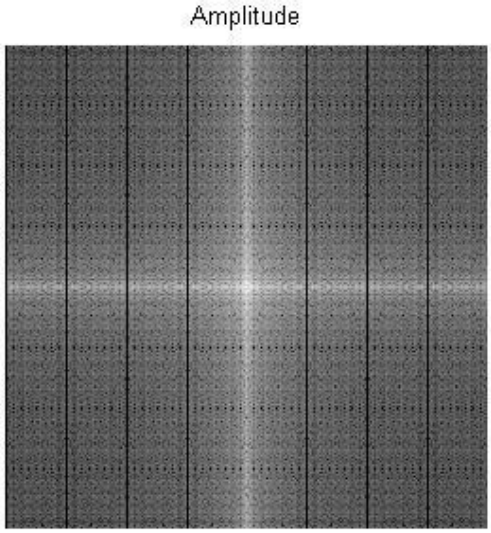
นายพีรณัฐ ธารทะเลทอง 550610530

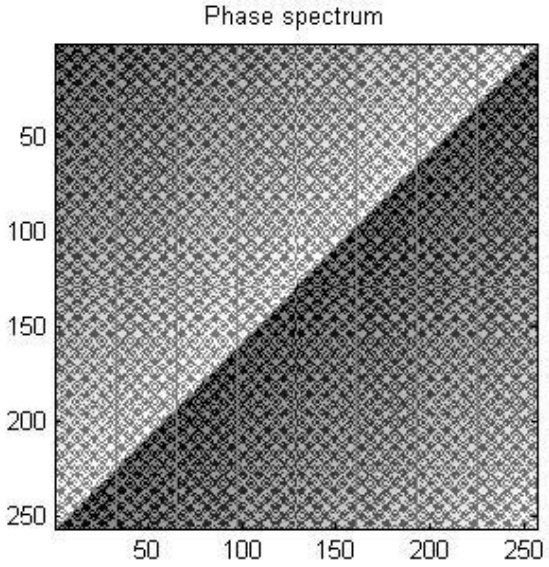
Digital Image Processing (261453)

Computer Assignment 2

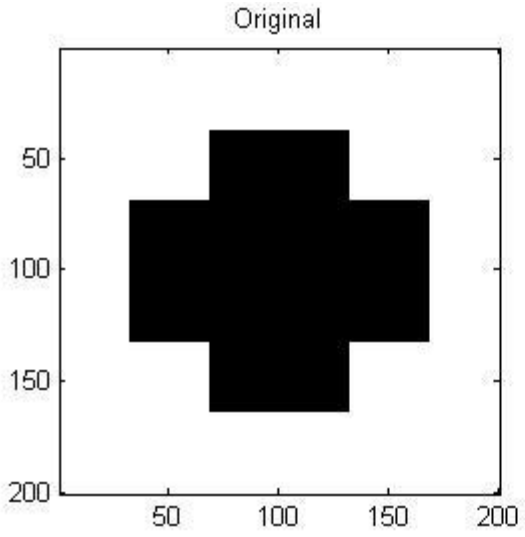
1. Properties of the Fourier Transform

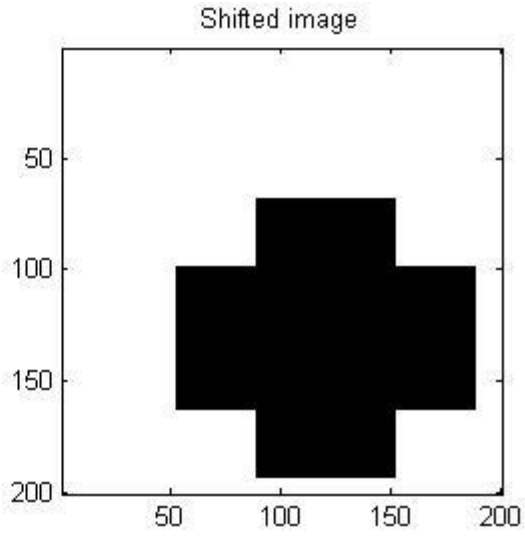
1.1 ทำการ pad รูป ให้มีขนาดเป็น 256x256 หลังจากนั้นจะทำการ shift จุดกำเนิดไปไว้ที่จุดศูนย์กลางของรูป และหา FFT จะได้ Amplitude และ Phase Spectrum ดังรูป



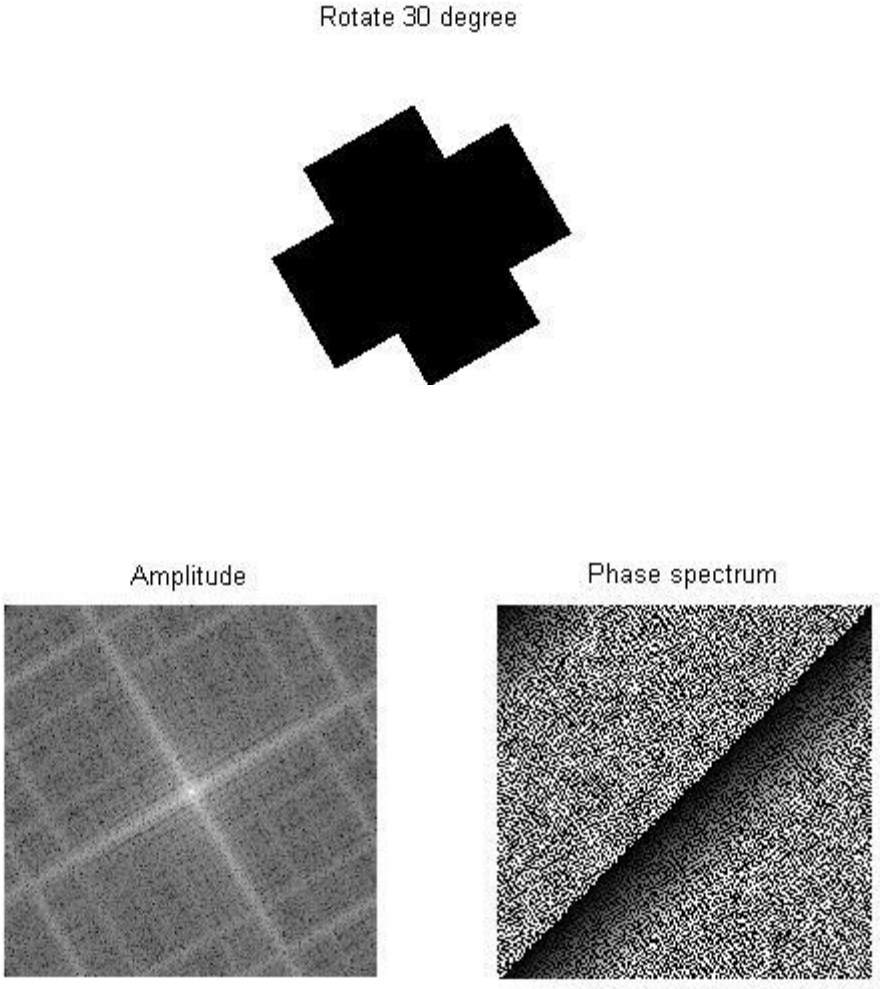


1.2 คูณ Phase Spectrum ด้วย $e^{-j2\pi(au/m + bv/n)}$ โดยแทน a = 20 , b = 30 หลังจากนั้นทำการ inverse FFT จะได้ผลลัพธ์ดังรูป

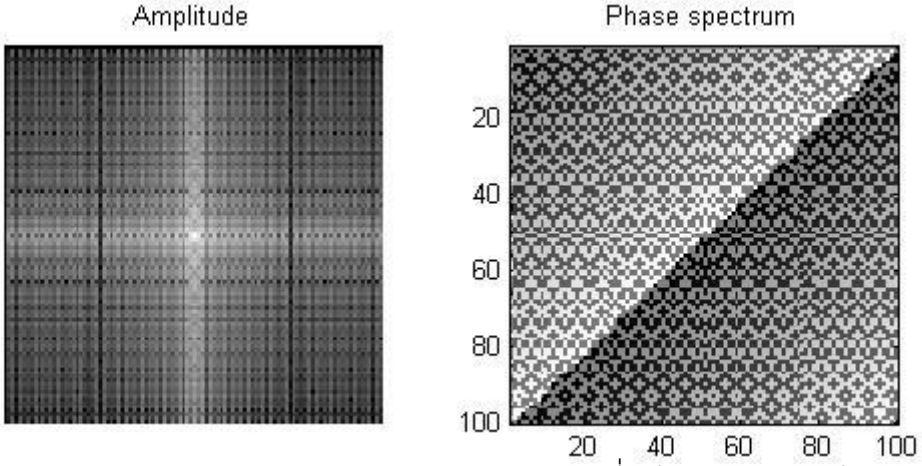




1.3 ทำการ หมุนภาพไป 30 องศา และหา Amplitude และ Phase Spectrum จะเห็นว่ารูปของ Amplitude จะหมุนในทิศทางเดียวกับรูปตามคุณสมบัติการหมุน โดยผลลัพธ์จะได้ดังรูป



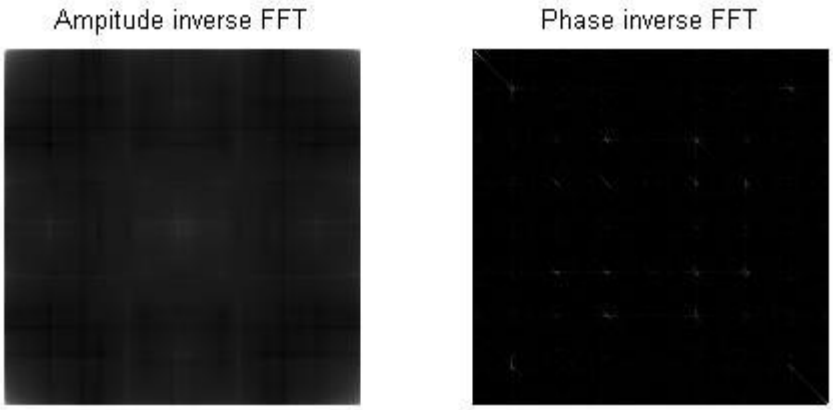
1.4 ทำการ down-sample ลงครึ่งหนึ่งให้รูปมีขนาด 100x100 จะได้ Amplitude และ Spectrum ดังรูป



จะเห็นว่าทั้ง Amplitude และ Phase Spectrum รูปร่างเหมือนเดิม เปลี่ยนไปแค่ความละเอียดเท่านั้น

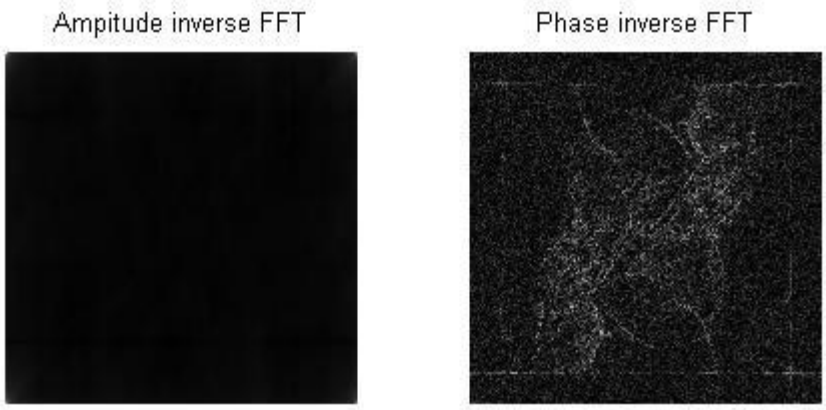
1.5 แปลง inverse FTT ของผลลัพธ์ที่ได้จากข้อ 1.1 โดยแปลงแบบไม่ใช้ข้อมูล phase และแบบไม่ใช้ข้อมูล amplitude

ได้ผลลัพธ์ดังรูป

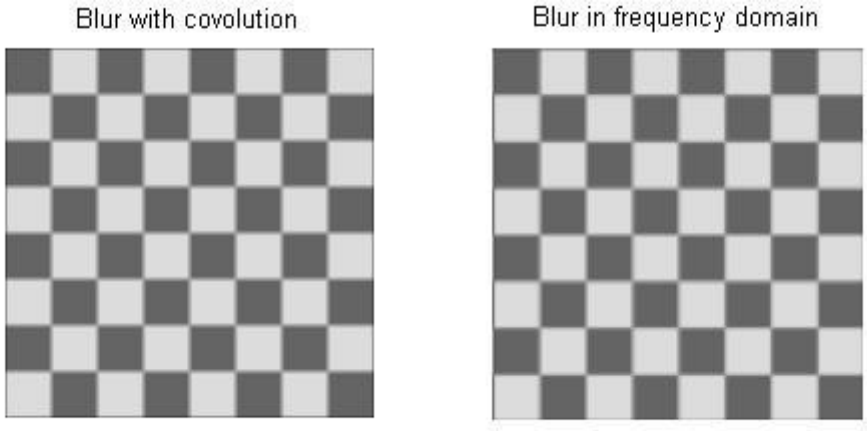


จะเห็นว่า ถ้าใช้แต่ Amplitude จะมีแต่ข้อมูลข้องความเข้มแสง โดยมีแค่ DC Component เท่านั้นที่เด่นชัด แต่ถ้าใช้แต่ phase จะทำให้เห็นขอบของวัตถุในภาพ ทำให้เห็นเป็นรูปร่างของภาพ

1.6 ทำเหมือนข้อ 1.5 แต่เปลี่ยนเป็นรูปLenna.pgm ผลลัพธ์ที่ได้ดังรูป



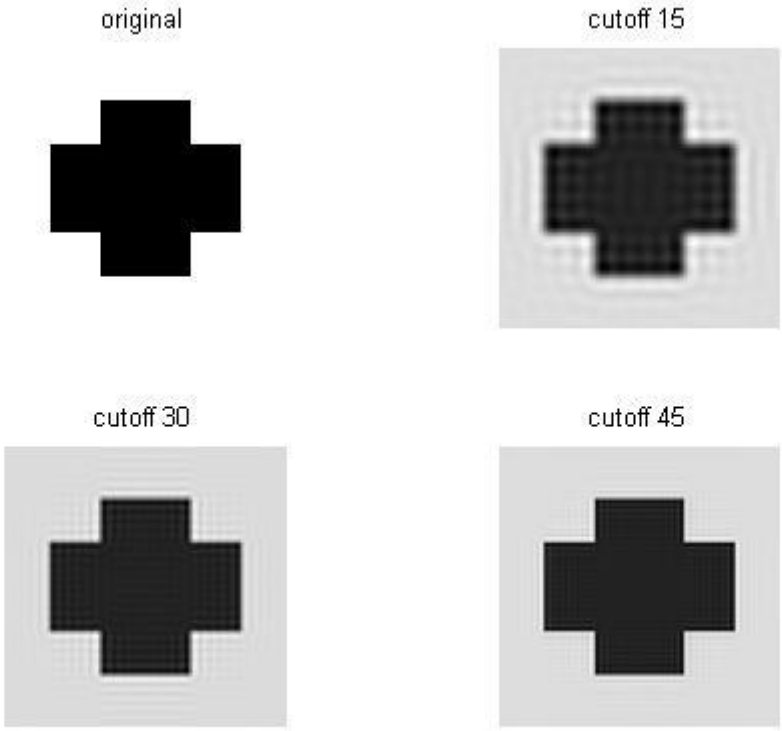
1.7 เบลอภาพโดยวิธีการ convolution ด้วย kernel ขนาด 3x3 และเบลอภาพด้วยวิธีการทำการ filter ใน frequency domain โดยการ pad kernel ให้มีขนาดเท่ากับรูป แล้วนำมาหา FFT หลังจากนั้นนำ FFT มา คูณกับ FFT ของภาพ Original แล้วนำผลลัพธ์ที่ได้มา Inverse FFT กลับเป็นรูปภาพ ผลลัพธ์ที่ได้ดังรูป



จะเห็นว่าผลลัพธ์ที่เกิดขึ้น จะมีลักษณะที่คล้ายกัน โดยภาพที่เบลอด้วยวิธีการ filter ใน frequency domain นั้น รูปจะสว่างกว่ารูปที่เบลอโดยวิธี convolution เล็กน้อย

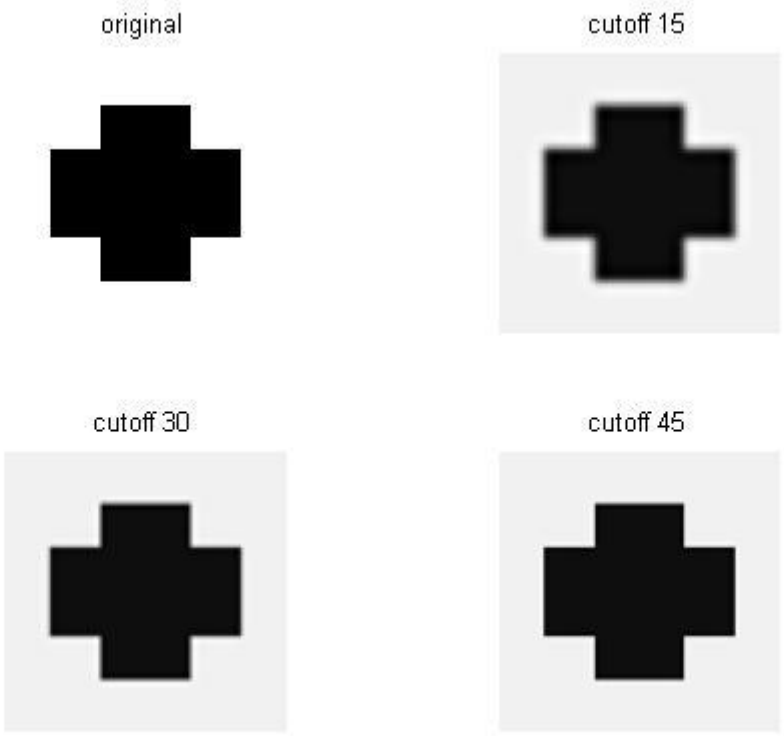
2. Filter Design

2.1) **Ideal low-pass filter** โดย cutoff frequency = 15, 30 ,45 ตามลำดับ ผลลัพธ์ดังรูป



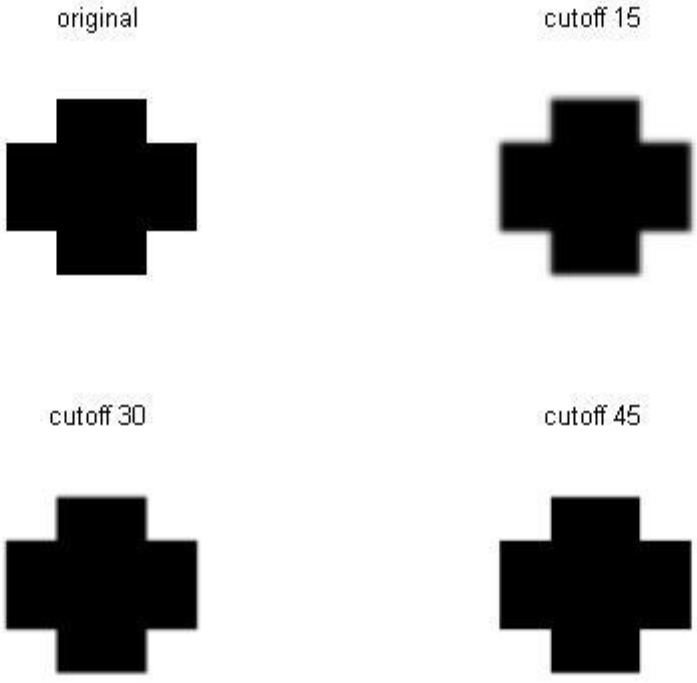
จะเห็นได้ว่า เมื่อใช้ความถี่ cutoff ต่ำๆ ภาพ output ที่ออกมาจะ เบลอจนเห็นขอบไม่ชัด และเกิด ringing effect ด้วย และถ้าใช้ความถี่ cutoff มากขึ้น ภาพก็จะคมชัดขึ้นไปเรื่อยๆ แต่ก็ยังมี ringing effectอยู่

Butterworth Low pass Filter โดย n = 2 ผลลัพธ์ดังรูป



จะเห็นได้ จะไม่เกิด ringing effect เหมือนวิธี Ideal Lowpass Filter และภาพก็จะชัดขึ้นเรื่อยๆ ตามความถี่ cutoff ที่เพิ่มขึ้น แต่พื้นหลังยังไม่เป็นสีขาว

Gaussian Low pass Filter ได้ผลลัพธ์ดังรูป



จะเห็นได้ว่า ไม่เกิด ringing effect และภาพก็จะคมชัดขึ้นเรื่อยๆ ตามความถี่ cutoff และพื้นหลังเป็นสีขาวด้วย

2.2 **Lenna.pgm**

Ideal Lowpass Filter

original

cutoff 15

cutoff 30

cutoff 45

RMS (เปรียบเทียบความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise)

Cutoff 15 = 19.8280

Cutoff 30 = 14.0525

Cutoff 45 = 12.6655

Butterworth Low pass Filter

original

cutoff 15

cutoff 30

cutoff 45

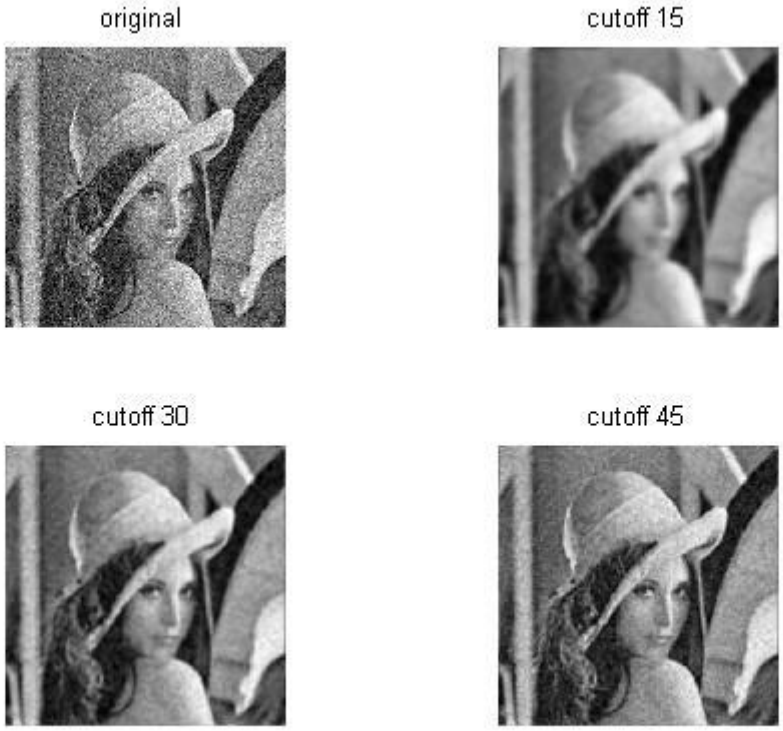
RMS (เปรียบเทียบความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise)

Cutoff 15 = 18.1013

Cutoff 30 = 12.6236

Cutoff 45 = 11.1266

Gaussian Lowpass Filter



RMS (เปรียบเทียบความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise)

Cutoff 15 = 16.8394

Cutoff 30 = 11.9274

Cutoff 45 = 11.0412

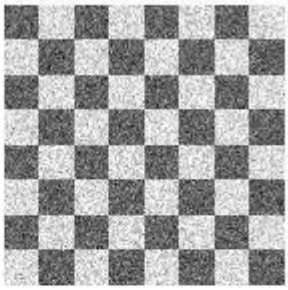
Median Filter

RMS = 12.9555

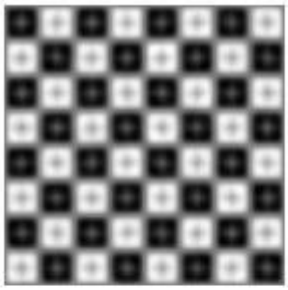
Chess.pgm

Ideal Lowpass Filter

original

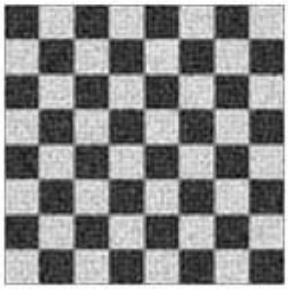


cutoff 15



cutoff 30

cutoff 45



RMS (เปรียบเทียบความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise)

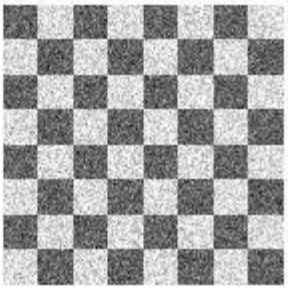
Cutoff 15 = 26.5968

Cutoff 30 = 19.3024

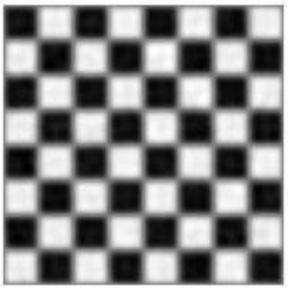
Cutoff 45 = 16.5888

Butterworth Low pass Filter

original

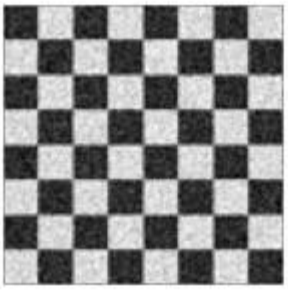


cutoff 15



cutoff 30

cutoff 45



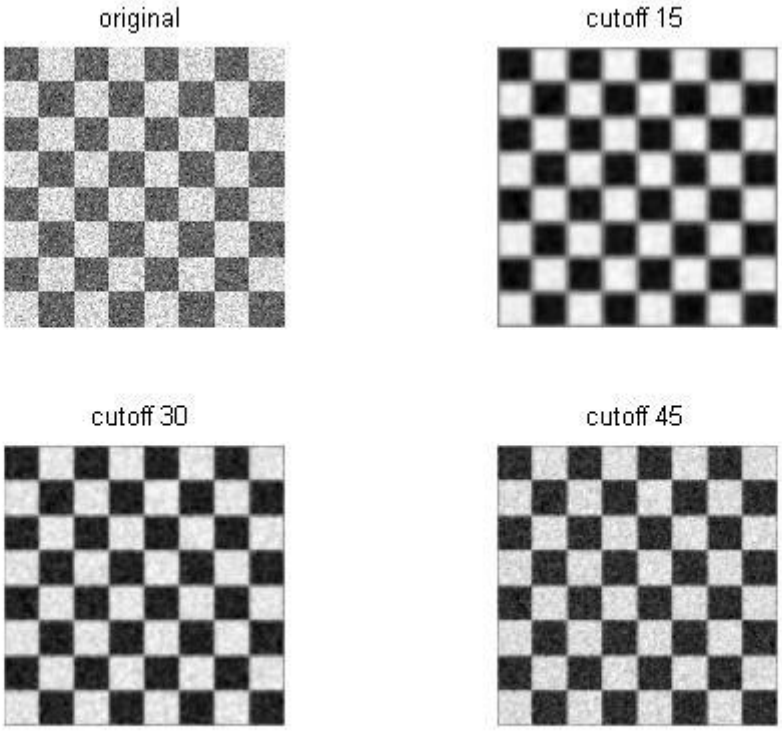
RMS (เปรียบเทียบความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise)

Cutoff 15 = 25.4389

Cutoff 30 = 18.0378

Cutoff 45 = 15.137

Gaussian Lowpass Filter



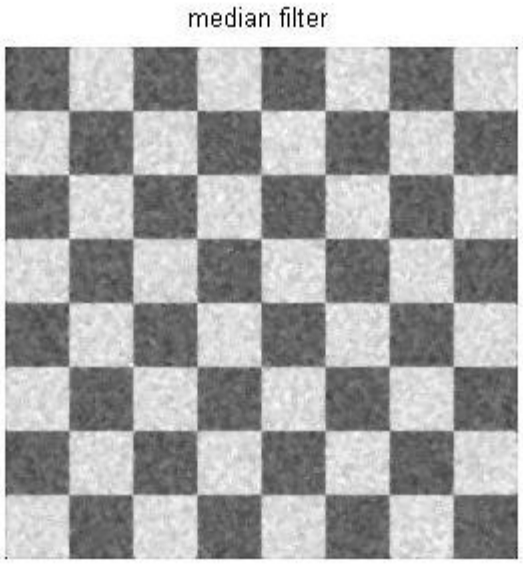
RMS (เปรียบเทียบความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise)

Cutoff 15 = 23.9303

Cutoff 30 = 17.0193

Cutoff 45 = 14.4644

Median Filter



RMS = 12.5777

Code (โค๊ดทั้งหมดอยู่ที่ https://github.com/porpeeranut/Digital_Image_Processing_Assignment2)

hw1_1.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Cross.pgm');
pad_size = (256-200)/2;
im_pad = padarray(im,[pad_size pad_size],'both');
fft = fftshift(fft2(im_pad));       %shift origin to center
ampitude = log(abs(fft)+1);      %start x axis at 1
ampitude = mat2gray(ampitude);   % scale 0-1
phase = angle(fft);
figure; subplot(1,2,1);
imshow(ampitude);
title ('Amplitude');
subplot(1,2,2);
imagesc(phase);
title('Phase spectrum');
```

hw1_2.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment
2\\Cross.pgm');
fft = fftshift(fft2(im));    %shift origin to center
ampitude = log(abs(fft)+1);      %start x axis at 1
ampitude = mat2gray(ampitude);    % scale 0-1

[u,v] = meshgrid(-100:99, -100:99);
shiftedVal = (exp(-2j*pi*( ((20*u)./200) + ((30*v)./200) ))).*fft;
im_new = ifft2(ifftshift(shiftedVal));

figure;
subplot(1,2,1);
imagesc(im);
colormap(gray);
title('Original')

subplot(1,2,2);
imagesc(im_new);
colormap(gray);
title('Shifted image');
```

hw1_3.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Cross.pgm');
one_arr = ones(size(im));
im_rotate = imrotate(im, 30, 'crop', 'bilinear');
tmp = imrotate(one_arr, 30, 'crop', 'bilinear');
im_rotate(tmp == 0) = 255;

fft = fftshift(fft2(im_rotate));    %shift origin to center
ampitude = log(abs(fft)+1);      %start x axis at 1
ampitude = mat2gray(ampitude);    % scale 0-1

figure;
imshow(im_rotate);
title ('Rotate 30 degree');

figure;
subplot(1,2,1);
imshow(ampitude);
title ('Amplitude');

subplot(1,2,2);
imshow(angle(fft));
title('Phase spectrum');
```

hw1_4.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Cross.pgm');
im_resize = imresize(im, 0.5);
fft = fftshift(fft2(im_resize));    %shift origin to center
ampitude = log(abs(fft)+1);     %start x axis at 1
ampitude = mat2gray(ampitude);   % scale 0-1

figure;
subplot(1,2,1);
imshow(ampitude);
title ('Amplitude');

subplot(1,2,2);
imagesc(angle(fft));
title('Phase spectrum');
```

hw1_5.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment
2\\Cross.pgm');
pad_size = (256-200)/2;
im_pad = padarray(im,[pad_size pad_size],'both');
fft = fftshift(fft2(im_pad));       %shift origin to center
ampitude = abs(fft);
ampitude_ifft = ifft2(ifftshift(ampitude));
phase_ifft = ifft2(ifftshift(angle(fft)));
phase_ifft = abs(phase_ifft);

figure;
subplot(1,2,1);
ampitude_ifft = mat2gray(ampitude_ifft);   % scale 0-1
imshow(ampitude_ifft);
title('Ampitude inverse FFT');

subplot(1,2,2);
phase_ifft = mat2gray(phase_ifft);   % scale 0-1
imshow(phase_ifft);
title('Phase inverse FFT');
```

hw1_6.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Lenna.pgm');
pad_size = (256-200)/2;
im_pad = padarray(im,[pad_size pad_size],'both');
fft = fftshift(fft2(im_pad));       %shift origin to center
ampitude = abs(fft);
ampitude_ifft = ifft2(ifftshift(ampitude));
phase_ifft = ifft2(ifftshift(angle(fft)));
phase_ifft = abs(phase_ifft);

figure;
subplot(1,2,1);
ampitude_ifft = mat2gray(ampitude_ifft);   % scale 0-1
imshow(ampitude_ifft);
title('Ampitude inverse FFT');

subplot(1,2,2);
phase_ifft = mat2gray(phase_ifft);   % scale 0-1
imshow(phase_ifft);
title('Phase inverse FFT');
```

hw1_7.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment
2\\Chess.pgm');
[M, N] = size(im);
im_pad = padarray(im, [1 1], 'both');
im_new = zeros(size(im));
kernel = ones(3,3);
kernel = kernel/9;    %kernel
[Mk, Nk] = size(kernel);

%convolute
for i = 1:M
    for j = 1:N
        result = 0;
        for x = 1:Mk
            for y = 1:Nk
                tmp = double(kernel(x, y)).*double(im_pad(x+i-1, y+j-1));
                result = result+tmp;
            end
        end
        im(i, j) = double(result);
    end
end

fft = fftshift(fft2(im));   %shift origin to center
pad_size = 256-3;
kernel_pad = padarray(kernel, [pad_size pad_size], 'post');
kernel_fft = fftshift(fft2(kernel_pad));
result = kernel_fft.*fft;
```

```
im_ifft =  ifft2(ifftshift(result));


figure;
subplot(1,2,1);
imshow(im);
title('Blur with covolution');

subplot(1,2,2);
imshow(uint8(abs(im_ifft)));
title('Blur in frequency domain');
```

hw2_1.m

```
im = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Cross.pgm');
[M, N] = size(im);
center_x = (M-1)/2;
center_y = (N-1)/2;
[u, v] = meshgrid(-center_x:center_x, -center_y:center_y);
D = sqrt(u.^2 + v.^2);
cutoff15 = 15;
cutoff30 = 30;
cutoff45 = 45;
fft = fftshift(fft2(im));    %shift origin to center

%ideal low pass
H15 = double(D <= cutoff15);
H30 = double(D <= cutoff30);
H45 = double(D <= cutoff45);
G15 = H15.*fft;
G30 = H30.*fft;
G45 = H45.*fft;
%inverse fft
lowpass15 = ifft2(ifftshift(G15));
lowpass30 = ifft2(ifftshift(G30));
lowpass45 = ifft2(ifftshift(G45));

%butterworth lowpass filter
n2 = 2;
H_B15 = double(1./(1+(D./cutoff15).^(2*n2)));
H_B30 = double(1./(1+(D./cutoff30).^(2*n2)));
```

```
H_B45 = double(1./(1+(D./cutoff45).^(2*n2)));
G_B15 = H_B15.*fft;
G_B30 = H_B30.*fft;
G_B45 = H_B45.*fft;
%inverse fft
BLP15 = ifft2(ifftshift(G_B15));
BLP30 = ifft2(ifftshift(G_B30));
BLP45 = ifft2(ifftshift(G_B45));

%gaussian lowpass filter
H_G15 = double(exp((-(D).^2)./(2.*((cutoff15).^2))));
H_G30 = double(exp((-(D).^2)./(2.*((cutoff30).^2))));
H_G45 = double(exp((-(D).^2)./(2.*((cutoff45).^2))));
G_G15 = H_G15.*fft;
G_G30 = H_G30.*fft;
G_G45 = H_G45.*fft;
%inverse fft
GLP15 = ifft2(ifftshift(G_G15));
GLP30 = ifft2(ifftshift(G_G30));
GLP45 = ifft2(ifftshift(G_G45));

figure('Name', 'Ideal lowpass filter');
subplot(2,2,1);
imshow(im);
title('original');

subplot(2,2,2);
imshow(real(lowpass15),[]);
title('cutoff 15');
```

```
subplot(2,2,3);
imshow(real(lowpass30),[]);
title('cutoff 30');

subplot(2,2,4);
imshow(real(lowpass45),[]);
title('cutoff 45');



figure('Name', 'Butterworth lowpass filter n=2');
subplot(2,2,1);
imshow(im);
title('original');

subplot(2,2,2);
imshow(real(BLP15),[]);
title('cutoff 15');

subplot(2,2,3);
imshow(real(BLP30),[]);
title('cutoff 30');

subplot(2,2,4);
imshow(real(BLP45),[]);
title('cutoff 45');



figure('Name', 'Gaussian lowpass filter');
```

```
subplot(2,2,1);
imshow(im);
title('original');

subplot(2,2,2);
imshow(real(GLP15),[]);
title('cutoff 15');

subplot(2,2,3);
imshow(real(GLP30),[]);
title('cutoff 30');

subplot(2,2,4);
imshow(real(GLP45),[]);
title('cutoff 45');
```

hw2_2.m

```
%im_noise = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Lenna_noise.pgm');
%im_original = double(imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Lenna.pgm'));
im_noise = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Chess_noise.pgm');
im_original = double(imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Chess.pgm'));
[M, N] = size(im_noise);
center_x = (M-1)/2;
center_y = (N-1)/2;
[u, v] = meshgrid(-center_x:center_x, -center_y:center_y);
D = sqrt(u.^2 + v.^2);
cutoff15 = 15;
cutoff30 = 30;
cutoff45 = 45;
fft = fftshift(fft2(im_noise));     %shift origin to center

%ideal low pass
H15 = double(D <= cutoff15);
H30 = double(D <= cutoff30);
H45 = double(D <= cutoff45);
G15 = H15.*fft;
G30 = H30.*fft;
G45 = H45.*fft;
%inverse fft
lowpass15 = real(uint8(ifft2(ifftshift(G15))));
lowpass30 = uint8(ifft2(ifftshift(G30)));
```

```
lowpass45 = uint8(ifft2(ifftshift(G45)));

% RMS ideal low pass
%cutoff 15
lowpass15_err = im_original - double(lowpass15);
MSE15 = (sum(sum(lowpass15_err.^2)))/(M * N);
RMS15 = sqrt(MSE15)
%cutoff 30
lowpass30_err = im_original - double(lowpass30);
MSE30 = (sum(sum(lowpass30_err.^2)))/(M * N);
RMS30 = sqrt(MSE30)
%cutoff 45
lowpass45_err = im_original - double(lowpass45);
MSE45 = (sum(sum(lowpass45_err.^2)))/(M * N);
RMS45 = sqrt(MSE45)

%butterworth lowpass filter
n2 = 2;
H_B15 = double(1./(1+(D./cutoff15).^(2*n2)));
H_B30 = double(1./(1+(D./cutoff30).^(2*n2)));
H_B45 = double(1./(1+(D./cutoff45).^(2*n2)));
G_B15 = H_B15.*fft;
G_B30 = H_B30.*fft;
G_B45 = H_B45.*fft;
%inverse fft
BLP15 = uint8(ifft2(ifftshift(G_B15)));
BLP30 = uint8(ifft2(ifftshift(G_B30)));
BLP45 = uint8(ifft2(ifftshift(G_B45)));
```

```
% RMS butterworth
%cutoff 15
BLP15_err = im_original - double(BLP15);
MSE15 = (sum(sum(BLP15_err.^2)))/(M * N);
RMSblp15 = sqrt(MSE15)
%cutoff 30
BLP30_err = im_original - double(BLP30);
MSE30 = (sum(sum(BLP30_err.^2)))/(M * N);
RMSblp30 = sqrt(MSE30)
%cutoff 45
BLP45_err = im_original - double(BLP45);
MSE45 = (sum(sum(BLP45_err.^2)))/(M * N);
RMSblp45 = sqrt(MSE45)

%gaussian lowpass filter
H_G15 = double(exp((-(D).^2)./(2.*((cutoff15).^2))));
H_G30 = double(exp((-(D).^2)./(2.*((cutoff30).^2))));
H_G45 = double(exp((-(D).^2)./(2.*((cutoff45).^2))));
G_G15 = H_G15.*fft;
G_G30 = H_G30.*fft;
G_G45 = H_G45.*fft;
%inverse fft
GLP15 = ifft2(ifftshift(G_G15));
GLP30 = ifft2(ifftshift(G_G30));
GLP45 = ifft2(ifftshift(G_G45));

% RMS gaussian
%cutoff 15
GLP15_err = im_original - double(GLP15);
```

```
MSE15 = (sum(sum(GLP15_err.^2)))/(M * N);
RMSglp15 = sqrt(MSE15)
%cutoff 30
GLP30_err = im_original - double(GLP30);
MSE30 = (sum(sum(GLP30_err.^2)))/(M * N);
RMSglp30 = sqrt(MSE30)
%cutoff 45
GLP45_err = im_original - double(GLP45);
MSE50 = (sum(sum(GLP45_err.^2)))/(M * N);
RMSglp45 = sqrt(MSE50)

figure('Name', 'Ideal lowpass filter');
subplot(2,2,1);
imshow(im_noise);
title('original');

subplot(2,2,2);
imshow(real(lowpass15),[]);
title('cutoff 15');

subplot(2,2,3);
imshow(real(lowpass30),[]);
title('cutoff 30');

subplot(2,2,4);
imshow(real(lowpass45),[]);
title('cutoff 45');
```

```
figure('Name', 'Butterworth lowpass filter n=2');
subplot(2,2,1);
imshow(im_noise);
title('original');

subplot(2,2,2);
imshow(real(BLP15),[]);
title('cutoff 15');

subplot(2,2,3);
imshow(real(BLP30),[]);
title('cutoff 30');

subplot(2,2,4);
imshow(real(BLP45),[]);
title('cutoff 45');


figure('Name', 'Gaussian lowpass filter');
subplot(2,2,1);
imshow(im_noise);
title('original');

subplot(2,2,2);
imshow(real(GLP15),[]);
title('cutoff 15');

subplot(2,2,3);
imshow(real(GLP30),[]);
```

```
title('cutoff 30');

subplot(2,2,4);
imshow(real(GLP45),[]);
title('cutoff 45');

%im_noise = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Lenna_noise.pgm');
%im_original = double(imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Lenna.pgm'));
im_noise = imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Chess_noise.pgm');
im_original = double(imread('D:\\Google Drive\\CMU\\3rd\\semester 2\\261453 Digital Image Processing\\Assignment 2\\Chess.pgm'));

im_pad = padarray(im_noise, [1 1],'both');
[M, N] = size(im_noise);

for i = 1:M
    for j = 1:N
        window = zeros(9,1);
        idx=1;
        for x = 1:3
            for y = 1:3
                window(idx) = im_pad(x+i-1, y+j-1);
                idx = idx+1;
            end
        end
        window = sort(window);
```

```
        im_new(i,j) = window(5);
    end
end

%MSE
im_new = uint8(im_new);
im_original = double(im_original);
im_new_dou = double(im_new);
[M N] = size(im_original);
err = im_original - im_new_dou;
MSE = (sum(sum(err.^2)))/(M * N);
RMSmed = sqrt(MSE)

figure('Name', 'median filter');
imshow(im_new);
title('median filter');
```