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

2.1 Contrast Stretching

The imadjust function from the Image Processing Toolbox (IPT) can be utilized to do contrast stretching. It maps a defined input intensity range, minimum to maximum pixel intensity of the original image, into a defined output intensity range, 0 to 255 {18}*. A requirement is that input and output must be normalized to be between 0 and 1 {16,17}.

Comparing the original and stretched image in fig. 1, the difference between light and dark spots are visually higher in the contrast stretched image. The histogram reflects the images, with the bright spike at a higher intensity and the low spikes at a lower intensity.

*{} indicate the line number(s) in the source code [Refer to Appendix A]

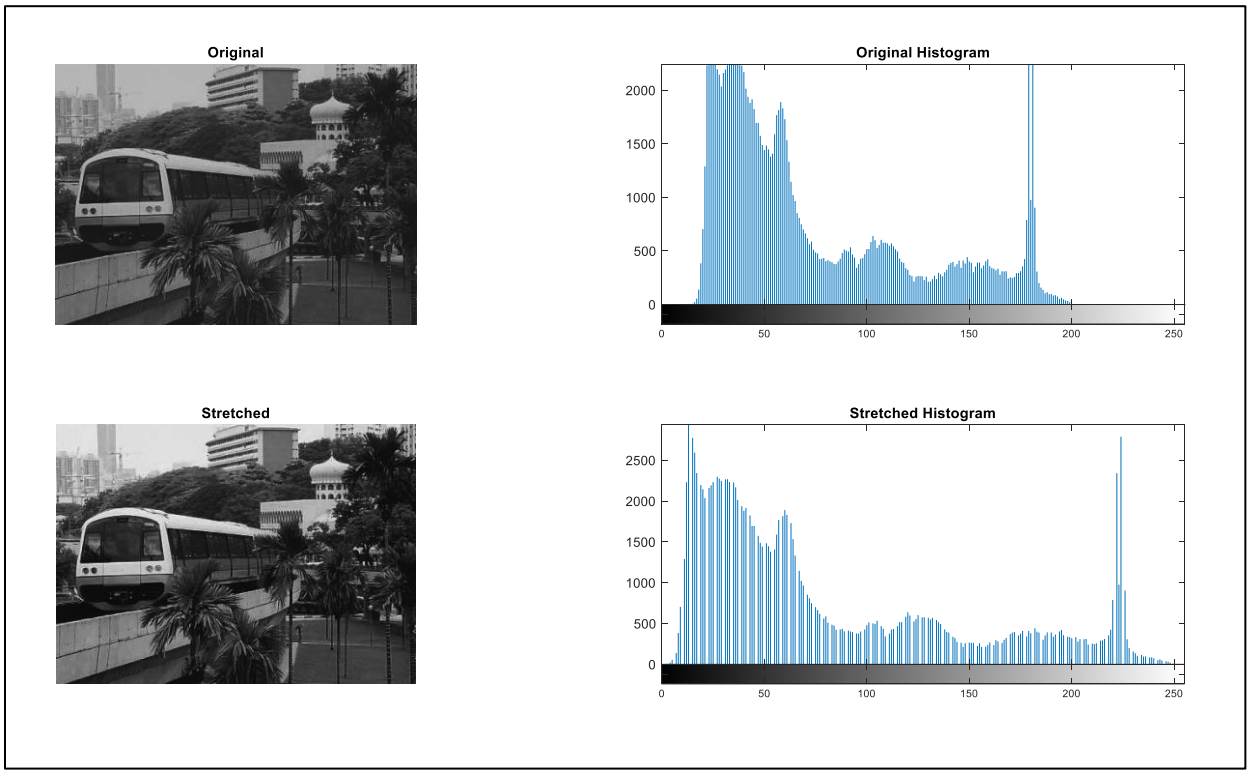

Fig. 1. Result of Contrast Stretching on the image, “mrt-train.jpg”.

2.2 Histogram Equalization

Contrast can also be enhanced using histogram equalization, spreading out the most frequent intensity values, demonstrated by the difference between “Original” and “Equalized” results in fig. 2. This can be done using the histeq function from the IPT {26}. Repeated application of histogram equalization results in no changes as it is idempotent {27}.

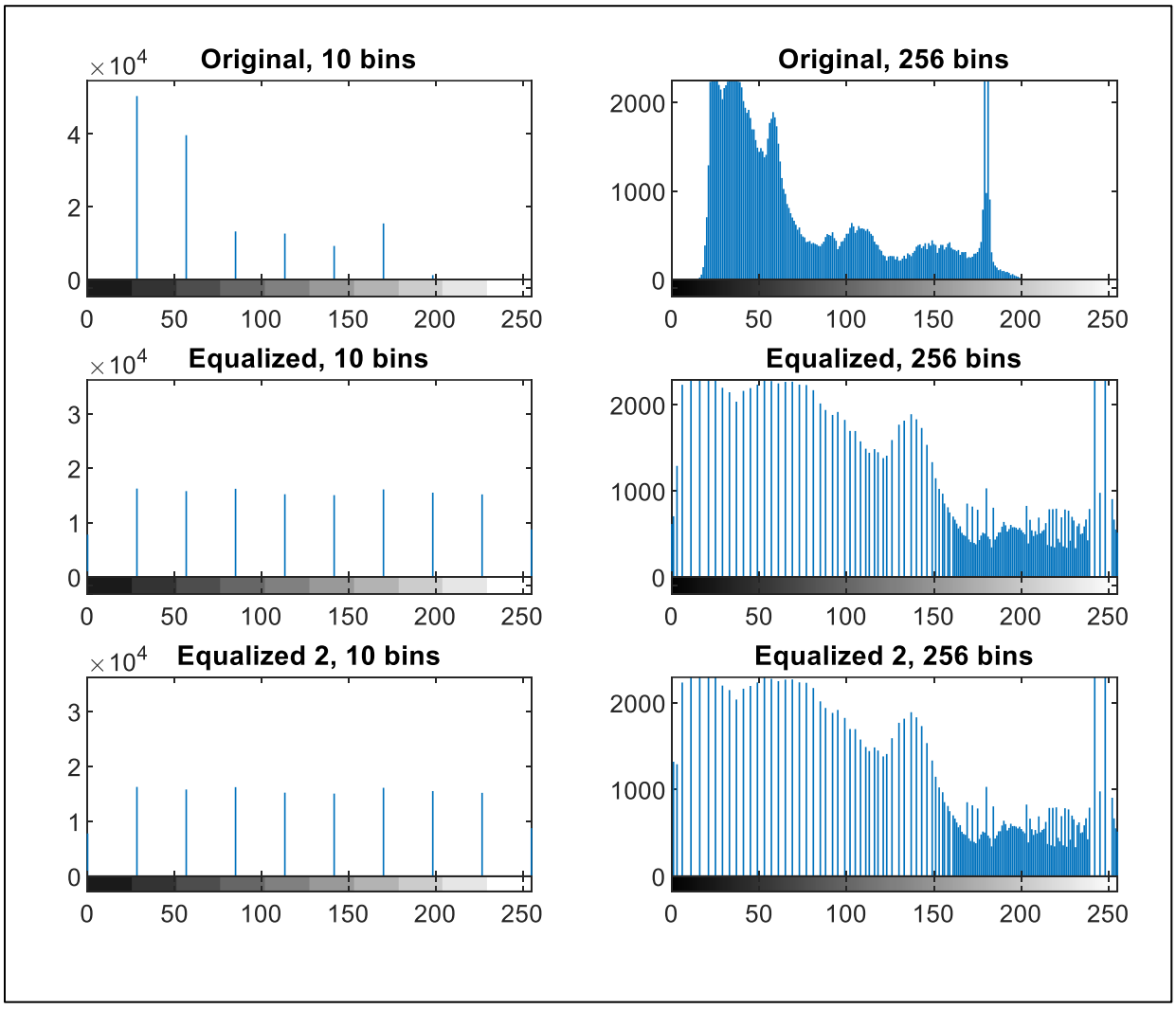

Fig. 2. Result of Histogram Equalization on the image, “mrt-train.jpg”.

2.3 Linear Spatial Filtering

Linear Spatial Filtering can be done convoluting the image with a small spatial filter in the form of a 5x5 kernel. The kernel is displayed in fig. 3.

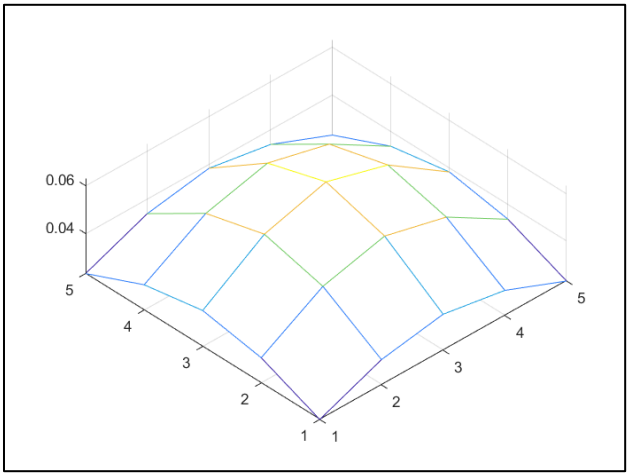

Fig. 3. 5x5 Kernel used for linear spatial filtering

As the sigma value in the filter equation, $h(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$, increases, more noise is removed both for the gaussian noise and the speckle noise. However, the contrast of the image is reduced and it becomes visually more blur as a result as shown in fig. 4.

Linear Spatial Filtering is more effective on Gaussian noise than speckle noise, this is visually noticeable for sigma=2. The Gaussian noise is mostly removed while the white speckles are still evident.

For code implementation, the kernel is created {48-55} and made into an averaging filter {56}. It is then convoluted with the images Gn and Sp {63-79}. The whole process is repeated for sigma = 2 {91-126}.

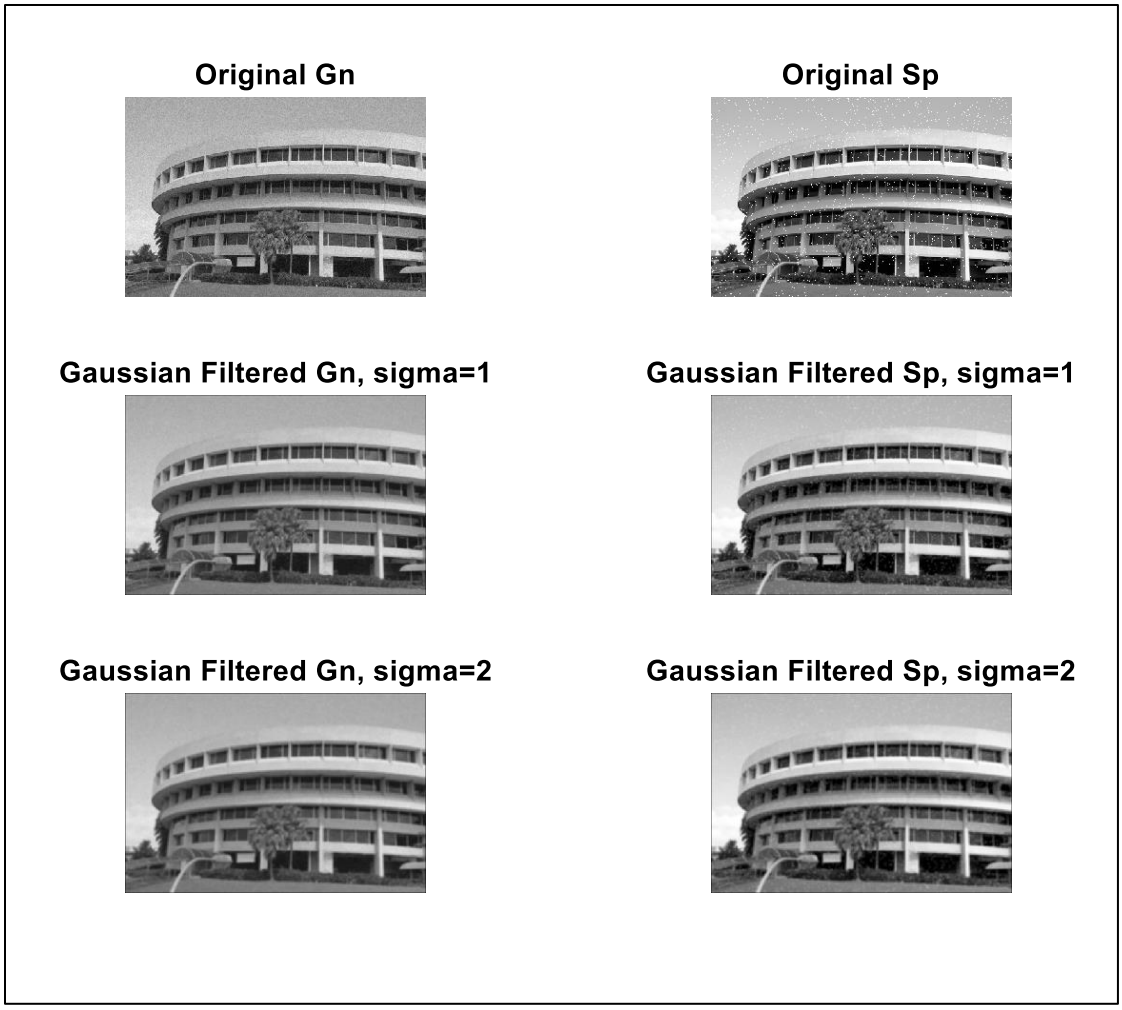

Fig. 4. Result of Linear Spatial Filtering on images with Gaussian noise, “ntu-gn.jpg”, and speckle noise, “ntu-sp.jpg”.

2.4 Median Filtering

The Median Filter, as opposed to the gaussian averaging filter, is non-linear. Comparing the results in fig. 5, we can observe the following. In terms of noise reduction, it is also much more effective at removing speckle noise, with a simple 3x3 median filter being able to completely remove speckle noise from the image. However, there is no visible difference for the image with Gaussian noise between the median filter and Gaussian averaging filter.

The advantages of the median filter over the gaussian averaging filter are its simplicity and its ability to retain edges well. The tradeoff is the higher computational time required, due to sorting. The gaussian averaging filter, in comparison, uses only addition and multiplication, resulting in lower computational time.

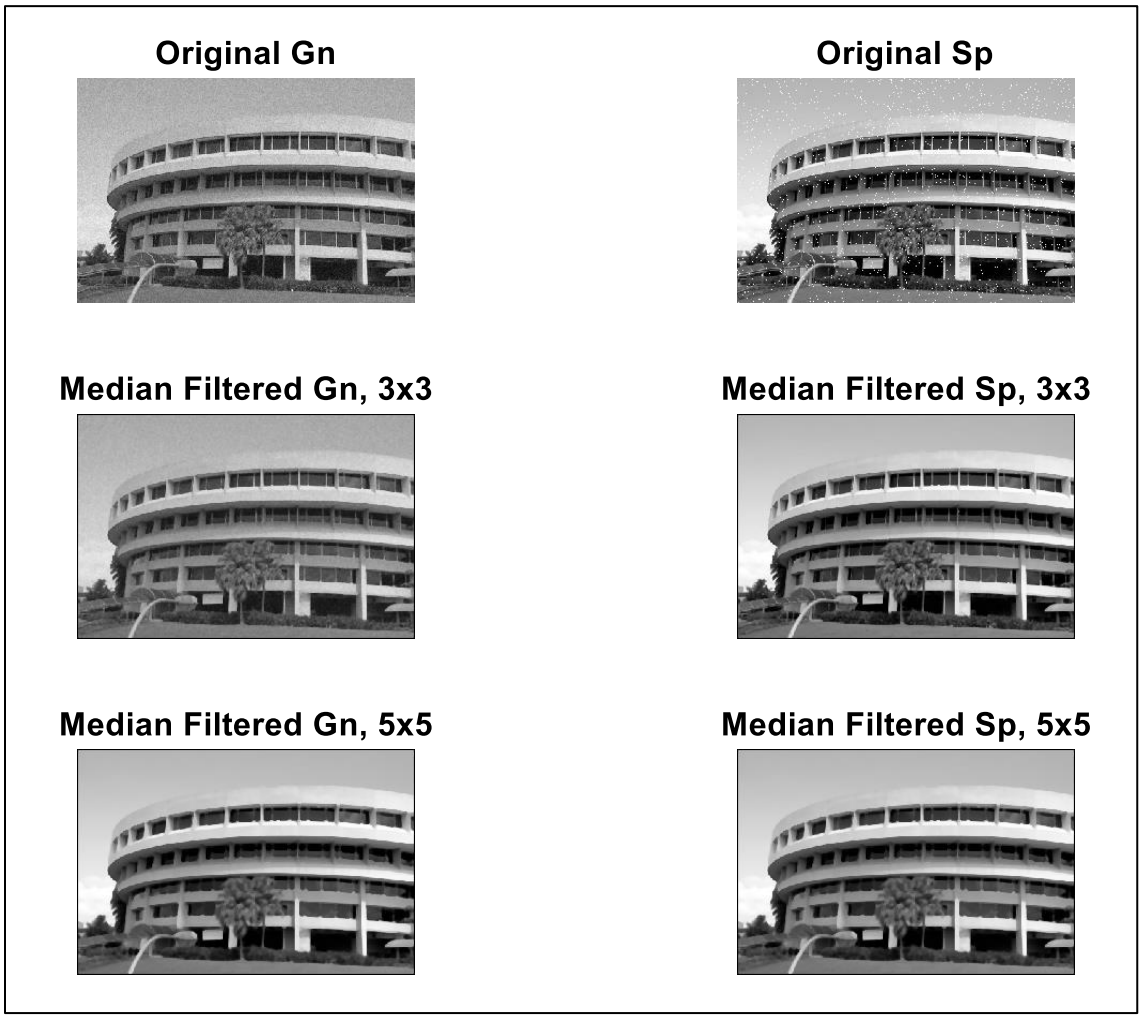

Fig. 5. Result of Median Filtering on images with Gaussian noise, “ntu-gn.jpg”, and speckle noise, “ntu-sp.jpg”.

2.5.1 Suppressing Noise Interference Patterns for “pck-int.jpg”

By editing the image in the Fourier domain, it allows for the removal of high-frequency noise while maintaining the image data that is stored at low frequency. From the “Final” image in fig. 6, most of the noise has been removed and the image beneath is much clearer. However, remnants of noise remain, especially at the edges of the image.

One possibility is due to the measurement of the noise frequency being inaccurate due to human error. The values used are solely based on human vision and decision on where the noise is {168-178}. A solution would be the segment the power spectrum into multiple frequency cubes and search for the range of noise frequencies using the intensity of yellow light, removing them. The center segments can be excluded to prevent important data from being removed.

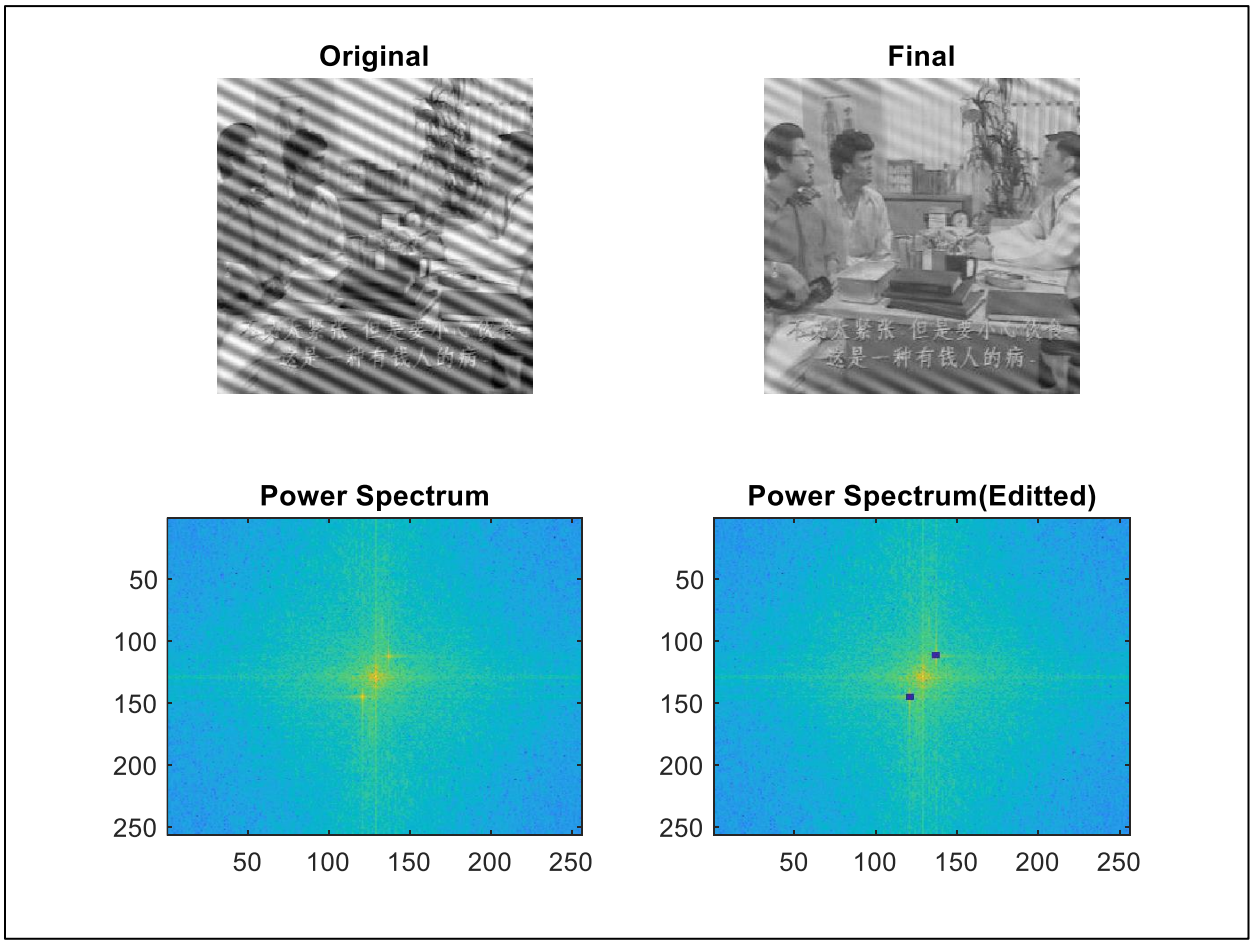

Fig. 6. Result of Noise Suppression on the image “pck-int.jpg”, and their respective power spectrum representations.

2.5.2 Suppressing Noise Interference Patterns for “primate-caged.jpg”

Applying the noise suppression technique from earlier on primate-caged.jpg, the “Final” result in fig. 7 was subpar as compared the pck-int.jpg image {191-222}. The cage remained visible despite removing the noise spots in the Fourier domain {203-213}. Increasing the size of the filtering boxes will further remove the cage. However, it will remove important data of the monkey, causing it to look blur. This is due to the frequency of the noise and the frequency of the monkey image being similar, as exhibited by the image dot and noise dots being close to each other in the “Power Spectrum”.

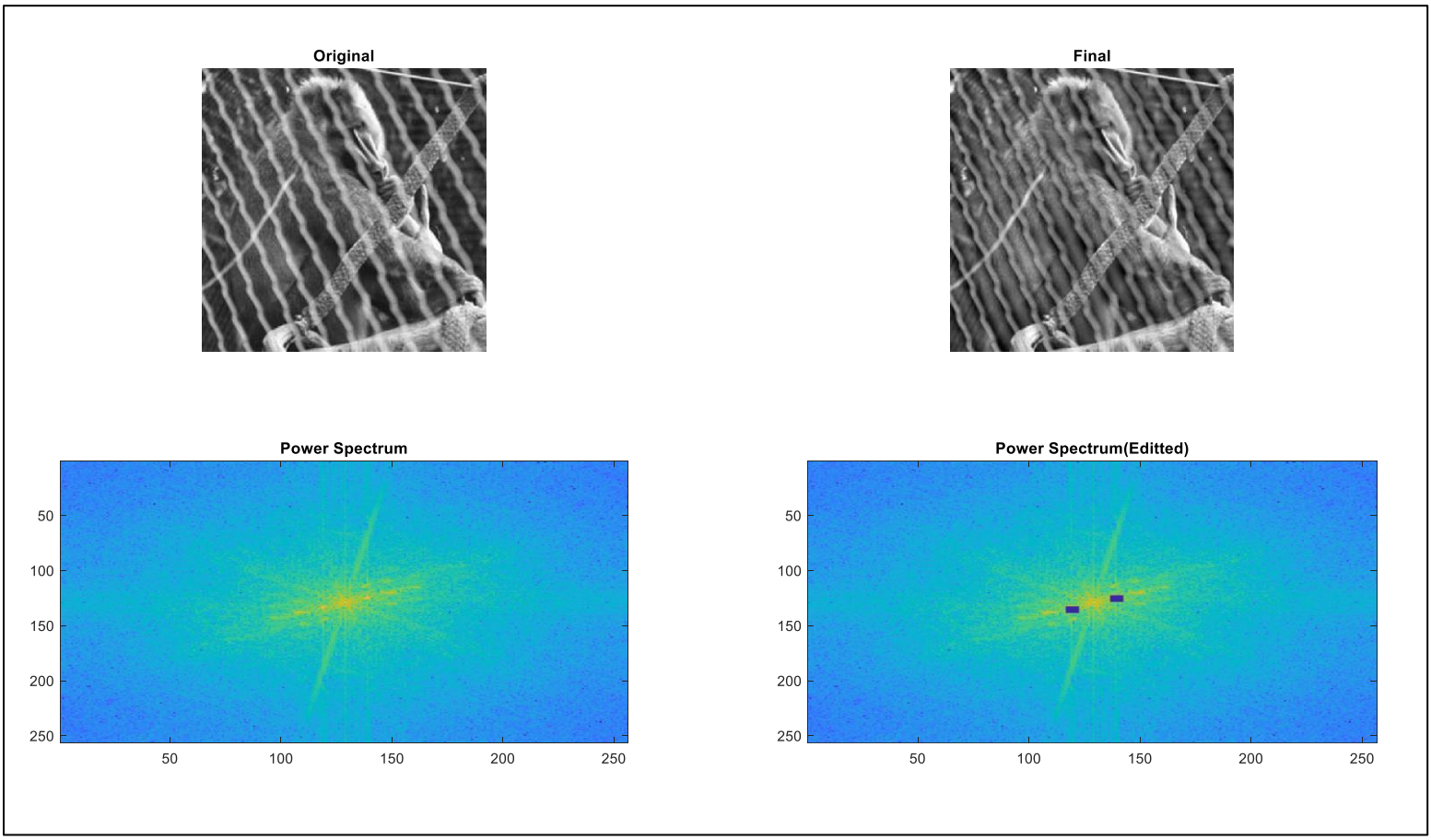

Fig. 7. Result of Noise Suppression on the image, “primate-caged.jpg”, and their respective power spectrum representations.

2.6 Undoing Perspective Distortion of Planar Surface

Undoing perspective distortion of a planar surface can be done by generating the 2D planar projective transform and applying the transform to the image.

The coordinates of each edge of the book in "book.jpg" are found using ginput {235} and placed into column and row vectors {238,239}. The four fixed points, in the base vector, represents the proposed coordinates of each edge of the book {240}. Using fitgeotrans* function, we can calculate the 3x3 projective transformation matrix, as displayed in table 1, from the initial column and row vectors, and the base vector {242-246}.

*The fitgeotrans function and the maketform function are interchangeable, but it is recommended by the mathworks documentation to use fitgeotrans.

Table 1. Projective transformation matrix

3x3 Transformation Matrix		
1.192233	-0.2642	0.000233
1.264489	2.436467	0.005419
-205.895	-30.4412	1

The matrix is then applied to the image using the imwarp* function {268}. Lastly, the book is cropped from the image for viewing purposes, using the image dimensions found previously after transformation {264-270}. From the "Transformed Image" in fig. 8, the bottom right of the book is clear and gradually gets increasingly blur towards the top left. This may be due to the resolution of the top left of the book being lower than the resolution of the bottom right of the book in the "Original" image.

*The imwarp function and the imtranform function are interchangeable, but it is recommended by the mathworks documentation to use imwarp.

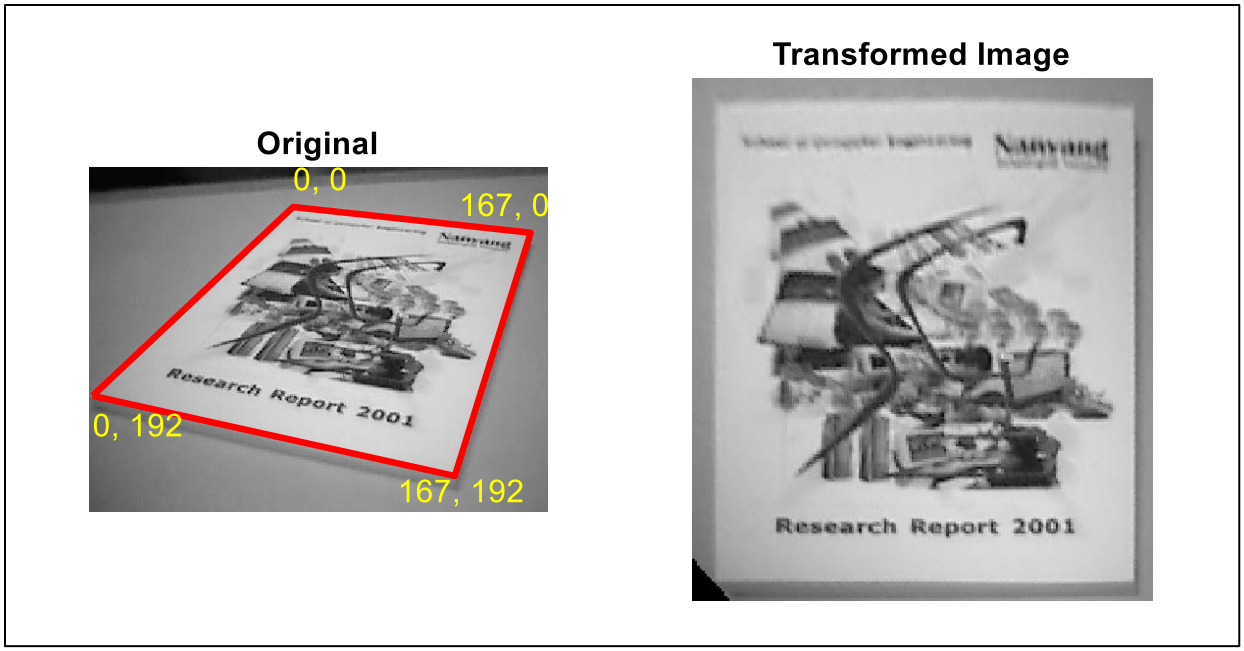

Fig. 8. Result of Undoing Perspective Distortion of Planar Surface on the image, “book.jpg”.

Appendix A: Source Code

```
clc
clear all
close all


%% Contrast Stretching

pc = imread('mrt-train.jpg');
%whos pc
 p=rgb2gray(pc);
 [m,n] = size(p);
 figure('name','Contrast Stretching')
 subplot(2,2,1),imshow(p), title('Original');
 subplot(2,2,2),imhist(p), title('Original Histogram');

 smallest = double(min(p(:)))/255;
 largest = double(max(p(:)))/255;
 s = imadjust(p,[smallest,largest],[]);

 subplot(2,2,3),imshow(uint8(s)), title('Stretched');
 subplot(2,2,4),imhist(uint8(s)), title('Stretched Histogram');


%% Histogram Equalisation

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

```
P4 = histeq(P3,255);
figure('name','Histogram Equalisation')
subplot(3,2,1),imhist(uint8(p),10), title('Original, 10 bins');
subplot(3,2,2),imhist(uint8(p)), title('Original, 256 bins');
subplot(3,2,3),imhist(uint8(P3),10), title('Equalized, 10 bins');
subplot(3,2,4),imhist(uint8(P3)), title('Equalized, 256 bins');
subplot(3,2,5),imhist(uint8(P4),10), title('Equalized 2, 10 bins');
subplot(3,2,6),imhist(uint8(P4)), title('Equalized 2, 256 bins');


%% Gaussian Averaging Filters

gn=imread('ntu-gn.jpg');
sp=imread('ntu-sp.jpg');
[m1,n1] = size(gn);
[m2,n2] = size(sp);

sigma = 1.0;

kernel = zeros(5,5);        %for 5x5 kernel
W = 0;                      %sum of elements of kernel (for normalisation)
coeff = 1/(2*pi*sigma*sigma);
for i = 1:5
    for j =1:5
        sq_dist = (i-3)^2 + (j-3)^2;
        kernel(i,j) = coeff*exp(-1*(sq_dist)/(2*sigma*sigma));
        W = W + kernel(i,j);
    end
```

```
55  end
56  kernel = kernel/W;
57
58  output1 = zeros(m1,n1);
59  output2 = zeros(m2,n2);
60  gn1 = padarray(gn,[2,2]);
61  sp1 = padarray(sp,[2,2]);
62
63  for i=1:m1
64      for j=1:n1
65          temp = gn1(i:i+4 , j:j+4);
66          temp = double(temp);
67          conv =temp.*kernel;
68          output1(i,j) = sum(conv(:));
69      end
70  end
71
72  for i=1:m2
73      for j=1:n2
74          temp = sp1(i:i+4 , j:j+4);
75          temp = double(temp);
76          conv =temp.*kernel;
77          output2(i,j) = sum(conv(:));
78      end
79  end
80
81  output1 = uint8(output1);
82  output2 = uint8(output2);
```

```
83
84  figure('name','Guassian Filtering')
85  subplot(321),imshow(gn),title('Original Gn');
86  subplot(322),imshow(sp),title('Original Sp');
87  subplot(323),imshow(output1),title('Gaussian Filtered Gn, sigma=1')
88  subplot(324),imshow(output2),title('Gaussian Filtered Sp, sigma=1')
89
90
91  sigma = 2.0;              %for sigma = 2.0
92
93  kernel = zeros(5,5);      %for 5x5 kernel
94  W = 0;                    %sum of elements of kernel (for normalisation)
95  for i = 1:5
96      for j =1:5
97          sq_dist = (i-3)^2 + (j-3)^2;
98          kernel(i,j) = coeff*exp(-1*(sq_dist)/(2*sigma*sigma));
99          W = W + kernel(i,j);
100     end
101 end
102 kernel = kernel/W;
103
104 output1 = zeros(m1,n1);
105 output2 = zeros(m2,n2);
106 gn1 = padarray(gn,[2,2]);
107 sp1 = padarray(sp,[2,2]);
108
109 for i=1:m1
110     for j=1:n1
```

```
111         temp = gn1(i:i+4 , j:j+4);
112         temp = double(temp);
113         conv =temp.*kernel;
114         output1(i,j) = sum(conv(:));
115     end
116 end
117
118 for i=1:m2
119     for j=1:n2
120         temp = sp1(i:i+4 , j:j+4);
121         temp = double(temp);
122         conv =temp.*kernel;
123         output2(i,j) = sum(conv(:));
124     end
125 end
126
127 output1 = uint8(output1);
128 output2 = uint8(output2);
129
130 subplot(325),imshow(output1),title('Gaussian Filtered Gn, sigma=2')
131 subplot(326),imshow(output2),title('Gaussian Filtered Sp, sigma=2')
132
133 figure('name','Kernel')
134 mesh(kernel);
135
136 %% median filtering
137
138 gn2 = padarray(gn,[1,1]);
```

```
sp2 = padarray(sp,[1,1]);

gn_median_3 = medfilt2(gn2,[3,3]);
sp_median_3 = medfilt2(sp2,[3,3]);

gn3 = padarray(gn,[2,2]);
sp3 = padarray(sp,[2,2]);

gn_median_5 = medfilt2(sp2,[5,5]);
sp_median_5 = medfilt2(sp2,[5,5]);

figure('name','Median Filtering')
subplot(321),imshow(gn),title('Original Gn');
subplot(322),imshow(sp),title('Original Sp');
subplot(323),imshow(gn_median_3),title('Median Filtered Gn, 3x3')
subplot(324),imshow(sp_median_3),title('Median Filtered Sp, 3x3')
subplot(325),imshow(gn_median_5),title('Median Filtered Gn, 5x5')
subplot(326),imshow(sp_median_5),title('Median Filtered Sp, 5x5')


%% Suppressing Noise Interference Patterns 1

pck=imread('pck-int.jpg');
[m3,n3] = size(pck);

F=fft2(pck);
S=log10(abs(fftshift(F)).^2);
S1=log10(abs(F).^2);
```

```
167
168 copy = zeros(m3,n3);
169
170 for i=1:m3
171     for j=1:n3
172         copy(i,j) = F(i,j);
173         if (i>=238 && i<=242) && (j>=7 && j<=11)
174            copy(i,j)=0;
175         end
176         if (i>=15 && i<=19) && (j>=247 && j<=251)
177             copy(i,j)=0;
178         end
179     end
180 end
181
182 S2=log10(abs(fftshift(copy)).^2);
183 iF=uint8(real(ifft2(copy)));
184
185 figure('name', 'PCK TV');
186 subplot(221),imshow(pck), title('Original');
187 subplot(222),imshow(iF), title('Final');
188 subplot(223),imagesc(S), title('Power Spectrum');
189 subplot(224),imagesc(S2), title('Power Spectrum(Editted)');
190
191 %% Suppressing Noise Interference Patterns 2
192
193 prim=imread('primate-caged.jpg');
194 prim=rgb2gray(prim);
```

```
[m4,n4] = size(prim);

F1=fft2(prim);
G=log10(abs(fftshift(F1)).^2);
G1=log10(abs(F1).^2);

copy1 = zeros(m4,n4);

for i=1:m4
    for j=1:n4
        copy1(i,j) = F1(i,j);
        if (i>=251 && i<=256) && (j>=9 && j<=14)
           copy1(i,j)=0;
        end
        if (i>=5 && i<=10) && (j>=245 && j<=250)
            copy1(i,j)=0;
        end
    end
end

G2=log10(abs(fftshift(copy1)).^2);
iF2=uint8(real(ifft2(copy1)));

figure('name', 'Caged Primate');
subplot(221),imshow(prim), title('Original');
subplot(222),imshow(iF2), title('Final');
subplot(223),imagesc(G), title('Power Spectrum');
subplot(224),imagesc(G2), title('Power Spectrum(Editted)');
```

```
223
224
225  %% Undoing Perspective Distortion of Planar Surface
226
227  img = im2double(rgb2gray(imread('book.jpg')));
228  msgid = 'Images:InitSize:adjustingMag'; %define warning msg
229  name = 'check2';
230
231  warning('off',msgid); %ignore warning message
232  figure('name', 'Image Transformation');
233  subplot(121),imshow(img),title('Original')
234
235  %[X Y] = ginput(4)
236
237  %column vector and row vectors for corners
238  c = [3 256 309 143]';
239  r = [160 216 46 28]';
240  base = [0 192; 167 192; 167 0; 0 0]; % fixed points
241
242  tf = fitgeotrans([c r],base,'projective'); %save geometric transformation based on vectors
243  disp('tf = ');
244  disp(tf)
245
246  T = tf.T; %transformation matrix
247  disp('T =');
248  format short g
249  disp(T);
250
```

```
%Overlay for original image to show red border and yellow labels
hold on;
plot([c;c(1)],[r;r(1)],'r','Linewidth',2);
text(c(1),r(1)+20,'0, 192','Color','y');
text(c(2)-40,r(2)+10,'167, 192','Color','y');
text(c(3)-50,r(3)-20,'167, 0','Color','y');
text(c(4),r(4)-20,'0, 0','Color','y');
hold off;

[xf1, xf1_ref] = imwarp(img,tf); %apply geomatric transform onto image
%subplot(122),imshow(xf1),title('Transformed Image')
xf1_ref

%Crop image
xf1_ref.XWorldLimits = [-10 175];
xf1_ref.YWorldLimits = [-10 200];
xf1_ref.ImageSize = [210 185];
[xf2 xf2_ref] = imwarp(img,tf,'OutputView',xf1_ref);
xf2_ref
subplot(122), imshow(xf2),title('Transformed Image');
```

Lab_1.m