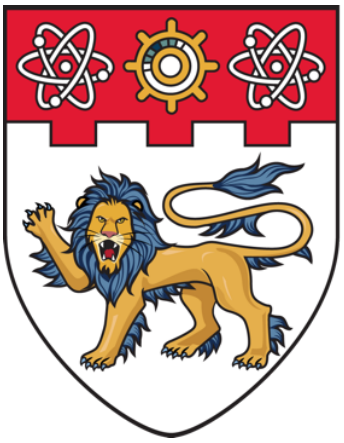


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4003: Computer Vision

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

Gede Bagus Bayu Pentium (U1420090G)

School Of Computer Science and Engineering

1. Introduction

1.1 Objectives

This laboratory aims to introduce image processing in MATLAB context. In this laboratory you will:

- Become familiar with the MATLAB and Image Processing Toolbox software package.
- Experiment with the point processing operations of contrast stretching and histogram equalization.
- Evaluate how different Gaussian and median filters are suitable for noise removal.
- Become familiar with the frequency domain operations
- Understand imaging geometry.

1.2 Tools

In this experiments we will be using **MATLAB** and **Image Processing Toolbox software package**.

2. Eksperiments And Results

2.1 Contrast Stretching

In this experiment we enhance the image with poor contrast with applying contrast stretching.

a. Input the image into a MATLAB matrix variable and convert into grayscale.

- **Code:**

```
Pc = imread('image/mrttrainbland.jpg');
whos Pc;
P = rgb2gray(Pc);
whos P;
```

- **Result:**

```
>> Pc = imread('image/mrttrainbland.jpg');
whos Pc;
  Name         Size                Bytes  Class    Attributes

  Pc         320x443x3            425280  uint8

>> P = rgb2gray(Pc);
whos P;
  Name         Size                Bytes  Class    Attributes

  P          320x443              141760  uint8
```

From the result we can see the dimension of the picture changes form `320*443*3` to `320*443`. The original image was a 3 dimensional matrix indicating a colour image. After the `rgb2gray` fucntion applied to the image the greayscaled image only have 2 dimension.

b. View the image.

Here we can see that the image has poor contrast. To improve the quality of the image some point processing can be done such as *Contrast Stretching, Power-law Transformation, and Histogram Equalization*.

- **Code:**

```
imshow(P);
```

- **Result:**

c. Check minimum and maximum intensities present in the image

- **Code:**

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

- **Result:**

```
ans =

  uint8

    13


ans =

  uint8

   204
```

Contrast stretching involve linear scaling of the gray level from minimum to the maximum value. Here we can say that contrast stretching still can be done by scaling the image gray level from `13` to `204`.

d. Do contrast stretching

we do the contrast stretching using `imsubtract` and `immultiply` to substract and multiply the gray level in the image and keep the `uint8` format.

- **Code:**

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

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

- **Result:**

```
ans =

  uint8

    0
```

```
ans =

  uint8

  255
```

From the result we can see that the image gray level now scaled with `min = 0` and `max = 255`. This show the image gray level is already scaled to their minimum and maximum

e. Redisplay the image

- **Code:**

```
imshow(P2);
imshow(P, []);
```

- **Result:**

From the result we can see both `imshow(P2)` and `imshow(P,[])` display image with similar contrast. This show our contrast stretching is successful. Both the new image (after contrast stretching) has better contrast compared to the original image.

2.2 Histogram Equalization

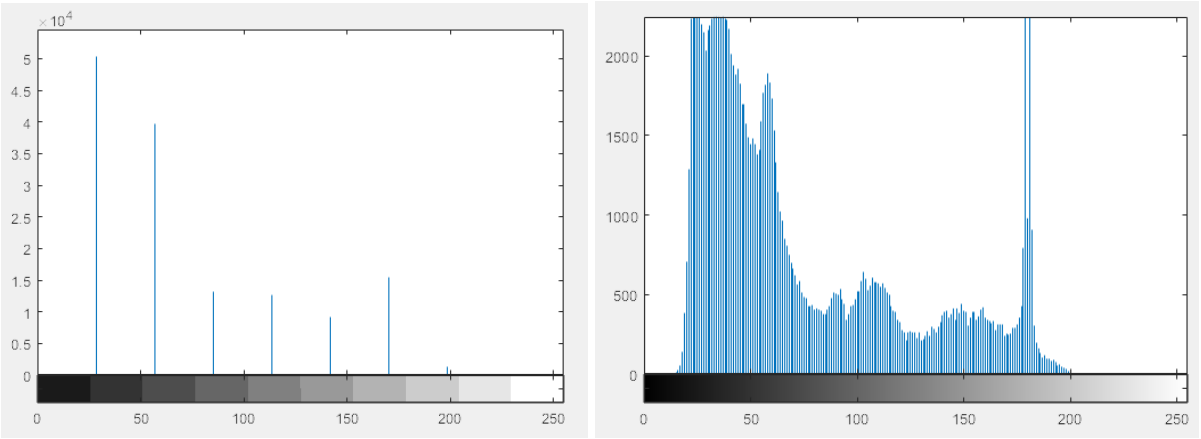
In this experiment we enhance the image with non-uniform gray level with applying histogram equalization (`histeq`).

a. Display the image intensity histogram with 10 bin and 256 bin.

- **Code**

```
Pc = imread('image/mrttrainbland.jpg');
P = rgb2gray(Pc);
imhist(P, 10);
imhist(P, 256);
```

- **Result:**



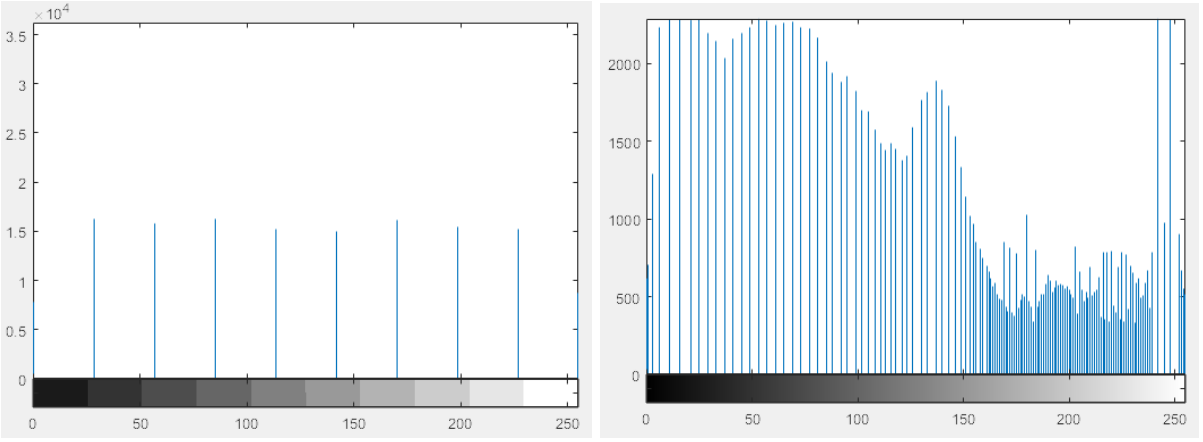
The two image intensity show how the gray level is distributed. The main differences is that the first image the gray level grouped into 10 bin and the second image grouped into 256 bin. This causing the bin in the first image contain more pixel compared with bin in the second image. In the first image (10 bin) show more generalized histogram as the gray level is grouped into larger group. In the second image (256 bin) we can see that the original gray level distribution without grouping. The result from the second historgram is showing more details but we can see the trend in both histogram is the same.

b. Carry out histogram equalizatioin

- **Code:**

```
P3 = histeq(P, 255);
imhist(P3, 10);
imhist(P3, 256);
```

- **Result:**



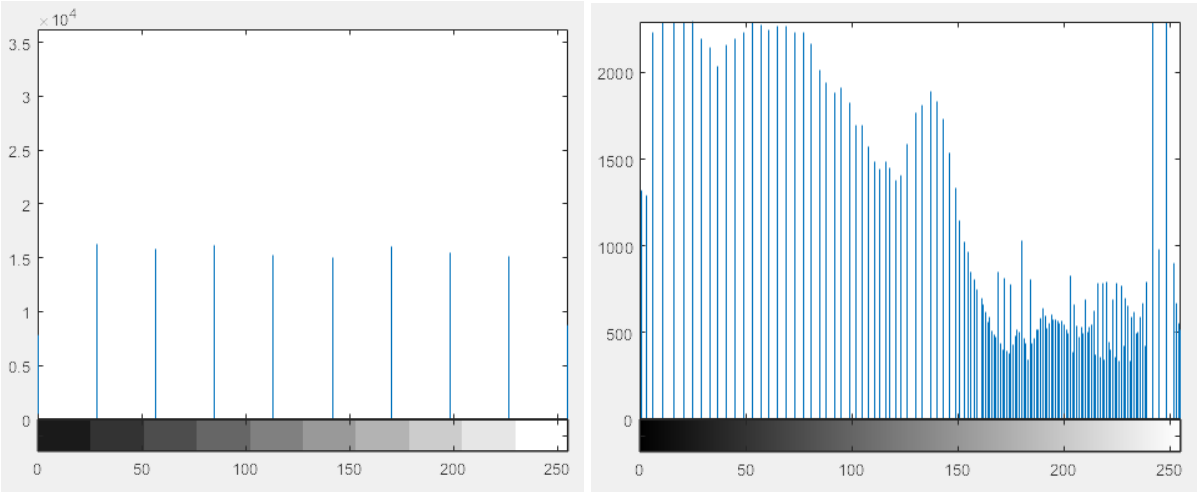
From the result we can see the effect of histogram equalization. On the first image we can see the histogram is almost equalized. But on the second image the histogram is not equalized. But on both image we can see that compare with before histogram equalization the number of pixel in each gray level is distributed more equally. The histogram equalization supposed to equalized the number of pixel in each gray level (bin) but as we know the image used in this experiment using discrete gray-levels, so histogram is only **approximately flat**.

c. Rerun the histogram equalization

- **Code:**

```
P3 = histeq(P3, 255);
imhist(P3, 10);
imhist(P3, 256);
```

- **Result:**



As we know reapplying the algorithm will not improve the result. The histogram not become more uniform. This is because the histogram equalization will try to shift the gray-level bins horizontally in the histogram. So after applying it once the resulting histogram will remain unchanged (because the histogram is already equalized).

Here is the image after equalization:

The resulting image is improved after the first histogram equalization but remain unchange after reapplying.

2.3 Linear Spatial Filtering

In this experiment we enhance the image quality by removing noise using gausian filtering

a. Generate Gausian Averaging filters
    i. Y and X dimensions are 5 and sigma = 1.0

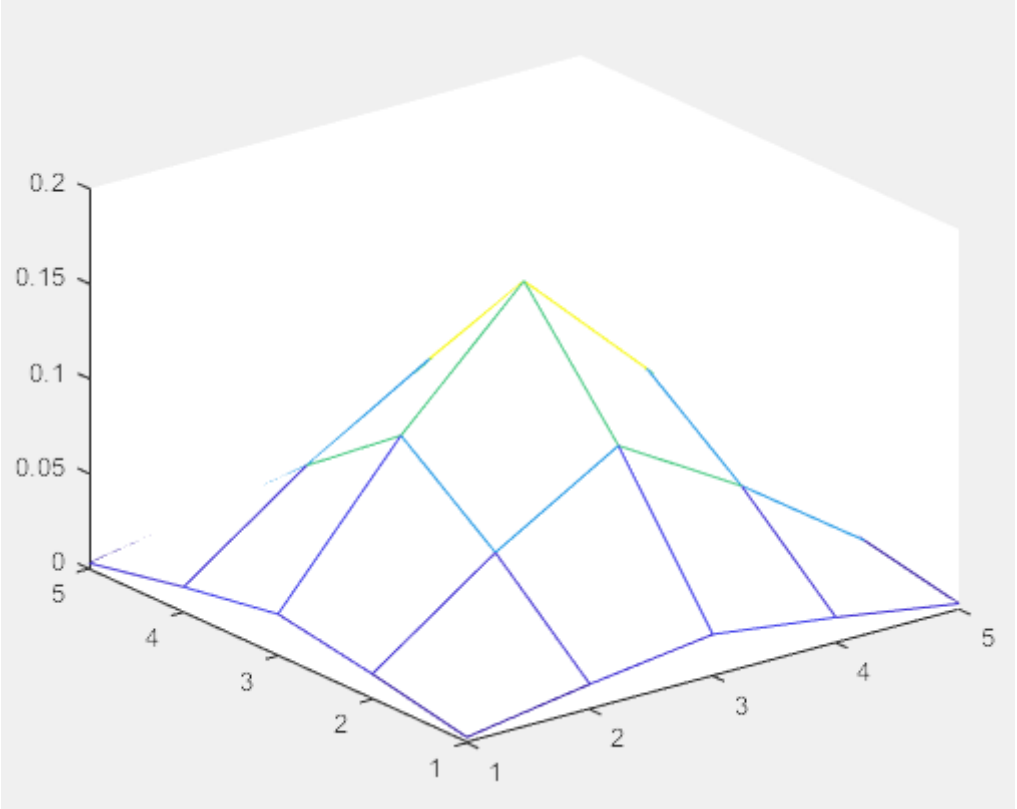
- **Code:**

```
x = -2:2;
y = -2:2;
sigma_1 = 1.0;

[X,Y] = meshgrid(x,y);

filter_1 = exp(-((X).^2 + (Y).^2) / (2 * sigma_1^2));
filter_1 = filter_1 ./ (2 * pi * sigma_1^2);
filter_1 = filter_1 ./ sum(filter_1(:));
mesh(filter_1);
```

- **Result:**



ii. Y and X dimensions are 5 and sigma = 2.0

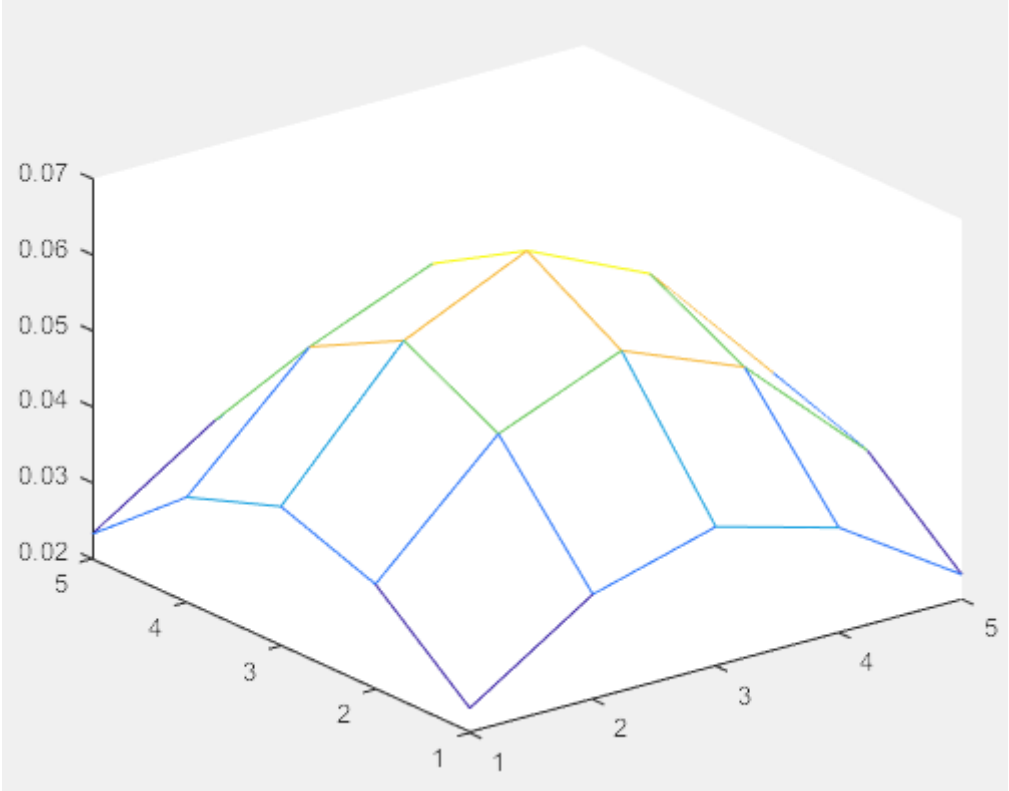
- **Code:**

```
x = -2:2;
y = -2:2;
sigma_2 = 2.0;

[X,Y] = meshgrid(x,y);

filter_2 = exp(-((X).^2 + (Y).^2) / (2 * sigma_2^2));
filter_2 = filter_2 ./ (2 * pi * sigma_2^2);
filter_2 = filter_2 ./ sum(filter_2(:));
mesh(filter_2);
```

- **Result:**



b. Display ntugn.jpg (image with additive Gaussian noise).

- **Code:**

```
P = imread('image/ntugn.jpg');
imshow(P);
```

- **Result:**

c. Apply Filter

- **Code:**

```
P1 = uint8(conv2(P,filter_1));
imshow(P1);
P2 = uint8(conv2(P,filter_2));
imshow(P2);
```

- **Result:**

As we can see from the result the first filter (sigma = 1.0) remove less noise compare to the second filter (sigma = 2.0). But on the resulting image the first filter has more detail compared with the second filter (second image is blurrer). Because of the blur effect the second image also look unrealistic. With this we can see higher filter coefficients (sigma) result in more noise filtered but less detailed image. This show that filtering results in information loss (less detail).

d. Display ntusp.jpg (image with additive speckle noise).

- **Code:**

```
P = imread('image/ntusp.jpg');
imshow(P);
```

- **Result:**

e. Apply Filter

- **Code:**

```
P1 = uint8(conv2(P,filter_1));
imshow(P1);
P2 = uint8(conv2(P,filter_2));
imshow(P2);
```

- **Result:**

Similar with the previous image the second image has less detail and less noise. But in this case the noise is not filtered very well (even on the second image we can still see the image). This shows Gaussian averaging filter is better in handling gaussian noise compare to speckle noise. The reason is because gaussian averaging filtering give weighted value to all surounding pixel. But on the speckle noise the noise pixel has a very different value compared with the surrounding pixel. Hence the best way is to give zero weight (or ignore) the pixel. And in gaussian filter those pixels are still calculated with various weight.

2.4 Median Filtering

a. Median filtering using function `medfilt2` from matlab.
b. Display ntugn.jpg (image with additive Gaussian noise).

- **Code:**

```
P = imread('image/ntugn.jpg');
imshow(P);
```

- **Result:**

c. Apply filter

- **Code:**

```
P1 = medfilt2(P,[3,3]);
imshow(P1);
P2 = medfilt2(P,[5,5]);
imshow(P2);
```

- **Result:**

We can see from the first picture that the median filter is not very effective in reducing additive gaussian noise. But on the second picture we can see the noise is removed in tradeoff with lossing more details (compared with gaussian averaging filter with the same size `5*5`).

d. Display ntusp.jpg (image with additive speckle noise).

- **Code:**

```
P = imread('image/ntusp.jpg');
imshow(P);
```

- **Result:**

e. Apply filter

- **Code:**

```
P1 = medfilt2(P,[3,3]);
imshow(P1);
P2 = medfilt2(P,[5,5]);
imshow(P2);
```

- **Result:**

From the result we can see that the median filtering is handling speckle noise very well. Even with smaller filter size (3*3) it handle speckle noise better than gaussian averaging filter.

To conclude, the gaussian noise is handled better with Gaussian averaging filter and speckle noise is handled better with median filter. With Gaussian filter the details is not loss because all of the pixel have weight. This is also make the Gaussian filter good for Gaussian noise because the noise itself is not on the single pixel but distributed along several pixels. In contrary speckle noise has different characteristic. This type of noise is better handled with median filter. The median filter will select only the median on the filter which will ignore the pixel noise in speckle noise.

If we only consider noise removal the median filter will generally work better compared to Gaussian averaging filter. But if we look at the quality (details) of the resulting filter Gaussian filter actually leave more details. So, the tradeoff here is that noise removal and information loss. Gaussian filter have less information loss but did not remove noise completely (the noise still have weight in the resulting image). In contrast, median filter remove more noise (ignore the pixel completely) but having more information loss.

2.5 Suppresing Noise Interference Patterns

a. Display pckint.jpg

- **Code:**

```
P = imread('image/pckint.jpg');
imshow(P);
```

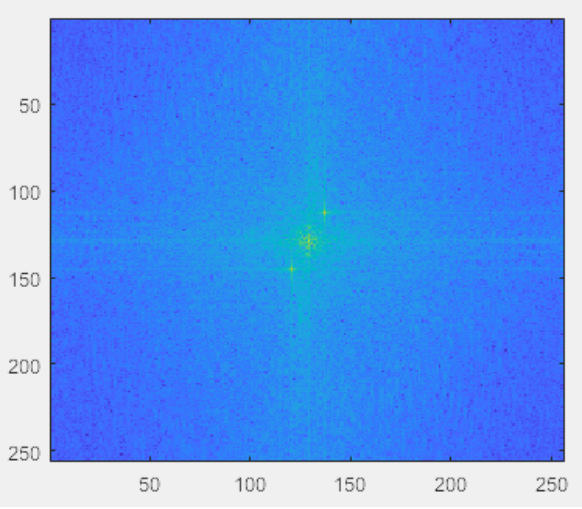
- **Result:**

b. Display power spectrum with Fourier transform

- **Code:**

```
F = fft2(P);
S = abs(F);
imagesc(fftshift(S.^0.1));
colormap('default');
```

- **Result:**



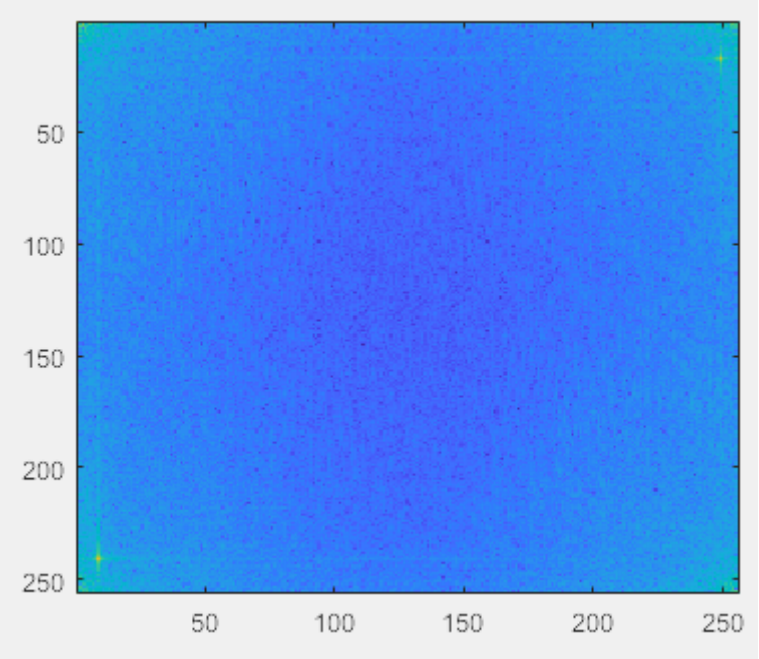
From the image above we can see the two symetry peak are the frequency of the interference pattern. This is captured as peak because the noise is making a pattern with the similar pixel value, hence it will increase the frequency.

c. Redisplay power spectrum without `fftshift` and get the coordinates of the peak.

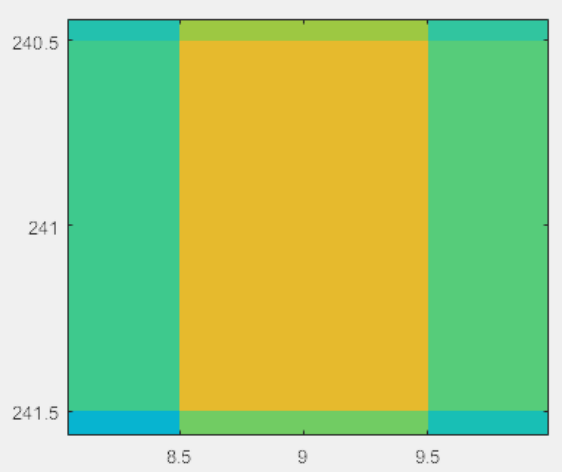
- **Code:**

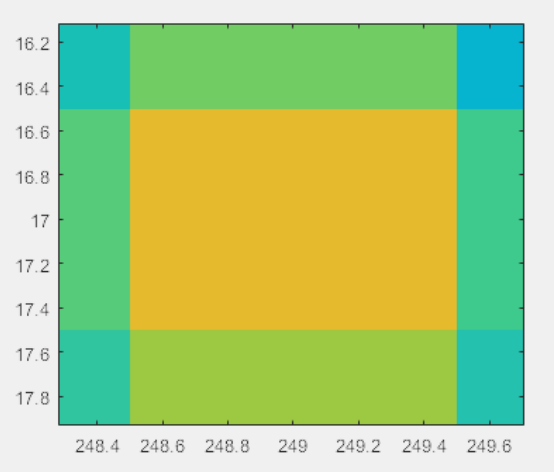
```
imagesc(S.^0.1);
colormap('default');
```

- **Result:**



From the figure above we can see 2 peak on the top right corner and bottom left corner. We will zoom the image to see the coordinates of the peaks.





From the zoomed image we can see the peak is at:

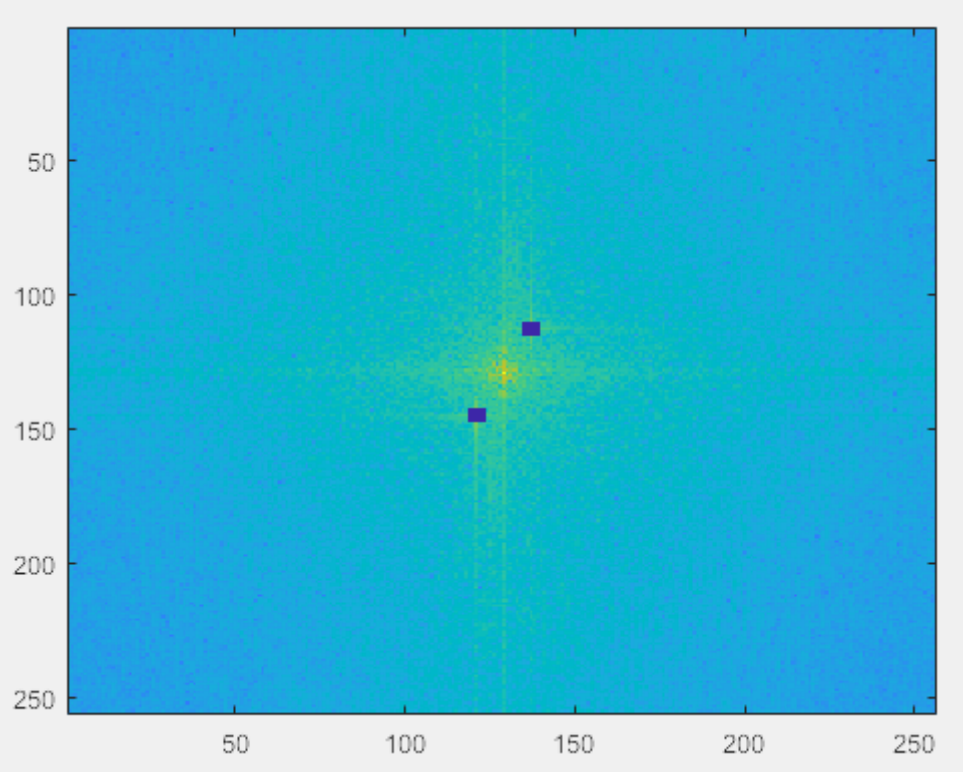
- `x = [240.5,241.5], y = [8.5,9.5] => (241,9)`
- `x = [16.5,17.5], y = [248.5,249.5] => (17,249)`

d. Recompute the power spectrum

- **Code:**

```
x1 = 241; y1 = 9;
x2 = 17; y2 = 249;
F(x1-2:x1+2, y1-2:y1+2) = 0;
F(x2-2:x2+2, y2-2:y2+2) = 0;
S = abs(F);
imagesc(fftshift(S.^0.1));
colormap('default');
```

- **Result:**



From the result we can see that the peak is removed (set to zero). That's mean the frequency was remove (frequency domain) correspond with removing the noise pattern in the original image (spatial domain).

e. Compute the inverse Fourier transform

- **Code:**

```
result = uint8(ifft2(F));
imshow(result);
```

- **Result:**

From the result we can find that the noise is removed. This because the high frequency of noise pattern in frequency domain (after Fourier transform) is set to zero. Because in the Fourier domain this noise is corespond to the high peak frequency that is set to zero. With this the pattern is removed from the spatial image.
We can see also that not all the noise is removed. This is because the noise is not comming from only two peaks. To improve the result finding another peak and set it to the same frequency as the other points.

f. Filter out the fence

- **Code:**

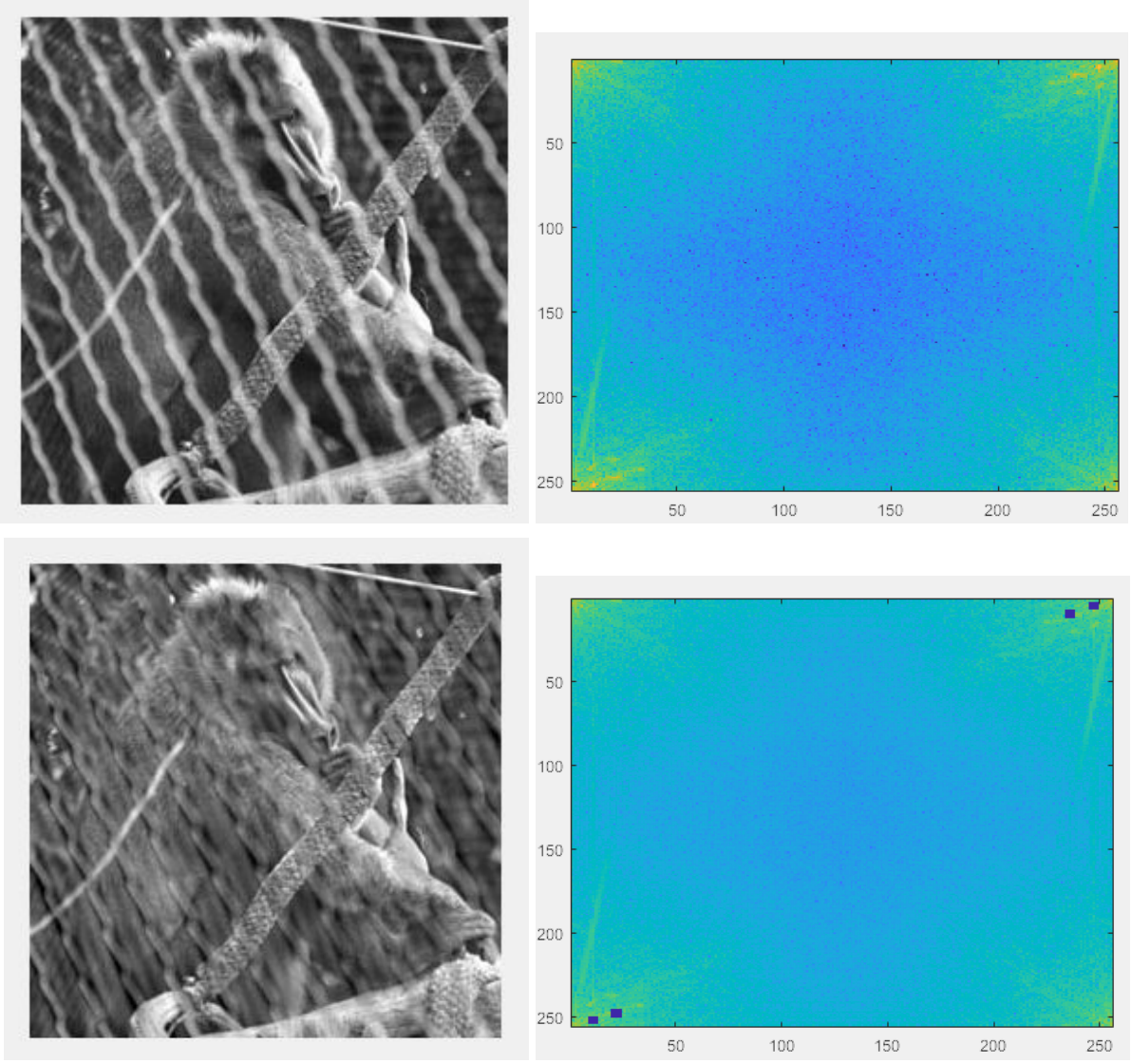
```
P = imread('image/primatecaged.jpg');
P = rgb2gray(P);
imshow(P);

F = fft2(P);
S = abs(F);
imagesc(S.^0.0001);
colormap('default');

x1 = 252; y1 = 11; F(x1-2:x1+2, y1-2:y1+2) = 0;
x2 = 248; y2 = 22; F(x2-2:x2+2, y2-2:y2+2) = 0;
x3 = 5; y3 = 247; F(x3-2:x3+2, y3-2:y3+2) = 0;
x4 = 10; y4 = 236; F(x4-2:x4+2, y4-2:y4+2) = 0;
S = abs(F);
imagesc(S.^0.1);
colormap('default');

result = uint8(ifft2(F));
imshow(result);
```

- **Result:**



There are 2 main peaks and 2 weak peaks found in the Fourier domain. In order to free the primate, we need to remove the cage which equivalent in removing the peaks in Fourier domain. In this case we consider the pattern created by the cage as noise. So the following peaks is removed:

- `x1, y1 = (252, 11)`
- `x2, y2 = (248, 22)`
- `x3, y3 = (5, 247)`
- `x4, y4 = (10, 238)`

From the result above we can see that the cage is not removed cleanly. The result shown above is by removing 4 peaks at the frequency domain. Removing more peaks and changing the size of removed pixel will not improve the result. This is because the cage is not exactly a noise pattern, the cage was a natural part of the picture hence the pattern created is not as good as the noise pattern. With this the Fourier transform cannot detect the pattern very accurate. To improve the result may be more complex filter need to be applied.

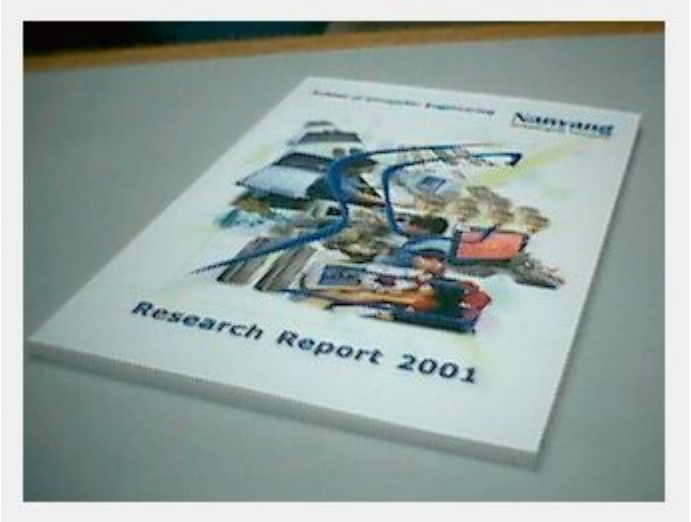
2.6 Undoing Perspective Distortion of Plannar Surface

a. Display book.jpg

- **Code:**

```
P = imread('image/book.jpg');
imshow(P);
```

- **Result:**



From the result can be seen that the image is a book that is looked from bottom right angle.

b. Obtain 4 corner coordinates

- **Code:**

```
[X, Y] = ginput(4);
Xim = [0; 210; 210; 0];
Yim = [0; 0; 297; 297];
```

- **Result:**

```
>> [X Y]

ans =

  142.0652   28.5000
  309.0217   46.9348
  256.5000  215.2826
    4.3261  158.9348
```

c. Setup matrices to estimate the projective transformation

- **Code:**

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

- **Result:**

```
A =

   1.0e+04 *

    0.0142    0.0028    0.0001         0         0         0         0         0
         0         0         0    0.0142    0.0028    0.0001         0         0
    0.0309    0.0047    0.0001         0         0         0   -6.4895   -0.9856
         0         0         0    0.0309    0.0047    0.0001         0         0
    0.0256    0.0215    0.0001         0         0         0   -5.3865   -4.5209
         0         0         0    0.0256    0.0215    0.0001   -7.6181   -6.3939
    0.0004    0.0159    0.0001         0         0         0         0         0
         0         0         0    0.0004    0.0159    0.0001   -0.1285   -4.7204
```

The image above is the value of transformation matrix. The transformation image is verified using by transforming the 4 corner of the books.

```
w =

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

From the result we can see that the corner of the desired image is transformed correctly (to `(0,0), (210, 0), (210, 297), (0, 297)`)

d. Wrap the image

- **Code:**

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

- **Result:**

The image P2 is the transformed image of P by matrix transformation.

e. Display the image

- **Code:**

```
imshow(P2);
```

- **Result:**

From the image above we can see the image of the deformed books (left) is trasnformed into the straight image (right). The result actually exceed the expectation in term of the proportionality. We can see that the image is well proportioned considering the original images. The bottom part of the image is transformed into a very good image.
However we can see that the top part of the image is very blurred and distorted (we cannot read the word) but again the result is expected since in the actual image you cannot even read the word at all.
So, overall the image result quality is very good considering the original image. Even the top part is not as good as the bottom part. This is mainly caused by the resolution of the original image is not very good and the top part of the book is further from the camera so it by original is blurrer compared with the bottom part.

3. Conclusion

After the experiment the following is learned:

- Familiar with MATLAB and Image Processing Toolbox Software Packages
- Learn the effect of contrast stretching and histogram equalization (on discrete gray level)
- Learn Gaussian and median filters with suitable noise that can be handled
- Familiarise concept of frequency domain using Fourier transformation (filter at Fourier domain)
- Understand transformation (translation and rotation) of images (imaging geometry)

4. Reference

- https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm
- https://en.wikipedia.org/wiki/Histogram_equalization
- https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm
- https://www.math.uci.edu/icamp/courses/math77c/demos/linear_spatial_filtering.pdf
- https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Image%20Filtering_2up.pdf
- CZ4003 - Lecture Notes
- CZ4003 - Lab Manuals