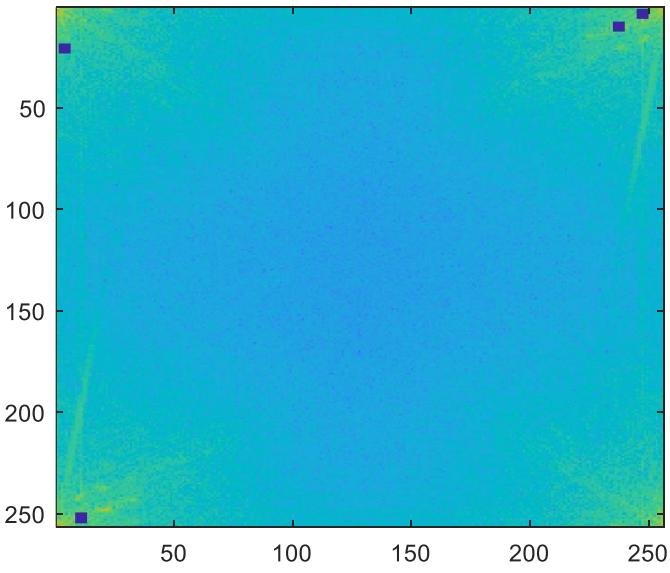
Coordinates of peaks: (11,252), (4,21), (247,4), (237,10)

Set 5x5 neighboring to zero:

```
%set 5x5 neighbouring to 0
F2(y1-2:y1+2, x1-2:x1+2) = zeros(5);
F2(y2-2:y2+2, x2-2:x2+2) = zeros(5);
F2(y3-2:y3+2, x3-2:x3+2) = zeros(5);
F2(y4-2:y4+2, x4-2:x4+2) = zeros(5);
```

Recompute power spectrum and display:

```
%recompute power spectrum and display
S = abs(F2);
imagesc(S.^0.1);
colormap('default');
```



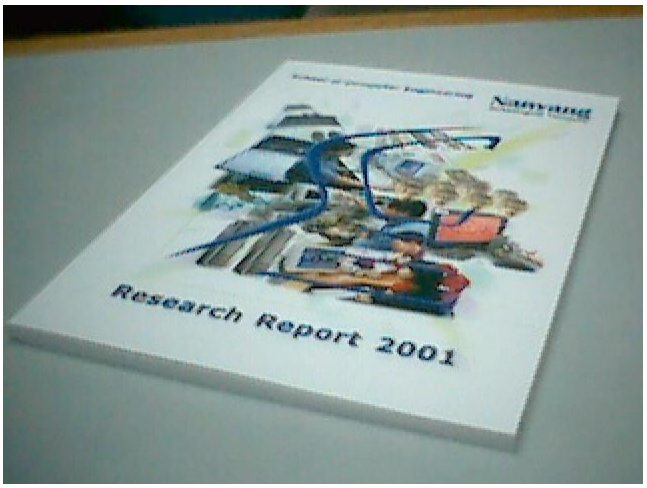
Compute inverse Fourier transform and display resultant image:

```
%compute inverse Fourier transform and display
P8 = uint8(ifft2(F2));
figure;imshow(P8);
```

2.6 Undoing Perspective Distortion of Planar Surface

2.6a. View book.jpg

```
P9 = imread('book.jpg');
figure;imshow(P9);
```



2.6b. Getting 4 corners of book.

```
figure;imshow(P9);
[x y]=ginput(4)
```

Coordinates of corners in clockwise direction, starting from top left:

(143,28), (308,47), (256, 216), (5,159)

Coordinates of desired corners in clockwise direction, starting from top left:

(0,0), (210,0), (210,297), (0,297)

2.6c. Setting up matrices for projective transformation.

Setting A and v:

```
v = [0;0;210;0;210;297;0;297];
A = [x1,y1,1,0,0,0,-x_im1*x1,-x_im1*y1;
     0,0,0,x1,y1,1,-y_im1*x1,-y_im1*y1;
     x2,y2,1,0,0,0,-x_im2*x2,-x_im2*y2;
     0,0,0,x2,y2,1,-y_im2*x2,-y_im2*y2;
     x3,y3,1,0,0,0,-x_im3*x3,-x_im3*y3;
     0,0,0,x3,y3,1,-y_im3*x3,-y_im3*y3;
     x4,y4,1,0,0,0,-x_im4*x4,-x_im4*y4;
     0,0,0,x4,y4,1,-y_im4*x4,-y_im4*y4;];
```

Find u from equation u = A^-1v:

```
u = A\v;
```

Reshape u:

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

Writing down matrix w:

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

Checking the 4 corners from transform:

```
w =

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

The 4 corners are the desired coordinates in order, (0,0), (210,0), (210,297), (0,297).

2.6d. Warp the image.

```
T = maketform('projective', U');
P2 = imtransform(P9,T,'XData',[0 210],'YData',[0 297]);
```

2.6e. View the resultant image.

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

The image became blurred, but this is what I expected. Projective transformation shows how the perceived objects change as he observer's viewpoint changes. But this causes some parts of the image to be stretched, which can cause the image to be unclear.

2.6f. Identifying pink area.

Select a point in pink area and get its RGB value:

```
%[x, y] = ginput(1)
x = 164;
y = 175;
%pixel = impixel(P2)
R = P2(:,:,1);
G = P2(:,:,2);
B = P2(:,:,3);

P2_pink = P2;
[R(x,y) G(x,y) B(x,y)]
```

```
ans =

  1×3 uint8 row vector

   201   144   135
```

R= 201, G = 144, B = 135

Keep similar RGB value in the image and set the rest to white.

```
for cha = 1:size(P2_pink, 3)
    for row = 1:size(P2_pink,1)
        for col = 1:size(P2_pink,2)
            if P2_pink(row,col,1) >= 180 & P2_pink(row,col,1) <= 230 &  P2_pink(row,col,2) <= 160 & P2_pink(row,col,3) <= 160
            else
                P2_pink(row,col,1) = 255;
                P2_pink(row,col,2) = 255;
                P2_pink(row,col,3) = 255;
            end
        end
    end
end
```

Display resultant image: