CZ4003:
Computer Vision

Lab 1:

Point Processing + Spatial Filtering + Frequency Filtering + Imaging Geometry

by
Chulpaibul Jiraporn (U1822666H)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

Table of Contents

Experiment 1: Contrast Stretching

Code

```
pc = imread('mrt-train.jpg');
whos pc;
p = rgb2gray(pc);
imshow(p);

min_p = min(p(:))    % min = 13
max_p = max(p(:))    % max = 204

p1(:,:) = imsubtract(p(:,:), double(min_p));
p2(:,:) = immultiply(p1(:,:), double(255/double(max_p-min_p)));

min_p2 = min(p2(:))    % min = 0
max_p2 = max(p2(:))    % max = 255

imshow(p2);
% display original and enhanced image (Fig. 1)
subplot(1,2,1); imagesc(p); axis off; title('Original Image');
subplot(1,2,2); imagesc(p2); axis off; title('New Enhanced Image');
colormap(gray);
```

Result

Fig. 1

Discussion

In this experiment, contrast stretching is performed using point processing technique. The point processing technique changes the gray-level intensity of each pixel based on the global statistics maximum and minimum. Fig. 1 shows an original image of mrt train on the left and the enhanced image after contrast stretching on the right. The original image(left) has maximum and minimum gray-level intensity of 204 and 13 respectively. After linear scaling of gray-level, the new image(right) has maximum and minimum gray-level intensity of 0 and 255 respectively. As a result, the new image has greater contrast which can be observed clearly as the bright part of the mrt train now becomes brighter.

Linear scaling of gray-level:

$$g(x,y) = 255 \times \frac{f(x,y) - min}{\max - min}$$

where min and max are global minimum and maximum gray level of the original image respectively.

Experiment 2: Histogram Equalization

Code

```
imhist(p, 10)

p3 = histeq(p, 255);
p4 = histeq(p3, 255);

size(pc)

% display histograms (Fig.2)
subplot(2,2,1); imhist(p, 10); axis on; title('Before Histogram
Equalization (10 bins)');
subplot(2,2,2); imhist(p, 256); axis on; title('Before Histogram
Equalization (256 bins)');
subplot(2,2,3); imhist(p3, 10); axis on; title('After Histogram
Equalization (10 bins)');
subplot(2,2,4); imhist(p3, 256); axis on; title('After Histogram
Equalization (256 bins)');

% display images and histograms together (Fig.3)
subplot(2,3,1); imagesc(p); axis off;  title('Original Image');
subplot(2,3,2); imagesc(p2); axis off;  title('Contrast Stretching');
subplot(2,3,3); imagesc(p3); axis off;  title('Histogram Equalization');
subplot(2,3,4); imhist(p, 256); axis on; title('Original Histogram');
subplot(2,3,5); imhist(p2, 256); axis on; title('Contrast Stretching
Histogram');
subplot(2,3,6); imhist(p3, 256); axis on; title('Histogram Equalization
Histogram');
colormap(gray);
```

Result

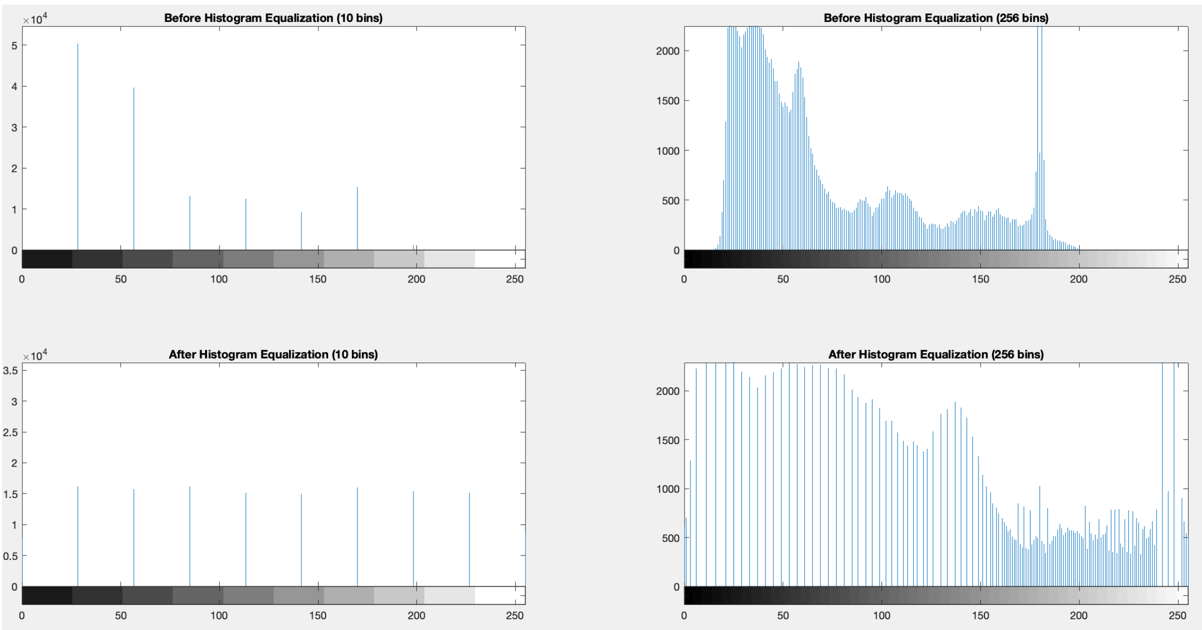

Fig. 2

Discussion

In this experiment histogram equalization technique is applied to the original mrt-train image. In Fig. 2 below, the images on the first row illustrated the gray-level histogram of the original image. The difference between 10-bin-histogram and 255-bin-histogram is that we can only

see that most pixels intensity are low from the 10-bin-histogram but we can observe that the uneven distribution of gray-level where most of the pixels have gray-level intensity between 25 and 75. Histogram equalization is used to flatten the histogram of gray-level by assigning each pixel to a new gray-level(bin). Since the 2D-dimension of the original image is 320 by 443, each bin supposed to have around 556 pixels ideally. The images on the second row show the new gray-level histogram after histogram equalization is performed. It is evident that the new distribution of gray-level is more uniformly distributed than before where the number of pixels in each bins of the 10-bin-histogram (left) are similar and the gray-levels ranges from around 0 to 255 in the new 256-bins-histogram (right). With reference to the new 256-bins-histogram, each bin with gray-level from 160 to 240 has around 556 pixels which ideal number of pixels in each bin. However, the number of pixels in bins with gray-level lower than 160 or higher than 240 significantly is much higher than 556. This is because pixels with same gray-level will always be mapped to the same new gray-level in histogram equalization. As a result, we can see that the maximum height of the 256-bins-histogram remains the same but the bins with high number of pixels are more spread out.
After the rerun of histogram equalization, the histograms remain the same as the histograms on the second row of Fig.2 and it does not become more uniform. This is because after the first-time histogram equalization is performed, each pixel is assigned to a new bin where it belongs to. Therefore, in subsequent histogram equalization, each pixel will be assigned to the same bin and there will not be any further changes in the histogram.

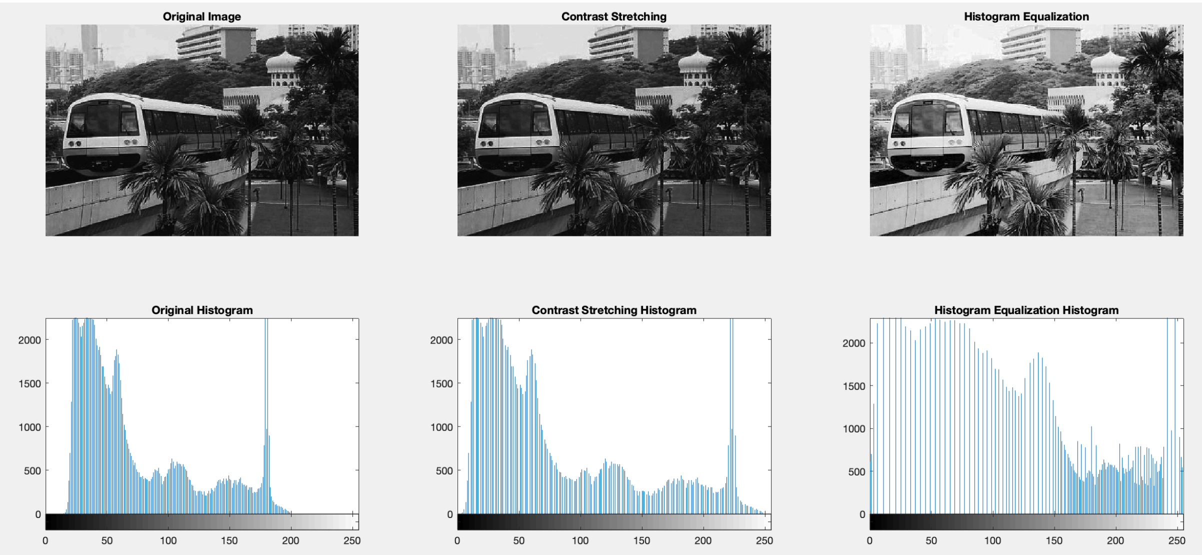

Fig. 3

Fig. 3 shows the original image and enhanced image as a result of contrast stretching and histogram equalization respectively. We can see that the enhanced image from histogram equalization is significantly brighter than that from contrast stretching. This is because unlike linearly scaling gray-level in contrast stretching, histogram equalization non-linearly mapped a gray-level to a new gray level which results in combination of high-intensity bins into a new bin, causing the low intensity bins to spread out.

Experiment 3: Linear Spatial Filtering

Code

```
x = -2:2;
y = -2:2;

[X,Y] = meshgrid(x,y)
sigma = 1;
h1 = exp(-((X.^2 + Y.^2)/(2*sigma^2)))/(2*pi*sigma^2)
h1_norm = h1./sum(h1(:))

sigma = 2;
h2 = exp(-((X.^2 + Y.^2)/(2*sigma^2)))/(2*pi*sigma^2)
h2_norm = h2./sum(h2(:))

% display Gaussian filters (Fig.2)
subplot(1,2,1); mesh(h1_norm); axis on; title('σ = 1');
subplot(1,2,2); mesh(h2_norm); axis on; title('σ = 2');

%% ntu-gn
n = imread('ntu-gn.jpg');
n1 = conv2(n,h1);
n2 = conv2(n,h2);

% display ntu-gn results (Fig.5 first row)
subplot(2,3,1); imagesc(n) ; axis off; title('NTU Gaussian Noise');
subplot(2,3,2); imagesc(n1) ; axis off; title('NTU Gaussian Noise (σ =
1)');
subplot(2,3,3); imagesc(n2) ; axis off; title('NTU Gaussian Noise (σ =
2)');
colormap(gray);

%% ntu-sp
m = imread('ntu-sp.jpg');
m1 = conv2(m,h1);
m2 = conv2(m,h2);

% display ntu-sp results (Fig.5 last row)
subplot(2,3,4); imagesc(m) ; axis off; title('NTU Speckle Noise');
subplot(2,3,5); imagesc(m1) ; axis off; title('NTU Speckle Noise (σ = 1)');
subplot(2,3,6); imagesc(m2) ; axis off; title('NTU Speckle Noise (σ = 2)');
colormap(gray);
```

Result

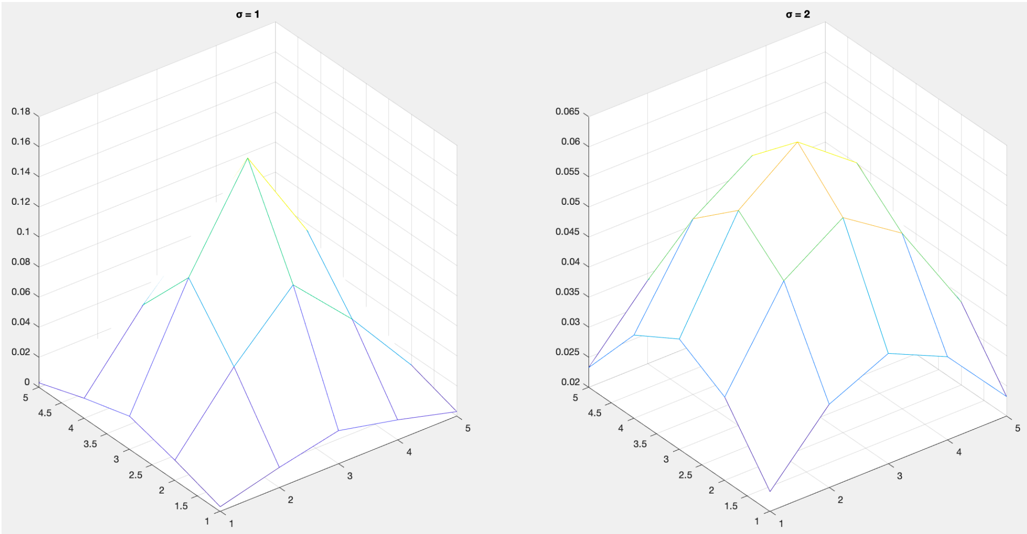

Fig. 4

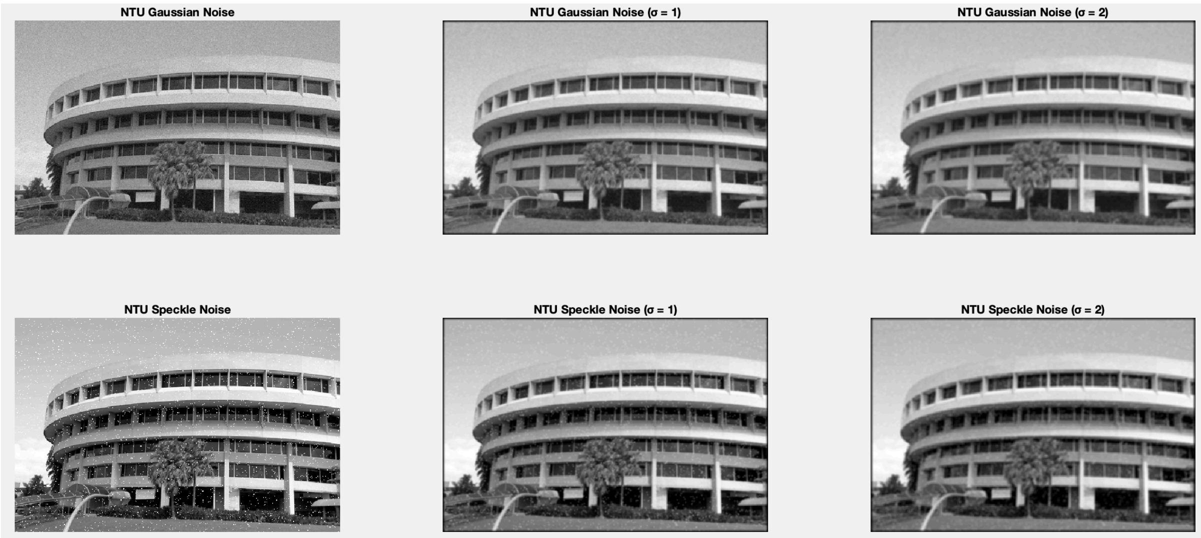

Fig. 5

Discussion

Fig. 4 shows the two 5x5 Gaussian filters with σ = 1 and σ = 2 respectively. Filter with σ = 2 are more spread out, but the peak is lower than that of σ = 1 because standard deviation represents the spread of distribution and the filters are normalized to 1. Gaussian filter has a shape of 2D normal density function. Gaussian filter computes each pixel's new gray-level based on the existing gray-levels of itself and neighboring pixels and nearer neighbors have higher weights. Hence, it can be used to reduce additive gaussian noise which has a mean of zero and causes the intensity of pixels to deviates slightly from the actual intensity. This can be illustrated in the first row of Fig. 5 where gaussian filters are convoluted with image with gaussian noise. We can see that the resulting images appear smoother and the noise is less visible as σ increases. However, gaussian filter assumes that neighboring pixels are similar which is not true at the edges of the objects in the image. The new gray-level edgel of an object is computed based on pixels which belong to another object in the image, resulting in enhanced image appears to be blurred. As σ increases, the weights of farther neighbors increase, worsening the blurriness as more detail(edge) is lost. Hence, there is a trade-off between the sharpness and noise reduction of enhanced image using the two filters.

Gaussian filter is not effective in reducing speckle noise as compared to gaussian noise. As illustrated in the second row of Fig.5, the speckle noise is still visible in the enhanced images.

Experiment 4: Median Filtering

Code

```
n = imread('ntu-gn.jpg');
n1 = medfilt2(n,[3,3]);
n2 = medfilt2(n,[5,5]);

% display ntu-gn results (Fig.6 first row)
subplot(2,3,1); imagesc(n) ; axis off; title('NTU Gaussian Noise');
subplot(2,3,2); imagesc(n1) ; axis off; title('NTU Gaussian Noise (3 x
3)');
subplot(2,3,3); imagesc(n2) ; axis off; title('NTU Gaussian Noise (5 x
5)');
colormap(gray);

m = imread('ntu-sp.jpg');
m1 = medfilt2(m,[3,3]);
m2 = medfilt2(m,[5,5]);

% display ntu-sp results (Fig.6 last row)
subplot(2,3,4); imagesc(m) ; axis off; title('NTU Speckle Noise');
subplot(2,3,5); imagesc(m1) ; axis off; title('NTU Speckle Noise (3 x 3)');
subplot(2,3,6); imagesc(m2) ; axis off; title('NTU Speckle Noise (5 x 5)');
colormap(gray);
```

Result

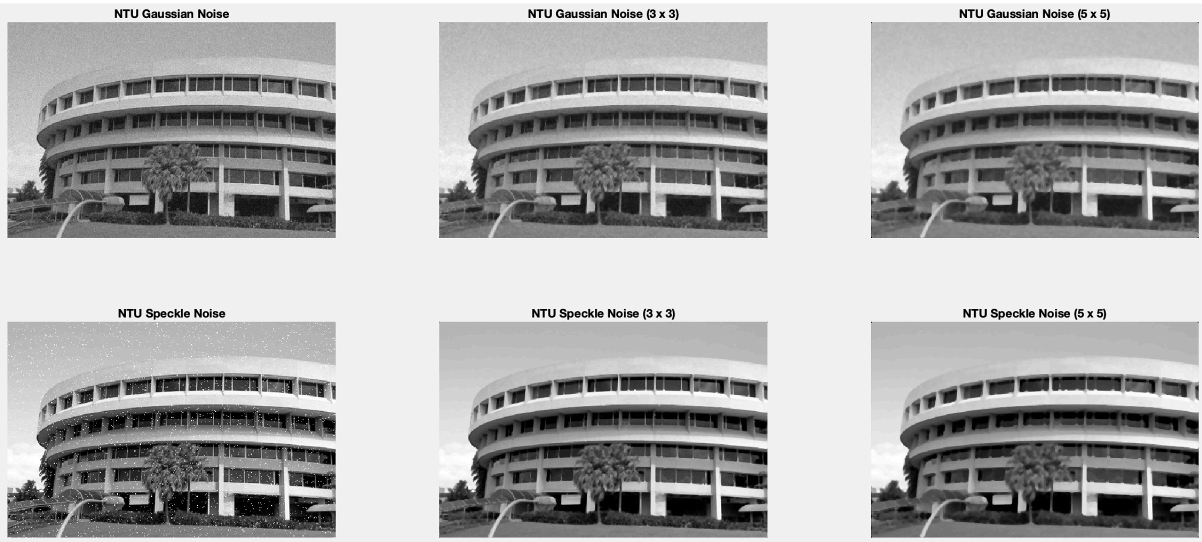

Fig. 6

Discussion

Median filter sorts the gray-level of neighbors and the pixel and assign the median gray-level to the pixel. As illustrated in the second row of Fig. 6, median filtering is very effective in

removing speckle noise, the enhanced images are very smooth, and the edges are preserved. This is because speckle noise has extreme intensity, close to 0 or 255, and usually will not be mapped to a new gray-level. Thus, the speckle noise is replaced with the gray-level of neighbor. The edge is preserved because the gray-level of edgel is usually the median gray-level in the local area. The enhanced image of 5x5 median filtering is more blurred compared to that of 3x3. This is because as the filter size increases, the number of neighbors increases and gray-level of other pixels belonging to other objects in the image are more likely to be included in the sorting process and selected as the median.
Median filtering is less effective in reducing additive gaussian noise because the noise affects the actual gray-level slightly and it can still be selected as the median. This can be observed as the enhanced in the second row is more smoother and shaper than that of the first row.

Experiment 5: Suppressing Noise Interference Patterns

Code

```
f = imread('pck-int.jpg');

F = fft2(f);
S = abs(F);
Ss = fftshift(S.^0.1);      %non-linearly scale the power

% display power spectrum of original image (Fig. 7)
subplot(1,2,1); imagesc(S.^0.1) ; axis off; title('Power Spectrum of
original image (without fftshift)');
subplot(1,2,2); imagesc(Ss) ; axis off; title('Power Spectrum of original
image (with fftshift)');
colormap('default');
```

Results

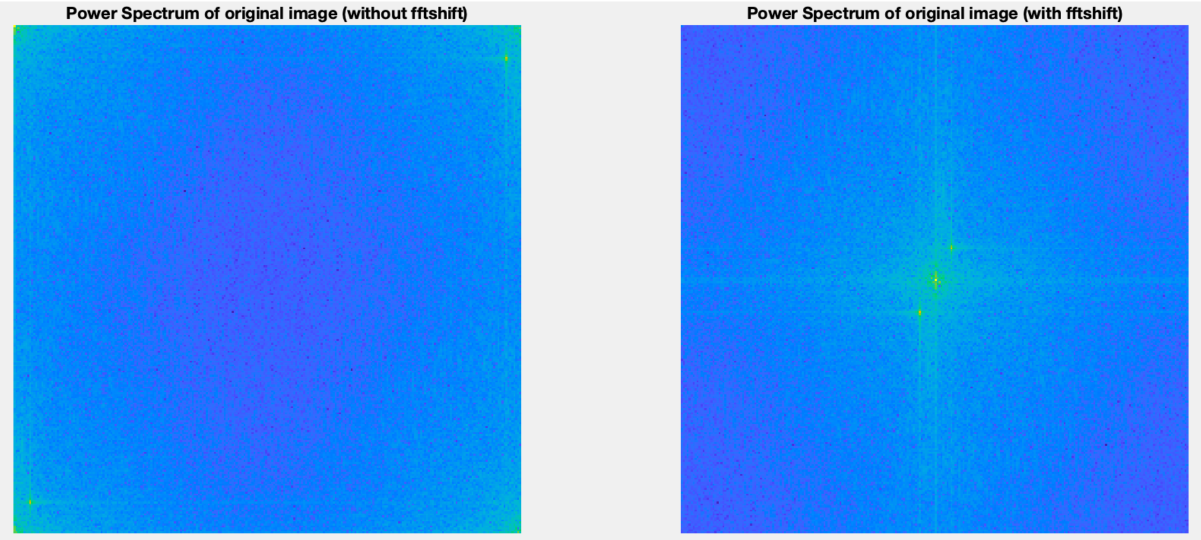

Fig. 7

Code

```
x1 = 241; y1 = 9;
x2 = 17; y2 = 249;
```

```
F_new = F;
F_new(x1-2:x1+2, y1-2:y1+2) = 0;
F_new(x2-2:x2+2, y2-2:y2+2) = 0;
S_new = abs(F_new);
Ss_new = fftshift(S_new.^0.1);


% display power spectrum of enhanced image (Fig. 10)
subplot(1,2,1); imagesc(S_new.^0.1); axis off; title('Power Spectrum of
enhanced image (without fftshift)');
subplot(1,2,2); imagesc(Ss_new) ; axis off; title('Power Spectrum of
enhanced image (with fftshift)');
colormap('default');

f_new = ifft2(F_new);
% display original and enhanced image (Fig. 11)
subplot(1,2,1); imagesc(f) ; axis off; title('Original
image');colormap(gray);
subplot(1,2,2); imagesc(f_new) ; axis off; title('Enhanced
image');colormap(gray);
```

Result

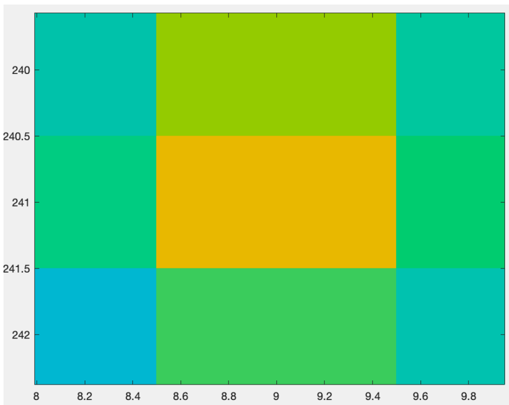

Fig. 8

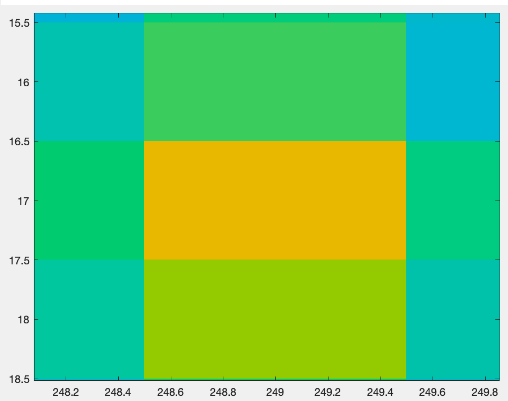

Fig. 9

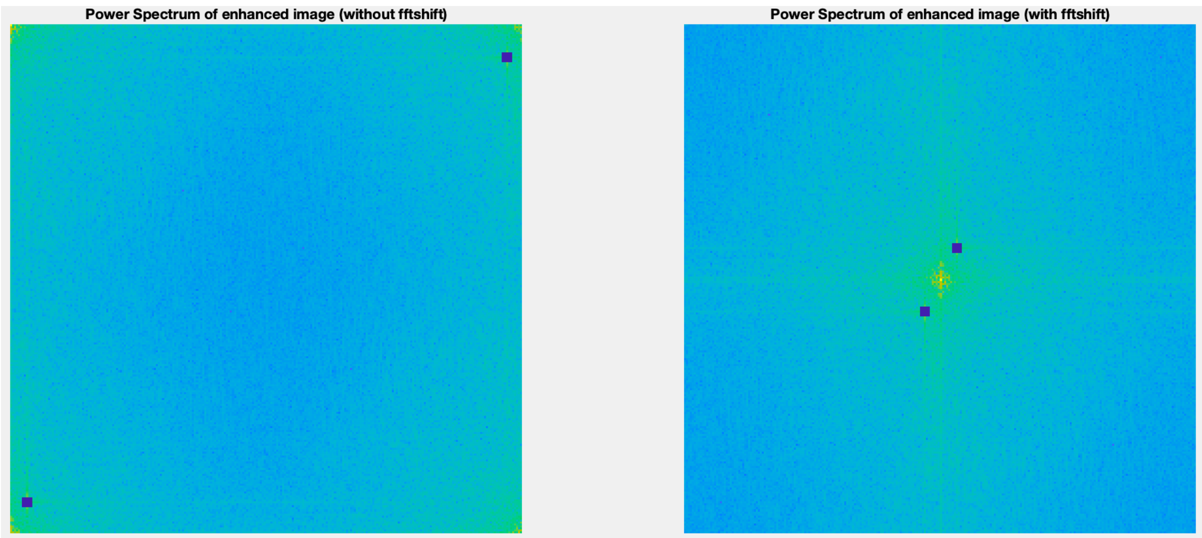

Fig. 10

Fig. 11

Discussion

The two frequency peaks represent interference pattern. By setting these frequencies and the frequencies around in the Fourier (frequency) domain, noise which corresponds to these frequencies is removed. An enhanced image is obtained as shown in Fig. 11. However, there is still interference pattern caused by other frequencies. With reference to Fig. 7, we can observe that there are less visible peaks around the two peaks (x=241, y=9 and x=17, y=249). These less visible peaks also extend along the line x = 241, x = 17, y = 9 and y =249. Thus, one way to remove noise further is by setting these less visible peaks to zero. Below shows a further enhanced image obtained by setting to 7x7 neighbor frequency elements of the 2 peaks and frequencies along x = 241, x = 17, y = 9 and y =249 to zero. Fig. 12 and Fig. 13 show the resulted power spectrum of the further enhanced image and the enhanced image respectively.

Fig. 14 shows the original, enhanced and further enhanced image respectively. We can observe that interference pattern is removed significantly in the further enhanced image while keeping the signal of the image. The details of the further enhanced image is slightly removed compared to the enhanced image as some frequencies that are set to zero may correspond to signal. However, the quality is still decent.

Code

```
F2_new = F_new;
F2_new(x1-7:x1+7, y1-7:y1+7) = 0;
F2_new(x2-7:x2+7, y2-7:y2+7) = 0;
F2_new(x1-230:x1+15, y1) = 0;
F2_new(x2-15:x2+230, y2) = 0;
F2_new(x1, y1-7:y1+240) = 0;
F2_new(x2, y2-240:y2+7) = 0;
S2_new = abs(F2_new);
S2s_new = fftshift(S2_new.^0.1);
f2_new = ifft2(F2_new);

% display power spectrum of further enhance image (Fig. 12)
imagesc(S2_new.^0.1) ; axis off; title('Power Spectrum of further enhanced
image (without fftshift)');
colormap('default');

% display the further enhance image (Fig. 13)
```

```
imagesc(f2_new) ; axis off; title('Further Enhanced image');
colormap(gray);

% display orginal, enhanced, further enhanced image (Fig. 14)
subplot(1,3,1); imagesc(f) ; axis off; title('Original image');
subplot(1,3,2); imagesc(f_new) ; axis off; title('Enhanced image');
subplot(1,3,3); imagesc(f2_new) ; axis off; title('Further Enhanced
image');
```

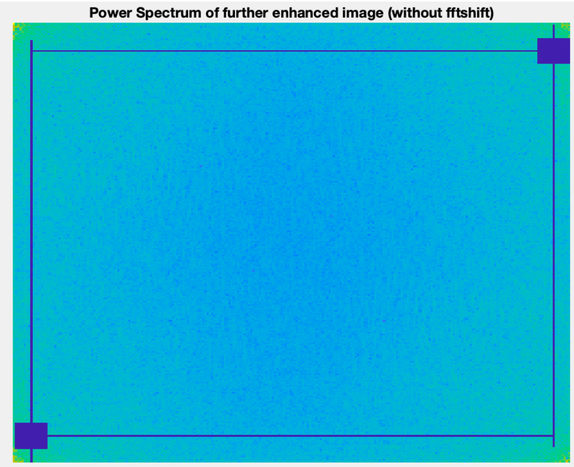

Fig. 12

Fig. 13

Fig. 14

Free the primate by filtering out the fence

Code

```
f_rgb = imread('primate-caged.jpg');
f = rgb2gray(f_rgb);

F = fft2(f);
S = abs(F);
Ss = fftshift(S.^0.1);       %non-linearly scale the power
imagesc(Ss);

x1 = 134; y1 = 119;
x2 = 124; y2 = 139;
x3 = 138; y3 = 108;
x4 = 120; y4 = 150;
```

```
x5 = 143; y5 = 97;
x6 = 115; y6 = 161;

Fs_new = fftshift(F);

threshold = 2;
Fs_new(x1-threshold:x1+threshold, y1-threshold:y1+threshold) = 0;
Fs_new(x2-threshold:x2+threshold, y2-threshold:y2+threshold) = 0;
Fs_new(x3-threshold:x3+threshold, y3-threshold:y3+threshold) = 0;
Fs_new(x4-threshold:x4+threshold, y4-threshold:y4+threshold) = 0;
Fs_new(x5-threshold:x5+threshold, y5-threshold:y5+threshold) = 0;
Fs_new(x6-threshold:x6+threshold, y6-threshold:y6+threshold) = 0;
f_new = ifft2(ifftshift(Fs_new));

% display Fig.15
subplot(1,3,1); imagesc(f) ; axis on; title('Original image');
subplot(1,3,2); imagesc(abs(f_new)) ; axis on; title('Enhanced image');
subplot(1,3,3); imagesc(abs((Fs_new).^0.1)) ; axis on; title('Power
Spectrum of Enhanced image (shifted)');
colormap(gray)
```

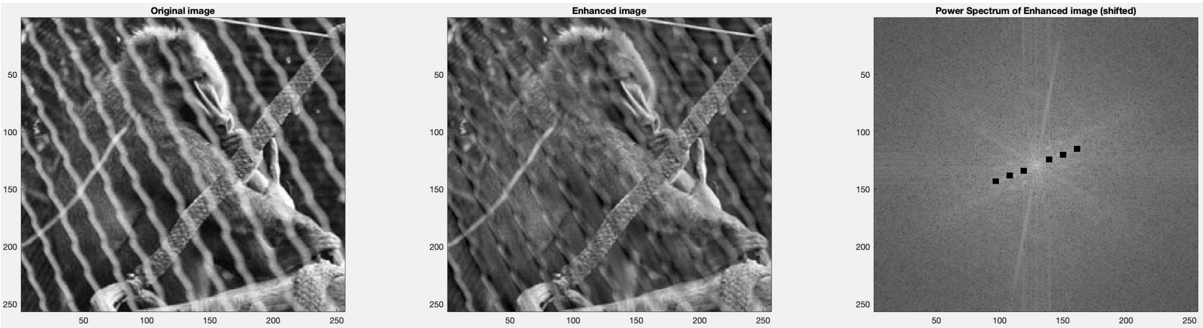

Fig. 15

Discussion

The cage on the original picture corresponds to high frequencies in Fourier domain. From the original image in Fig. 15, we can observe that the cage vary more along y-axis than x-axis, as it cuts the x and y axes 4 and 10 times respectively. Hence, the frequencies of the cage is expected to be represented as bright spots along a line with lower slope in the power spectrum as it contains higher-frequency horizonal component. The enhanced image is obtained by setting these bright spots and their neighbours to zero. The cage is not removed completely because there are other frequencies that made up the cage. However, these frequencies also made up the primate, removing them will blur the primate.

Experiment 6: Undoing perspective distortion of planar surface

Code

```
P = imread('book.jpg');
imagesc(P);
[X, Y] = ginput(4);

% A4 210 x 297
Xim = [0 210 210 0];
Yim = [0 0 297 297];

A = [
```

```
    [X(1), Y(1), 1, 0, 0, 0, -Xim(1)*X(1), -Xim(1)*Y(1)];
    [0, 0, 0, X(1), Y(1), 1, -Yim(1)*X(1), -Yim(1)*Y(1)];
    [X(2), Y(2), 1, 0, 0, 0, -Xim(2)*X(2), -Xim(2)*Y(2)];
    [0, 0, 0, X(2), Y(2), 1, -Yim(2)*X(2), -Yim(2)*Y(2)];
    [X(3), Y(3), 1, 0, 0, 0, -Xim(3)*X(3), -Xim(3)*Y(3)];
    [0, 0, 0, X(3), Y(3), 1, -Yim(3)*X(3), -Yim(3)*Y(3)];
    [X(4), Y(4), 1, 0, 0, 0, -Xim(4)*X(4), -Xim(4)*Y(4)];
    [0, 0, 0, X(4), Y(4), 1, -Yim(4)*X(4), -Yim(4)*Y(4)];
];
v = [Xim(1); Yim(1); Xim(2); Yim(2); Xim(3); Yim(3); Xim(4); Yim(4);];
u = A\v;
U = reshape([u;1], 3, 3)'           % matrix of m11 to m31

w = U*[X'; Y'; ones(1,4)]           %[[kxim][kyim][k]]
w = w ./ (ones(3,1) * w(3,:))       %[[xim][yim][1]]
T = maketform('projective', U');
P2 = imtransform(P, T, 'XData', [0 210], 'YData', [0 297]);

% display Fig.16
imagesc(P2);
```

Results

```
U =

    1.3853    1.4577 -234.3069
   -0.4203    3.4923  -28.0647
    0.0001    0.0049    1.0000


w =

         0  263.5282  438.1923    0.0000
    0.0000    0.0000  619.7291  529.3552
    1.1380    1.2549    2.0866    1.7823


w =

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

Fig. 16

Fig. 17

Discussion

In Fig. 16, we solve for U which is the matrix of m11, m12, m12, m21, m22, m23, m31, m32, m33. We then solve for $\begin{bmatrix} kx_{im} \\ ky_{im} \\ k \end{bmatrix}$ which is represented by w (second row). Lastly, we transformed $\begin{bmatrix} kx_{im} \\ ky_{im} \\ k \end{bmatrix}$ into $\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix}$ (last row w) which is the new coordinates of the book cover.

The results of projective transformation is shown in Fig. 17. The quality at the bottom of the result book cover is much better than the top part. This is because the original perspective of the bottom part of the book is closer to the new perspective than that of the top part. The overall quality of the image is still good compared to the original image and the new image shows the frontal view of the book.