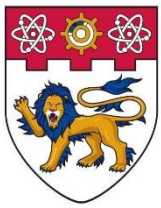


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

AY2021 Semester 1

Computer Vision

Lab 1

Title:

Point Processing + Spatial Filtering + Frequency Filtering + Imaging Geometry

Name: Raymond Toh

Matric Number: U1821599H

Tutor: Qian Kemao

Experiments Result

2.1 Contrast Stretching

After loading image, **whos** command was used to check image have how many channels (3 channel – RGB, 1 channel – Grayscale). The output size shows us it has 320x443 by 3 colour channels (RGB Channel). Hence, we convert it into grayscale image by using rgb2gray function.

```
>> image = imread('mrt-train.jpg');
>> whos image
  Name          Size                Bytes  Class    Attributes

  image       320x443x3            425280  uint8
```

```
>> gray_image = rgb2gray(image);
```

imshow function was used to view the image and **min()** and **max()** function was also used to check the minimum and maximum intensity of the image.

```
>> imshow(gray_image);
>> min(gray_image(:)), max(gray_image(:))

ans =

  uint8

    13


ans =

  uint8

   204
```

Contrast Stretching was achieved by normalising the image intensity to a range of 0 to 255 as that is the range of colour.

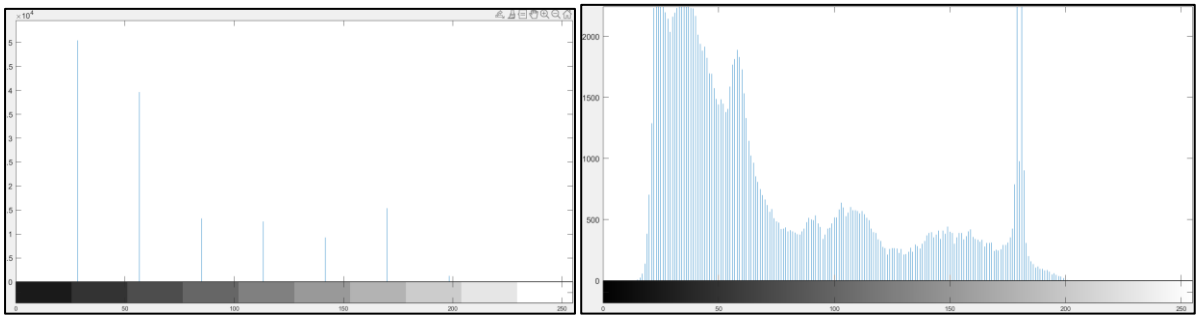
```
>> stretched_image = (255/191)*imsubtract(gray_image,13);
>> min(stretched_image(:)), max(stretched_image(:))

ans =

  uint8

    0


ans =

  uint8

   255

>> imshow(stretched_image,[]);
```

2.2 Histogram Equalization

histeq() function was used to do histogram equalization.

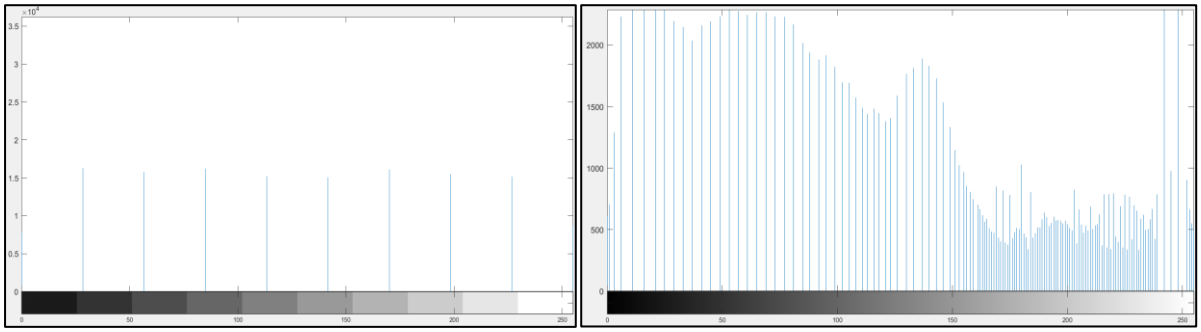
imhist() function was used to display the image intensity in this lab with 10 and 256 bins.

```
>> imhist(gray_image, 10)
>> imhist(gray_image, 256)
```



Histogram equalization result

```
>> histeq_image = histeq(gray_image, 255);
>> imhist(histeq_image, 10)
>> imhist(histeq_image, 256)
```



Histogram become more uniform because it spread out the most frequent intensity values which allows for area with lower local contrast to gain a higher contrast.

2.3 Linear Spatial Filtering

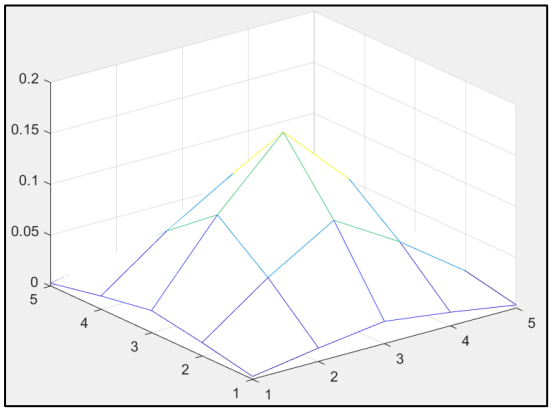
Function for filter:

```
function h = h(x,y,sigma)
    h=(1/(2*pi*power(sigma,2)))*(exp(-(power(x,2)+power(y,2))/(2*power(sigma,2))));
end
```

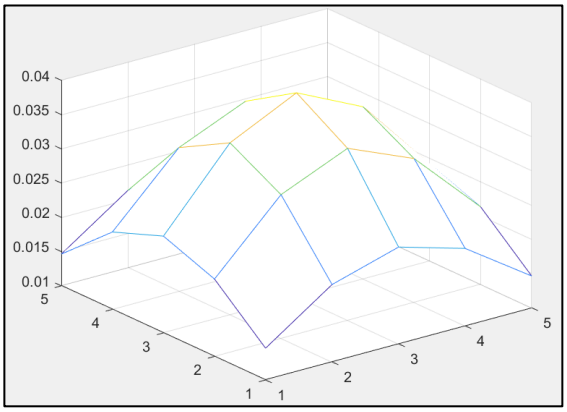
Create Filter (X and Y dimension = 5) and Normalize Filter

```
%Declare Variable
sigma_1 = 1.0;
sigma_2 = 2.0;

%Axis for filter
dim=5;
range = -floor(dim/2) : floor(dim/2);
[X, Y] = meshgrid(range,range);

%Write h(x,y,sigma) function in matladb -> h.m
%To use function -> h(x,y,sigma)
%Fill values in 5x5 Filter
G1 = h(X,Y,sigma_1);
G2 = h(X,Y,sigma_2);

%Normalize Filter
G1 = G1/sum(G1(:));
G2 = G2/sum(G1(:));

%View my Gaus_Filter
mesh(G1)
mesh(G2)
```

Sigma_1 filter



Sigma_2 filter



Original Image

Image after Convolute with Gaussian Filter

```
%Convolution NTU Image with Filter
ntu_gn = imread('lib-gn.jpg');
G1OP = uint8(conv2(ntu_gn, G1));
G2OP = uint8(conv2(ntu_gn, G2));
```

Sigma_1 filter result

Sigma_2 filter result

Using Gaussian averaging filter, with the increase of sigma, the noise is reduced at the expense of the clarity and brightness of the image, resulting in a blurrier and darker image. In this example, original image will still be preferred. I believe human eye able to pick up more information in original image compared to the filtered image.

Image with Speckle Noise Convolute with Gaussian Filter

```
%Filter with another NTU Image with Filter
ntu_sp = imread('lib-sp.jpg');
G1OP2 = uint8(conv2(ntu_sp, G1));
G2OP2 = uint8(conv2(ntu_sp, G2));
```

Sigma_1 filter result

Sigma_2 filter result

After applying to speckle noise, speckle noise still able to be seen from human eye. Image also became blurry. Hence, gaussian averaging filter still work better in removing gaussian noise

2.4 Median Filtering

Using Median Filter with Gaussian Noise Images and Speckle Noise Images

```
%Applying Median Filter to NTU Image
M1 = medfilt2(ntu_gn,[3,3]); %G_noise
M2 = medfilt2(ntu_gn,[5,5]); %G_noise
M3 = medfilt2(ntu_sp,[3,3]); %S_noise
M4 = medfilt2(ntu_sp,[5,5]); %S_noise
```

Results

3x3 neighbourhood with Gaussian Noise

5x5 neighbourhood with Gaussian Noise

3x3 neighbourhood with Speckle Noise

5x5 neighbourhood with Speckle Noise

Median Filter for neighbour sizes of 3x3 work extremely good on NTU image with speckle noise, as compared to 5x5 median filter was a little blurry. Median filter does not work well for Gaussian Noise image as the noise still can be seen after applying the filter. In Conclusion, Gaussian Average Filter work better for image with gaussian noise, and Median Filter work better for image with speckle noise.

2.5 Supressing Noise Interference Pattern

Load and display image

```
%Load and display Image
pck = imread('pck-int.jpg');
imshow(pck)
```

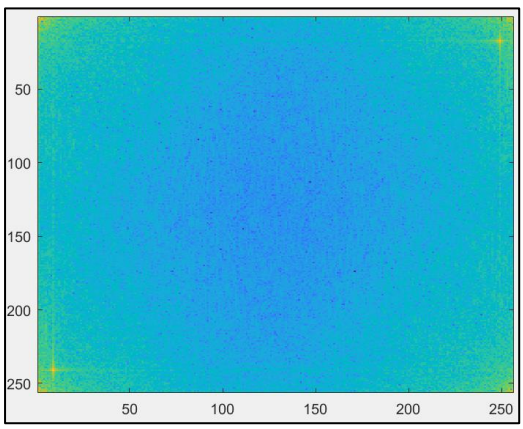
Fast Forward Transform change image from spatial domain to frequency domain. Hence, it allows us to spot different noises that are from different frequencies and able to remove them easily.

Transform Image with FFT2

```
%Transform Image using Fast Fourier Transform 2D
fftpck = fft2(pck);
```

Low frequency is on the corner of the image and high frequency is in the centre of image.

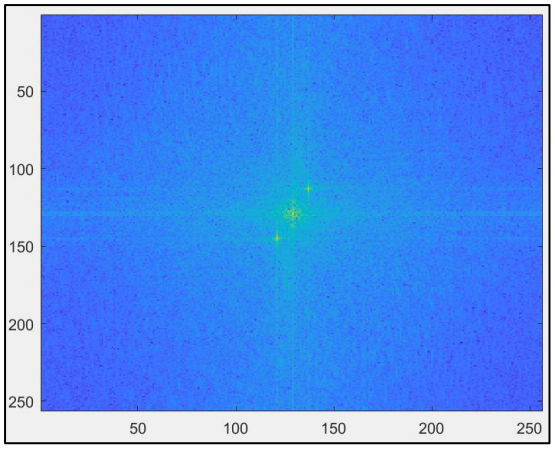
```
imagesc(real(fftpck.^0.01))
```



Fast Fourier Transform Shift move the low frequency component to centre of image including the DC component and high frequency component to the corners of image.

Notice the star-like components are the unwanted frequencies that produces the interference in the original images.

```
%Shift DC component to centre of image
imagesc(fftshift(real(fftpck.^0.1)))
colormap('default')
```

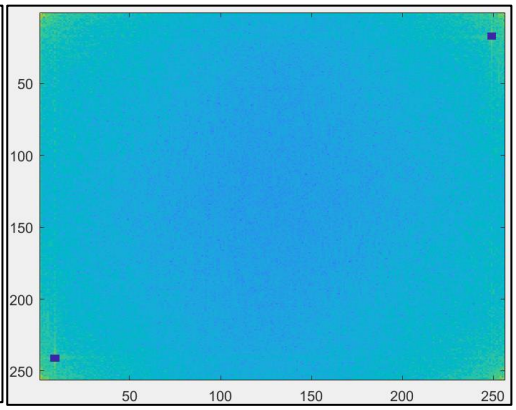


ginput() function was used to determine the x and y coordinate for the unwanted frequencies. Algorithm was implemented to zero the 5x5 neighbourhood elements from the unwanted frequencies and used Inverse Fast Fourier Transform to change from frequency domain to spatial domain to allow us to view the image.

Code

```
%Capture X Y Corrdinate of two distinct frequency peak
%coord1 = [9.0161, 241.1676], coord2 = [249.1882, 17.5552]
%[x y] = ginput(2)

%Remove Two distinct frequency peak in spatial domain
fftpck_zero = fftpck;
dim=floor(5/2);

%Notice [X Y], in domain will be [Y X]
fftpck_zero(241-dim:241+dim,9-dim:9+dim)=0;
fftpck_zero(17-dim:17+dim,249-dim:249+dim)=0;
imagesc(real(fftpck_zero.^0.1))

%Perform Inverse Fourier Transform
fftpck_zero_ifft = uint8(ifft2(fftpck_zero));
imshow(fftpck_zero_ifft)
```

zero 5x5 neighbour



Result after remove zero 5x5 neighbour of the unwanted frequencies.

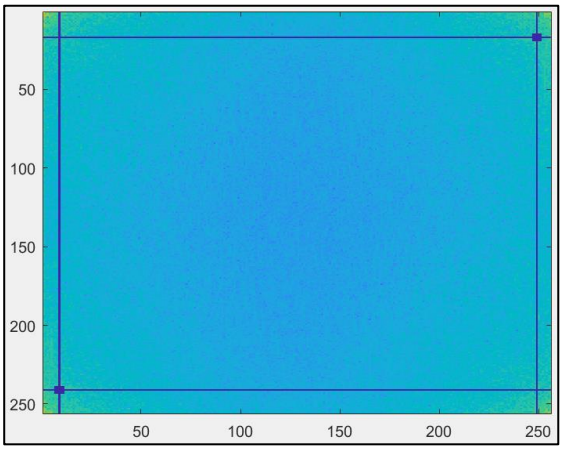
The interference patterns were reduced. When we do a fourier transform on image, we can see the origin on the top left-hand corner of the image due to low frequency and the interference pattern frequency was in the top right and bottom left edge due to Euler's identity. By applying fftshift, the low frequency component of image will be at the centre of image and frequency will go higher as it approaches the edge of image.

As there are still interference frequency shown in spatial domain, we can further improve the image by zeroing the frequency by doing this.

Code

```
%Make another copy
fftpck_zero2 = fftpck;

%Notice [X Y], in domain will be [Y X]
%coord1 = [9.0161, 241.1676], coord2 = [249.1882, 17.5552]
fftpck_zero2(241-dim:241+dim,9-dim:9+dim)=0;
fftpck_zero2(17-dim:17+dim,249-dim:249+dim)=0;

%Extend the zero to edge
fftpck_zero2(:,9)=0;
fftpck_zero2(241,:)=0;
fftpck_zero2(:,249)=0;
fftpck_zero2(17,:)=0;

%Display the zero
imagesc(real(fftpck_zero2.^0.1))

%Perform Inverse Fourier Transform
fftpck_zero2_ifft = uint8(ifft2(fftpck_zero2));
imshow(real(fftpck_zero2_ifft))
```

Zero other interference



Results

Original | zero 5x5 neighbour | zero 5x5 neighbour + others

Remove cage from Primate Caged image. In this experiment, we will consider the cage as the noise interference.

Check for number of channels, convert to grayscale if there are RGB channels.

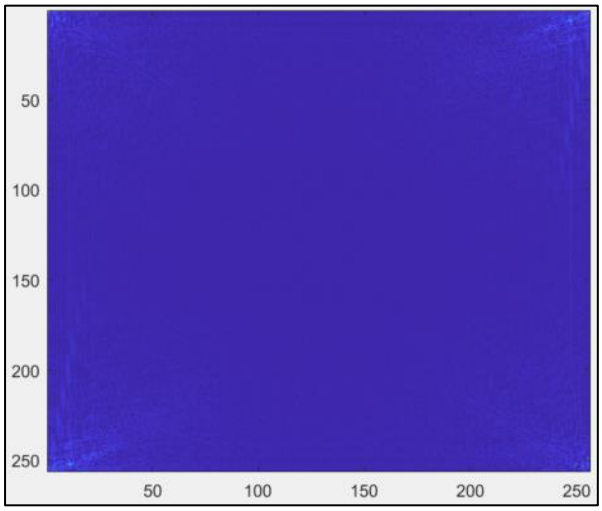
```
pc = imread('primate-caged.jpg');

%Check for RGB or grayscale image
whos pc

%Change to grayscale
pc = rgb2gray(pc);
```

Original image

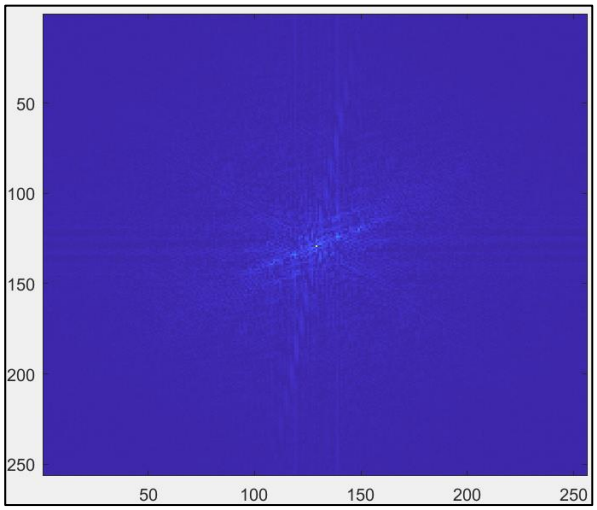
Code for finding interference and zero it

```
%Fourier Transform 2D
fftpc = fft2(pc);

%Check for spectrum, we can see there are 4 tiny spectrum causes the effect
imagesc(fftshift(real(fftpc.^0.5)));

fftpc_zero = fftpc;
imagesc(real(fftpc_zero.^0.5));


%[x y] = ginput(2)
x1=251; y1=11; fftpc_zero(x1-3:x1+3,y1-3:y1+3) = 0;
x2=10; y2=237; fftpc_zero(x2-3:x2+3,y2-3:y2+3) = 0;
x3=247; y3=21; fftpc_zero(x3-3:x3+3,y3-3:y3+3) = 0;
x4=6; y4=247; fftpc_zero(x4-3:x4+3,y4-3:y4+3) = 0;
imagesc(real(fftpc_zero.^0.5));
```

Fast Forward Transform

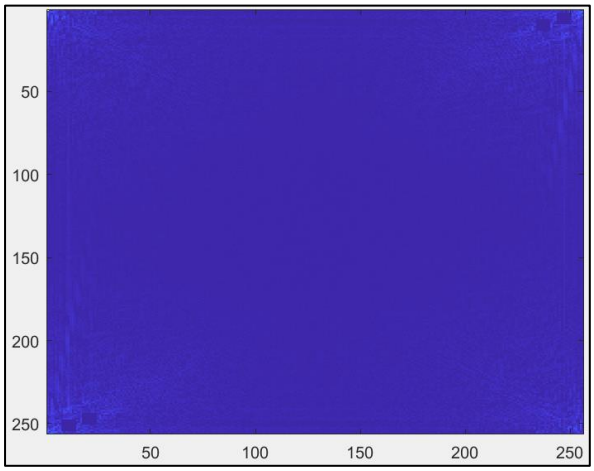


Fast Forward Transform and Shift



Notice there are 4 peak frequencies we are trying to reduce/remove.

Zeroed Noise Interference



After zeroed unwanted frequencies in frequency domain, inverse fast forward transform was applied to change image back to spatial domain and this is the result.

```
%Perform Inverse Fourier Transform
filted_fftpc_zero = uint8(ifft2(fftpc_zero));
imshow(real(filted_fftpc_zero))
```

Original Image

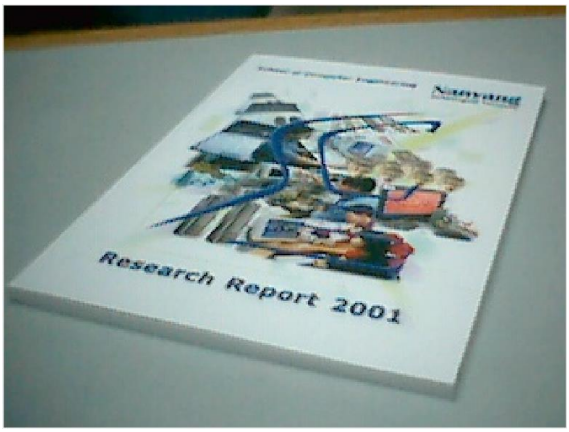
Zeroed Image

From the result above, the fence was not removed fully. Removing more peaks frequencies will not improve the image as the cage is not really a noise pattern but natural part of the image. Hence, to improve this image, a more complex filter will need to be applied.

2.6 Undoing Perspective Distortion of Planar Surface

Load and view image

```
%Load and display Image
book = imread('book.jpg');
imshow(book)
```



Obtain 4 edges of the book. (Order: Top Left, Top Right, Bottom Right, Bottom Left)

```
[x, y] = ginput(4);

%A4 Dimension
op = [0;0;210;0;210;297;0;297];
```

Matrix Calculation and Transform Image Code

```
%8x8 Matrix
A = [
    [x(1),y(1),1,0,0,0,-op(1)*x(1),-op(1)*y(1)];
    [0,0,0,x(1),y(1),1,-op(2)*x(1),-op(2)*y(1)];
    [x(2),y(2),1,0,0,0,-op(3)*x(2),-op(3)*y(2)];
    [0,0,0,x(2),y(2),1,-op(4)*x(2),-op(4)*y(2)];
    [x(3),y(3),1,0,0,0,-op(5)*x(3),-op(5)*y(3)];
    [0,0,0,x(3),y(3),1,-op(6)*x(3),-op(6)*y(3)];
    [x(4),y(4),1,0,0,0,-op(7)*x(4),-op(7)*y(4)];
    [0,0,0,x(4),y(4),1,-op(8)*x(4),-op(8)*y(4)];
];

%8x1 Matrix = 8x8 \ 8x1
u = inv(A)*op;

U = reshape([u;1],3,3)';
w = U*[x'; y'; ones(1,4)];
w = w ./ (ones(3,1)*w(3,:));
T = maketform('projective', U');
P2 = imtransform(book, T, 'XData', [0 210], 'YData', [0 297]);
```

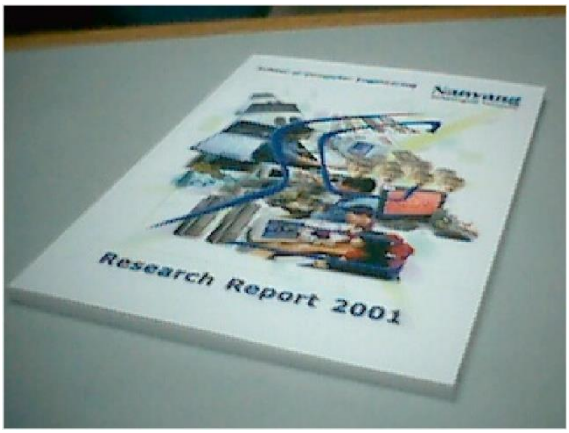
Does the transformation give you back the 4 corners of the desired image?

Yes. It transform into the dimension that we wanted. [0,0,210,0,210,297,0,297]

```
w =

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

Result after Matrix Transformation

Original Image



Transformation Image

Display the image. Is this what you expect? Comment on the quality of the transformation and suggest reasons.

Yes, Matrix transformation managed to change the perspective of view of the book from slanted to straight. However, some areas are blurred. In order to improve the results, a higher resolution image is required.