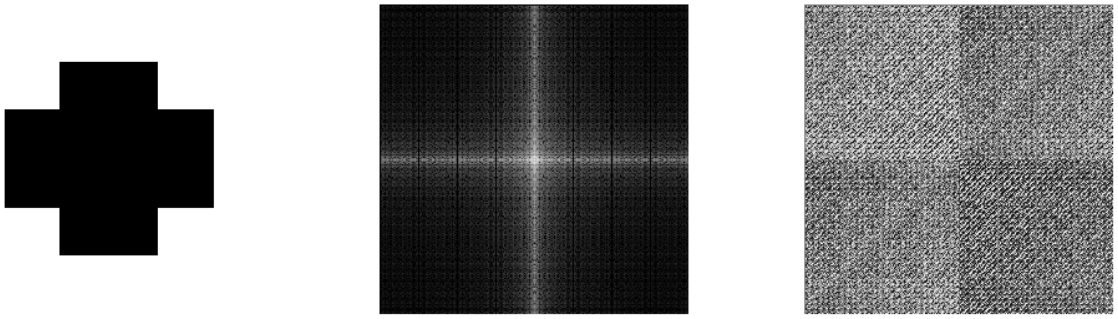
Digital Image Processing (261453)

Computer Assignment 2

วรธรรม ฆังตระกูล 570610597

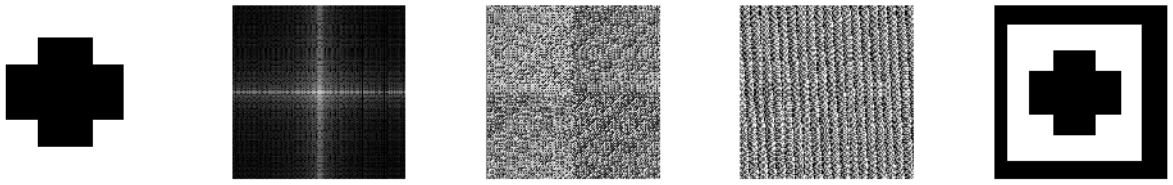
1.1)



จากทางซ้าย

1. รูปต้นฉบับ
2. Magnitude ของ Fourier transform ของรูป ซึ่งผ่านฟังก์ชั่น log
3. Phase ของ Fourier transform ของรูป

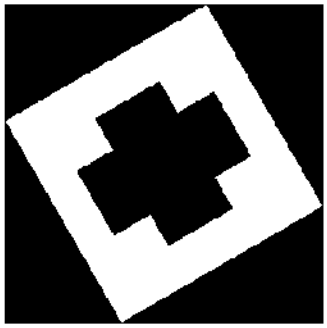
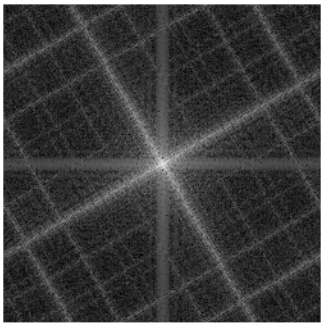
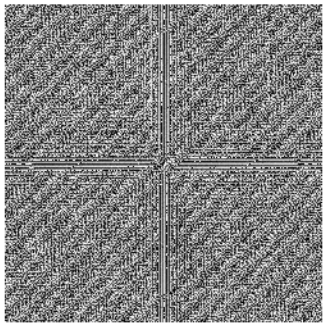
1.2)



จากทางซ้าย

1. รูปต้นฉบับ
2. Magnitude ของ Fourier transform ของรูป ซึ่งผ่านฟังก์ชั่น log
3. Phase ของ Fourier transform ของรูป
4. Phase ของ Fourier transform ของรูป ซึ่งผ่านการคูณกับเมตริกซ์ H
5. รูปที่ได้จากการ inverse Fourier transform จะเห็นว่ามีการเลื่อนไป (20, 30) ตามแกน x, y

1.3)

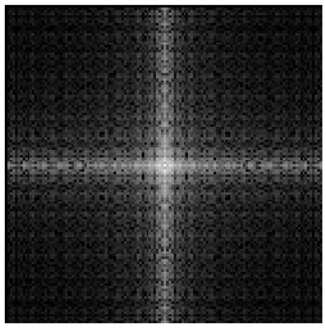
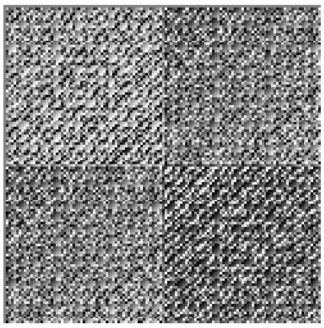
จากทางซ้าย

1. รูปต้นฉบับที่ถูกหมุน 30 องศาทวนเข็มนาฬิกา
2. Magnitude ของ Fourier transform ของรูปที่ถูกหมุน ซึ่งผ่านฟังก์ชั่น log
3. Phase ของ Fourier transform ของรูปที่ถูกหมุน

วิเคราะห์

เห็นได้ชัดว่า magnitude ของ Fourier transform หมุนไปในมุมเดียวกับรูปต้นฉบับ ส่วน phase นั้นเปลี่ยนแปลงแต่ไม่ได้หมุนตามมุมแบบ magnitude

1.4)

จากทางซ้าย

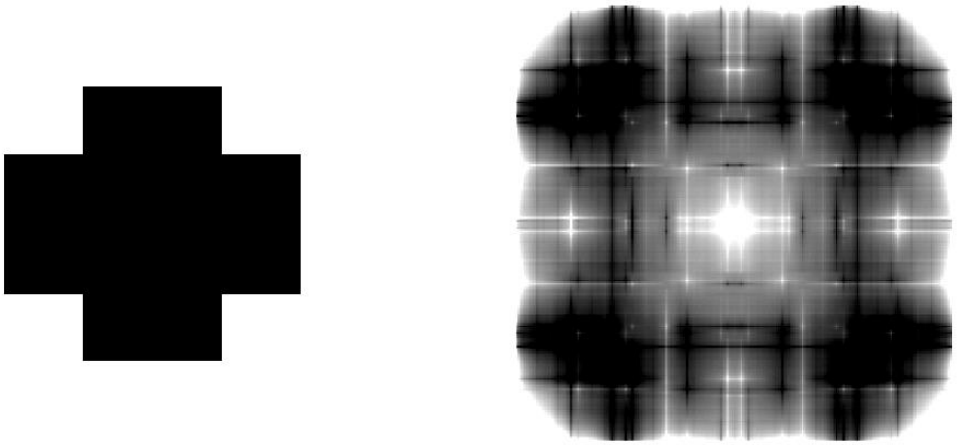
1. รูปต้นฉบับที่ถูกลดขนาดลงเหลือ 100x100
2. Magnitude ของ Fourier transform ของรูปที่ถูกลดขนาด ซึ่งผ่านฟังก์ชั่น log
3. Phase ของ Fourier transform ของรูปที่ถูกลดขนาด

วิเคราะห์

ข้อมูล magnitude และ phase ของ Fourier transform ก็ถูก down sample ในอัตราส่วนเดียวกับรูปต้นฉบับ

1.5)

จากทางซ้าย 1. รูปต้นฉบับ 2. รูปที่ได้จากการ inverse Fourier transform โดยไม่มี magnitude

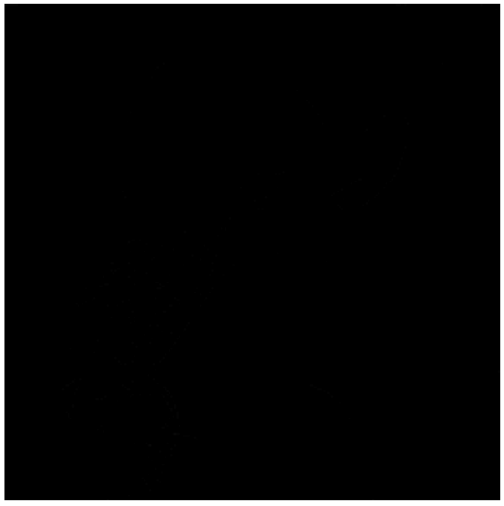


จากทางซ้าย 1. รูปต้นฉบับ 2. รูปที่ได้จากการ inverse Fourier transform โดยไม่มี phase

วิเคราะห์

รูปที่ได้จากการ inverse Fourier transform โดยไม่มี magnitude ก็คือ inverse Fourier transform ของฟังก์ชั่นค่าคงที่ 0 นั่นเอง ส่วนรูปที่ได้จากการ inverse Fourier transform โดยไม่มี phase คือ phase = 0 เสมอ รูปที่ได้ก็คือผลจากการที่ phase = 0 เสมอ

1.6)



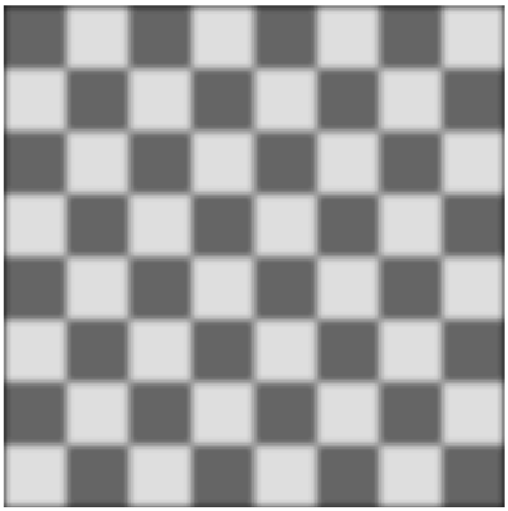
จากทางซ้าย 1. รูปต้นฉบับ 2. รูปที่ได้จากการ inverse Fourier transform โดยไม่มี magnitude

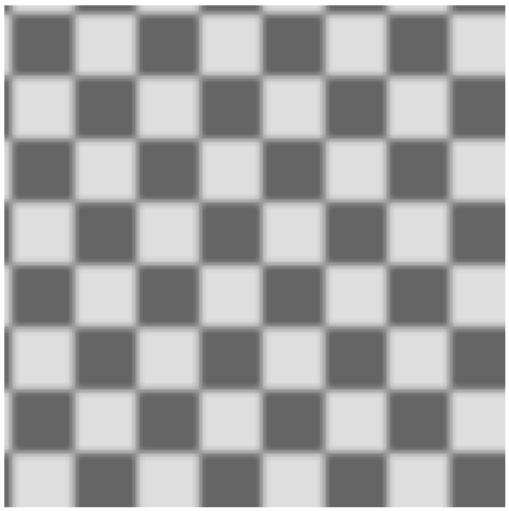
จากทางซ้าย 1. รูปต้นฉบับ 2. รูปที่ได้จากการ inverse Fourier transform โดยไม่มี phase

วิเคราะห์

รูปที่ได้จากการ inverse Fourier transform โดยไม่มี magnitude ก็คือ inverse Fourier transform ของฟังก์ชั่นค่าคงที่ 0 นั่นเอง ส่วนรูปที่ได้จากการ inverse Fourier transform โดยไม่มี phase คือ phase = 0 เสมอ รูปที่ได้ก็คือผลจากการที่ phase = 0 เสมอ

1.7)





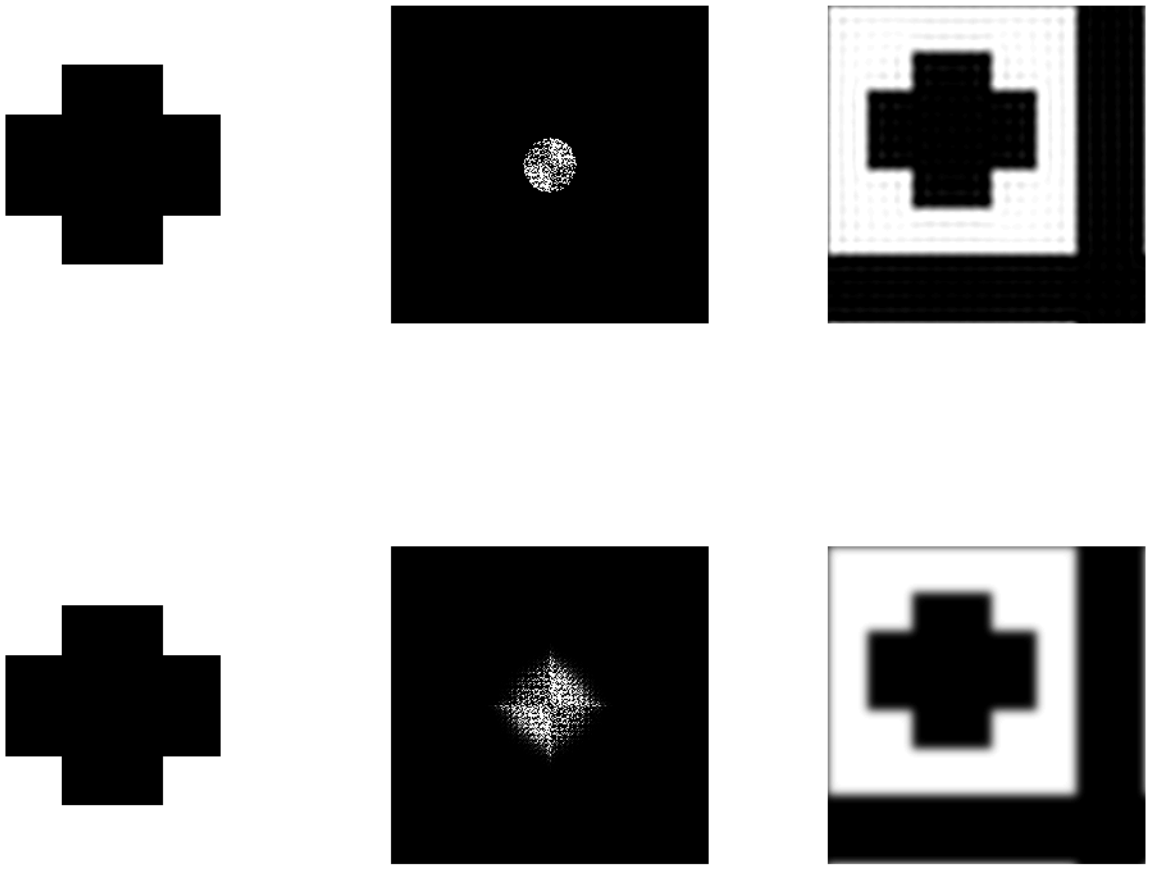
จากทางซ้าย

1. รูปหลังผ่านการ convolute ด้วย Gaussian kernel
2. รูปที่ได้จากการ inverse Fourier transform ของผลคูณระหว่าง transform ของรูปต้นฉบับกับ transform ของ Gaussian kernel

วิเคราะห์

จากที่ Fourier transform ของ Gaussian kernel ยังคงความเป็น Gaussian ทำให้รูปที่ได้ถูกเบลอไปในแบบเดียวกัน ยกเว้นตรงขอบที่การ convolute จะ pad 0 ทำให้มีสีดำปนเข้ามา ส่วนรูปที่ได้จากการ inverse Fourier transform มีความ symmetric เพราะเป็น periodic

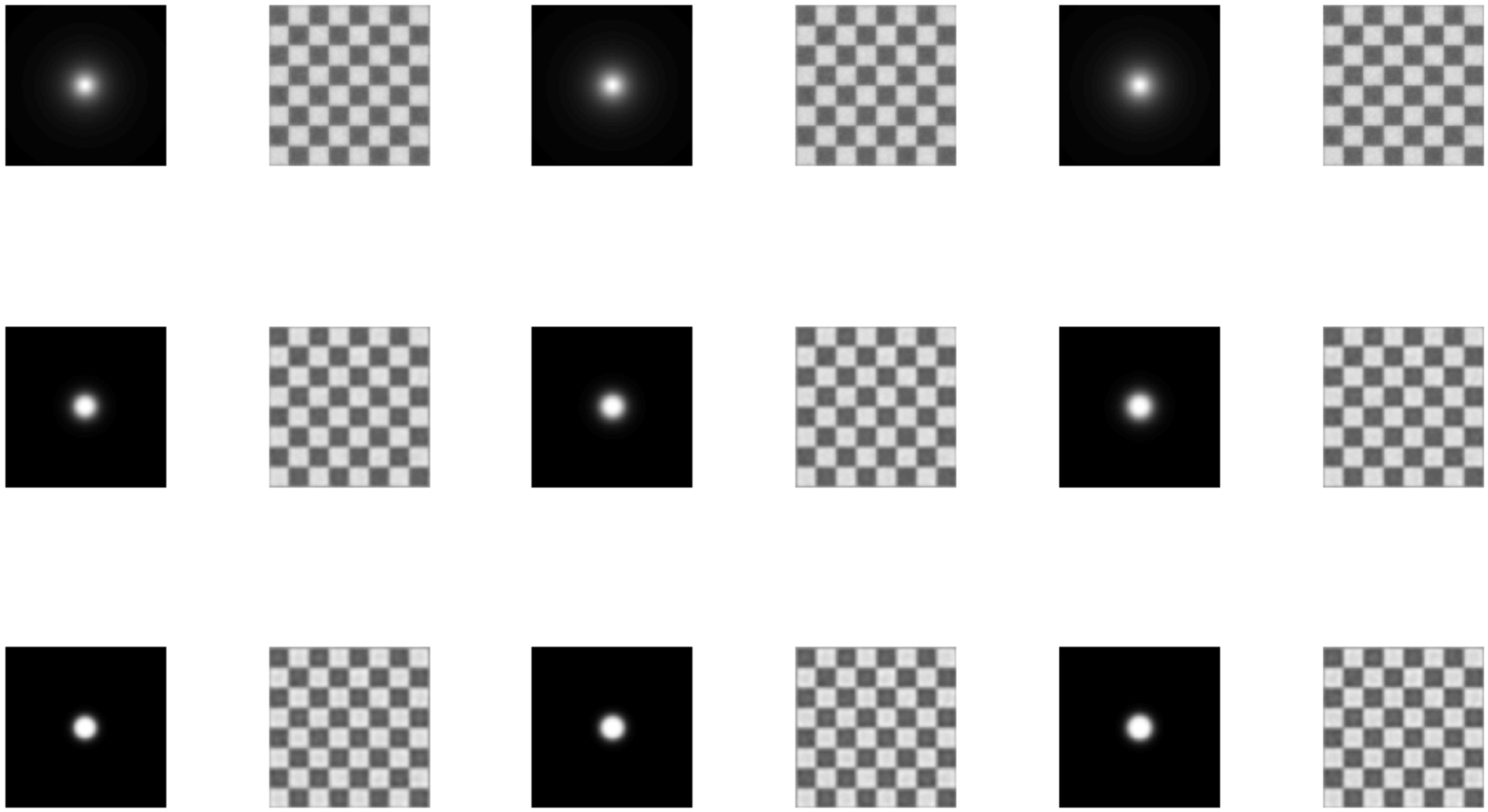
2.1)



ลำดับจากซ้ายไปขวาและบนลงล่าง

1. รูปต้นฉบับ
2. Fourier transform ของรูปต้นฉบับ ที่ผ่าน ideal low-pass filter
3. รูปที่ได้จากการ inverse Fourier transform ดังกล่าว
4. รูปต้นฉบับ
5. Fourier transform ของรูปต้นฉบับที่ผ่าน Gaussian low-pass filter
6. รูปที่ได้จากการ inverse Fourier transform ดังกล่าว

2.2)

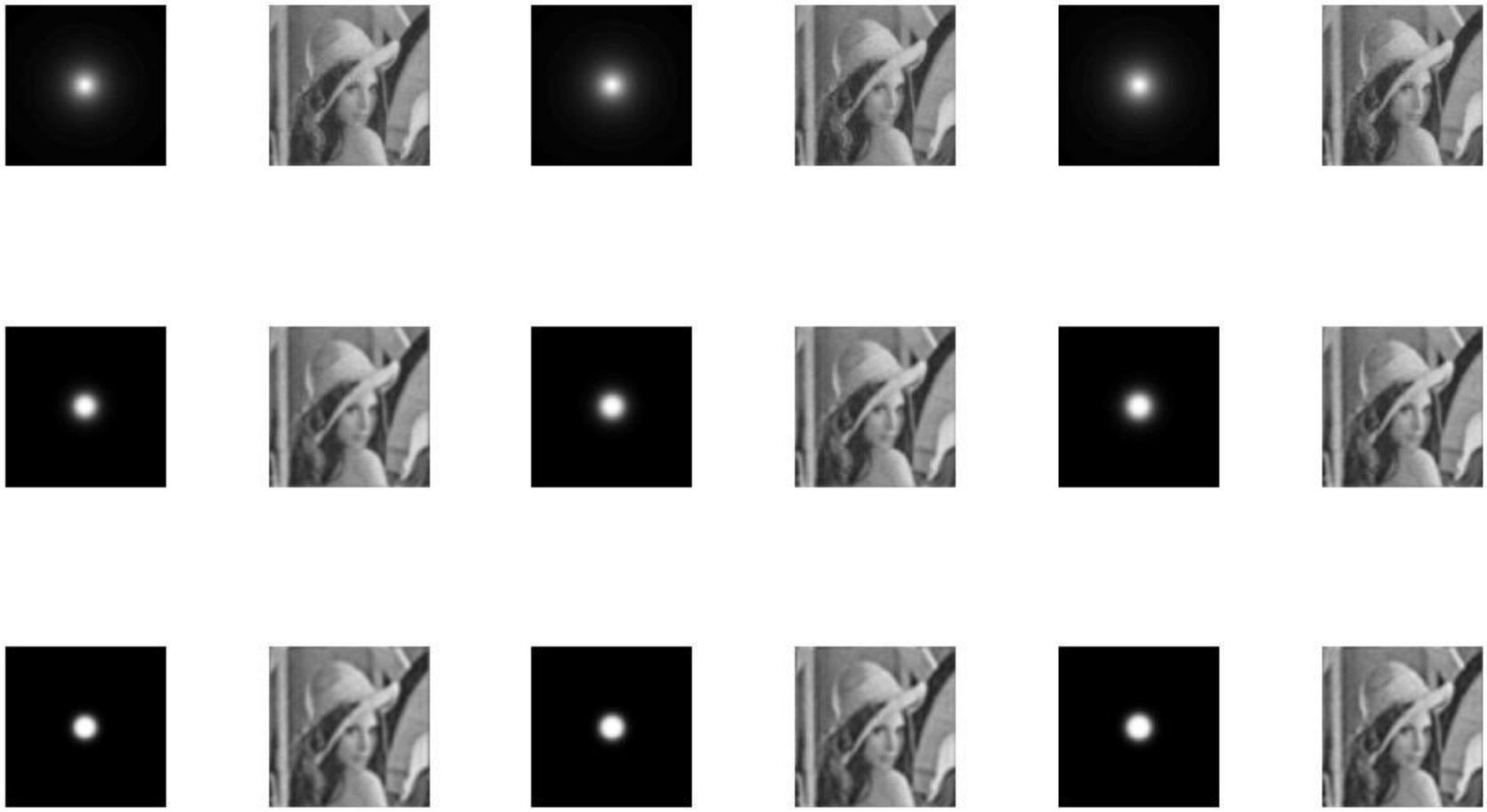


การลด noise ของรูป Chess ผ่าน Butterworth low-pass filter โดยค่า n ขึ้นอยู่กับลำดับแถวและค่า D_0 ขึ้นอยู่กับลำดับคอลั่มน์

ค่าความผิดพลาด RMS ของรูปที่ผ่าน Butterworth low-pass filter เมื่อเทียบกับรูปต้นฉบับที่ไม่มี noise คือ (ลำดับจากซ้ายไปขวาและบนลงล่าง)

1. เซ็ตค่า n = 1 และ D_0 = 18 ให้ค่า RMS = 8.25
2. เซ็ตค่า n = 1 และ D_0 = 19 ให้ค่า RMS = 8.07
3. เซ็ตค่า n = 1 และ D_0 = 20 ให้ค่า RMS = 7.91
4. เซ็ตค่า n = 2 และ D_0 = 18 ให้ค่า RMS = 7.37
5. เซ็ตค่า n = 2 และ D_0 = 19 ให้ค่า RMS = 7.24
6. เซ็ตค่า n = 2 และ D_0 = 20 ให้ค่า RMS = 7.12 (local minimum)
7. เซ็ตค่า n = 3 และ D_0 = 18 ให้ค่า RMS = 7.69
8. เซ็ตค่า n = 3 และ D_0 = 19 ให้ค่า RMS = 7.56
9. เซ็ตค่า n = 3 และ D_0 = 20 ให้ค่า RMS = 7.42

2.2)

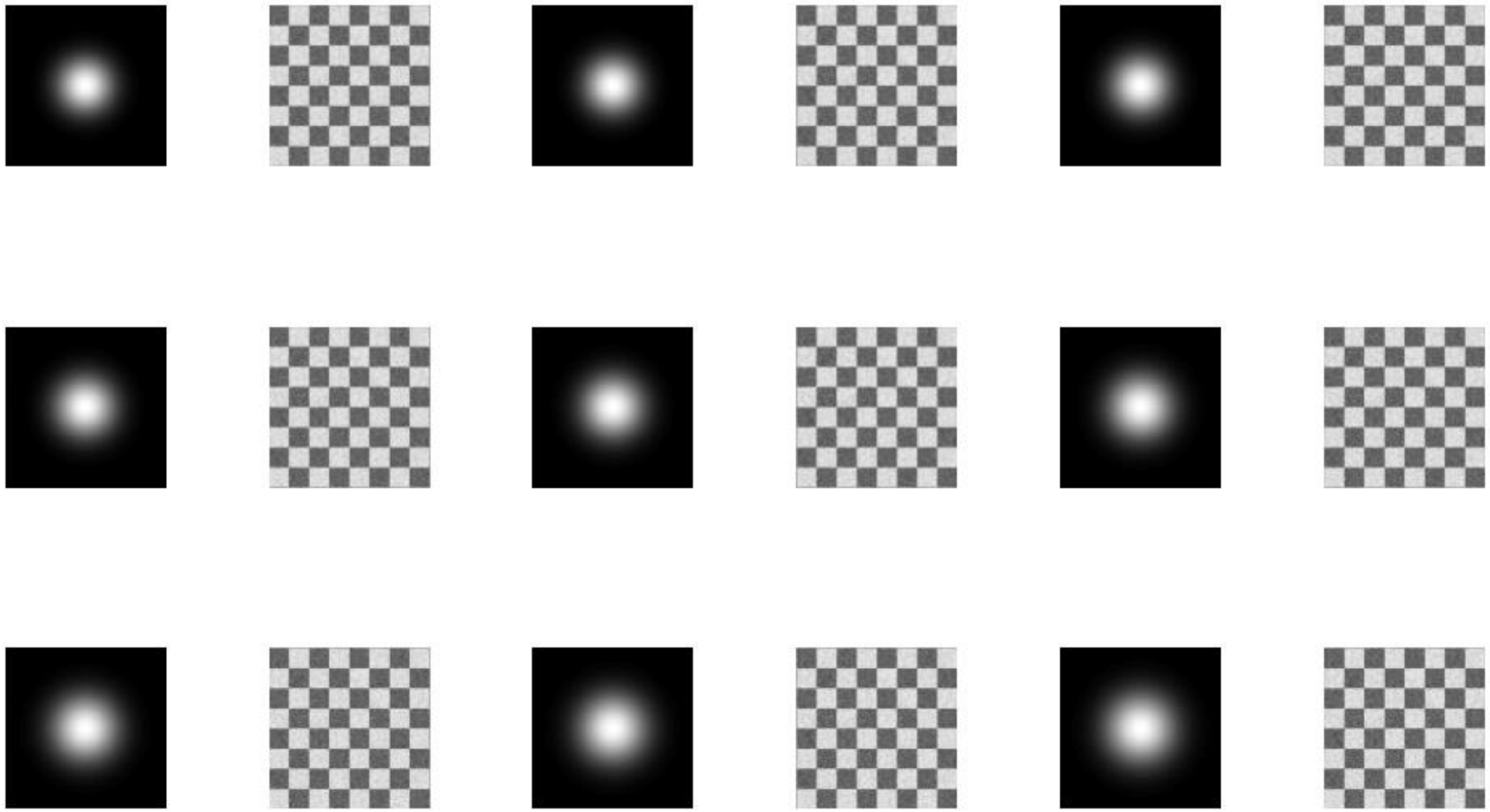


การลด noise ของรูป Lenna ผ่าน Butterworth low-pass filter โดยค่า n ขึ้นอยู่กับลำดับแถวและค่า D_0 ขึ้นอยู่กับลำดับคอลัมน์

ค่าความผิดพลาด RMS ของรูปที่ผ่าน Butterworth low-pass filter เมื่อเทียบกับรูปต้นฉบับที่ไม่มี noise คือ (ลำดับจากซ้ายไปขวาและบนลงล่าง)

1. เซ็ตค่า n = 1 และ D_0 = 18 ให้ค่า RMS = 6.98
2. เซ็ตค่า n = 1 และ D_0 = 19 ให้ค่า RMS = 6.86
3. เซ็ตค่า n = 1 และ D_0 = 20 ให้ค่า RMS = 6.75
4. เซ็ตค่า n = 2 และ D_0 = 18 ให้ค่า RMS = 6.59
5. เซ็ตค่า n = 2 และ D_0 = 19 ให้ค่า RMS = 6.49
6. เซ็ตค่า n = 2 และ D_0 = 20 ให้ค่า RMS = 6.41 (local minimum)
7. เซ็ตค่า n = 3 และ D_0 = 18 ให้ค่า RMS = 6.76
8. เซ็ตค่า n = 3 และ D_0 = 19 ให้ค่า RMS = 6.66
9. เซ็ตค่า n = 3 และ D_0 = 20 ให้ค่า RMS = 6.57

2.2)

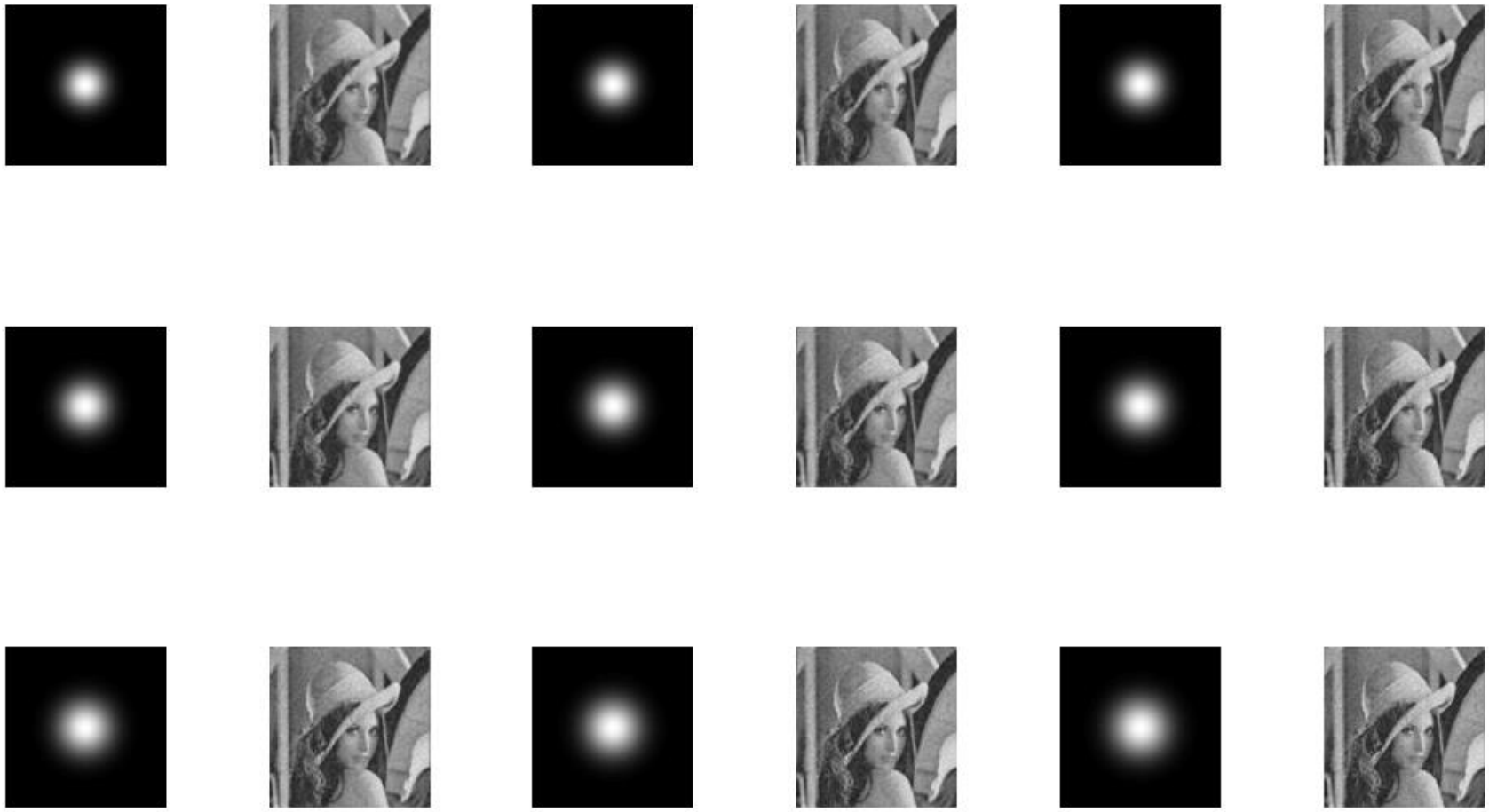


การลด noise ของรูป Chess ผ่าน Gaussian low-pass filter โดยค่า D_0 (standard deviation) ขึ้นอยู่กับลำดับแถวและคอลั่มน์

ค่าความผิดพลาด RMS ของรูปที่ผ่าน Gaussian low-pass filter เมื่อเทียบกับรูปต้นฉบับที่ไม่มี noise คือ (ลำดับจากซ้ายไปขวาและบนลงล่าง)

1. เซ็ตค่า D_0 = 30 ให้ค่า RMS = 6.34
2. เซ็ตค่า D_0 = 31 ให้ค่า RMS = 6.31
3. เซ็ตค่า D_0 = 32 ให้ค่า RMS = 6.29
4. เซ็ตค่า D_0 = 33 ให้ค่า RMS = 6.27
5. เซ็ตค่า D_0 = 34 ให้ค่า RMS = 6.26
6. เซ็ตค่า D_0 = 35 ให้ค่า RMS = 6.26 (local minimum)
7. เซ็ตค่า D_0 = 36 ให้ค่า RMS = 6.26
8. เซ็ตค่า D_0 = 37 ให้ค่า RMS = 6.27
9. เซ็ตค่า D_0 = 38 ให้ค่า RMS = 6.28

2.2)

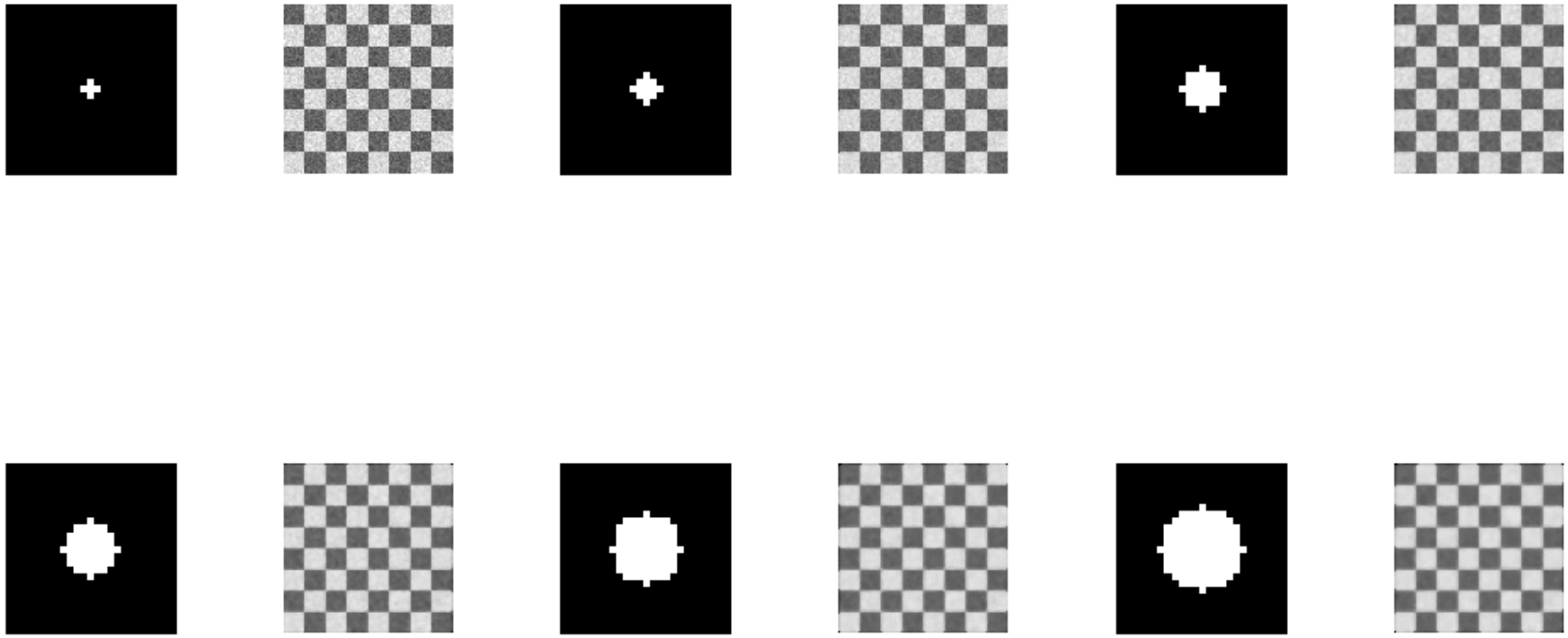


การลด noise ของรูป Lenna ผ่าน Gaussian low-pass filter โดยค่า D_0 (standard deviation) ขึ้นอยู่กับลำดับแถวและคอลัมน์

ค่าความผิดพลาด RMS ของรูปที่ผ่าน Gaussian low-pass filter เมื่อเทียบกับรูปต้นฉบับที่ไม่มี noise คือ (ลำดับจากซ้ายไปขวาและบนลงล่าง)

1. เซ็ตค่า D_0 = 25 ให้ค่า RMS = 6.04
2. เซ็ตค่า D_0 = 26 ให้ค่า RMS = 6.01
3. เซ็ตค่า D_0 = 27 ให้ค่า RMS = 5.99
4. เซ็ตค่า D_0 = 28 ให้ค่า RMS = 5.97
5. เซ็ตค่า D_0 = 29 ให้ค่า RMS = 5.96
6. เซ็ตค่า D_0 = 30 ให้ค่า RMS = 5.95
7. เซ็ตค่า D_0 = 31 ให้ค่า RMS = 5.95 (local minimum)
8. เซ็ตค่า D_0 = 32 ให้ค่า RMS = 5.95
9. เซ็ตค่า D_0 = 33 ให้ค่า RMS = 5.96

2.2)

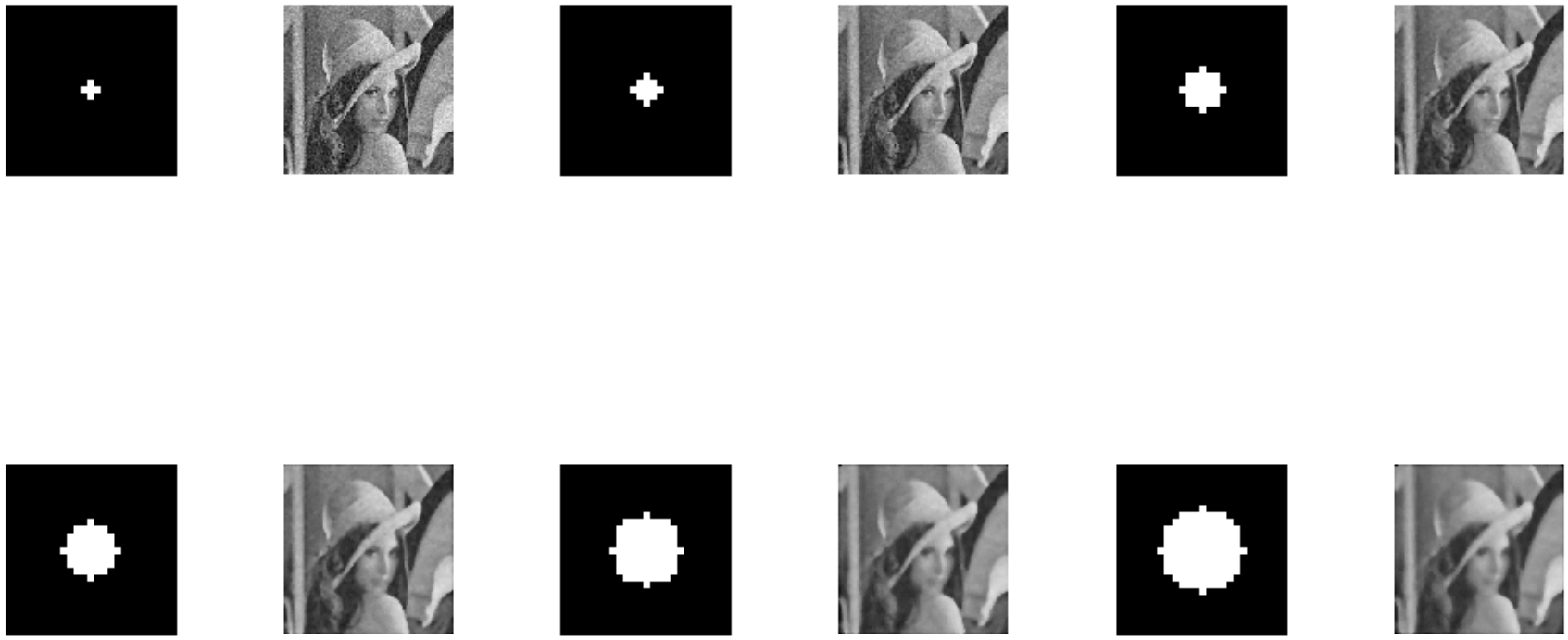


การลด noise ของรูป Chess ผ่าน Median filter โดยใช้ mask เป็นรูปวงกลมที่มีรัศมีต่างๆ

ค่าความผิดพลาด RMS ของรูปที่ผ่าน Median filter เมื่อเทียบกับรูปต้นฉบับที่ไม่มี noise คือ (ลำดับจากซ้ายไปขวาและบนลงล่าง)

1. เซ็ตรัศมีเป็น 1 จะให้ค่า RMS = 7.66
2. เซ็ตรัศมีเป็น 2 จะให้ค่า RMS = 6.39
3. เซ็ตรัศมีเป็น 3 จะให้ค่า RMS = 5.93 (local minimum)
4. เซ็ตรัศมีเป็น 4 จะให้ค่า RMS = 6.02
5. เซ็ตรัศมีเป็น 5 จะให้ค่า RMS = 6.48
6. เซ็ตรัศมีเป็น 6 จะให้ค่า RMS = 6.83

2.2)



การลด noise ของรูป Lenna ผ่าน Median filter โดยใช้ mask เป็นรูปวงกลมที่มีรัศมีต่างๆ

ค่าความผิดพลาด RMS ของรูปที่ผ่าน Median filter เมื่อเทียบกับรูปต้นฉบับที่ไม่มี noise คือ (ลำดับจากซ้ายไปขวาและบนลงล่าง)

1. เซ็ตรัศมีเป็น 1 จะให้ค่า RMS = 7.84
2. เซ็ตรัศมีเป็น 2 จะให้ค่า RMS = 6.83
3. เซ็ตรัศมีเป็น 3 จะให้ค่า RMS = 6.46 (local minimum)
4. เซ็ตรัศมีเป็น 4 จะให้ค่า RMS = 6.58
5. เซ็ตรัศมีเป็น 5 จะให้ค่า RMS = 6.89
6. เซ็ตรัศมีเป็น 6 จะให้ค่า RMS = 7.15

โปรแกรม

1.1)

```
I = imread("Cross.pgm");
F = fft2(I, 256, 256);
F = fftshift(F);
F1 = abs(F);
F1 = log(F1+1);
F1 = mat2gray(F1);
F2 = angle(F);
F2 = mat2gray(F2);
subplot(1, 3, 1);
imshow(I);
subplot(1, 3, 2);
imshow(F1);
subplot(1, 3, 3);
imshow(F2);
```

1.2)

```
I = imread("Cross.pgm");
subplot(1, 5, 1);
imshow(I);
F = fft2(I, 256, 256);
F = fftshift(F);
subplot(1, 5, 2);
imshow(mat2gray(log(1+abs(F))));
subplot(1, 5, 3);
imshow(mat2gray(angle(F)));
Y = zeros(256, 256);
Y(30, 20) = 1;
Z = fft2(Y, 256, 256);
Z = fftshift(Z);
F .*= Z;
subplot(1, 5, 4);
imshow(mat2gray(angle(F)));
J = im2bw(real(ifft2(ifftshift(F))), 0.5);
subplot(1, 5, 5);
imshow(J);
```

1.3)

```
I = imread("Cross.pgm");
I = imrotate(I, 30);
F = fft2(I, 256, 256);
F = fftshift(F);
F1 = abs(F);
F1 = log(F1+1
F1 = mat2gray(F1);
F2 = angle(F);
F2 = mat2gray(F2);
subplot(1, 3, 1);
imshow(I);
subplot(1, 3, 2);
imshow(F1);
subplot(1, 3, 3);
imshow(F2);
```

1.4)

```
I = imread("Cross.pgm");
I = imresize(I, 0.5);
F = fft2(I, 128, 128);
F = fftshift(F);
F1 = abs(F);
F1 = log(F1+1);
F1 = mat2gray(F1);
F2 = angle(F);
F2 = mat2gray(F2);
subplot(1, 3, 1);
imshow(I);
subplot(1, 3, 2);
imshow(F1);
subplot(1, 3, 3);
imshow(F2);
```

1.5)

```
I = imread("Cross.pgm");
F = fft2(I, 256, 256);
F1 = abs(F);
F2 = angle(F);
H = exp(i*F2);
J = ifft2(H);
subplot(1, 2, 1);
imshow(I);
subplot(1, 2, 2);
imshow(J);
```

1.6)

```
I = imread("Cross/Lenna.pgm");
F = fft2(I, 256, 256);
F1 = abs(F);
F2 = angle(F);
H = F1;
J = ifft2(H);
subplot(1, 2, 1);
imshow(I);
subplot(1, 2, 2);
imshow(J);
```

1.7)

```
I = imread("Chess.pgm");
F = fft2(I, 256, 256);
G = fspecial("gaussian", 9, 2);
H = fft2(G, 256, 256);
J1 = imfilter(I, G, "conv");
J2 = uint8(real(ifft2(F .* H)));
subplot(1, 2, 1);
imshow(J1);
subplot(1, 2, 2);
imshow(J2);
```

2.1)

```
I = imread("Cross.pgm");
F = fft2(I, 256, 256);
F = fftshift(F);
Fshow = log(1+abs(F));
G = fspecial("gaussian", 256, 12);
G /= max(max(G));
F1 = F .* im2bw(G, 0.2);
F2 = F .* G;
F1show = log(1+abs(F1));
F2show = log(1+abs(F2));
J1 = ifft2(ifftshift(F1));
J2 = ifft2(ifftshift(F2));
subplot(2, 3, 1);
imshow(I);
subplot(2, 3, 4);
imshow(I);
subplot(2, 3, 2);
imshow(F1);
subplot(2, 3, 5);
imshow(F2);
subplot(2, 3, 3);
imshow(J1);
subplot(2, 3, 6);
imshow(J2);
```

2.2.1) Median filter

```
I = imread("Chess/Lenna_noise.pgm");
K = imread("Chess/Lenna.pgm");
total_i = 6;
start_i = 1;
total_col = 6;
total_row = 1 + (2 * total_i) / total_col;
plot_count = 1;
for i = start_i:1:(start_i+total_i-1)
  subplot(total_row, total_col, plot_count++);
  imshow(I);
end
[U V] = meshgrid(1:25);
for i = start_i:1:(start_i+total_i-1)
  N = sqrt((U - 13).^2 + (V - 13).^2) <= i;
  J = medfilt2(I, N);
  subplot(total_row, total_col, plot_count++);
  imshow(N);
  subplot(total_row, total_col, plot_count++);
  imshow(J);
  fprintf("RMS error for J%d is %6.2f\n", i, sqrt(sum(sum(imsubtract(K,
J).^2))/prod(size(I))));
end
```

2.2.2) Butterworth low-pass filter

```
wb = waitbar(0, 'Loading images...');
I = imread("Chess/Lenna_noise.pgm");
K = imread("Chess/Lenna.pgm");
np2 = 2.^nextpow2(max(size(I)));
F = fft2(I, np2, np2);
F = fftshift(F);
total_n = 3;
total_i = 3;
start_n = 1;
start_i = 18;
plot_count = 1;
for i = 1:1:(2*total_i)
  subplot(1+total_n, 2*total_i, plot_count);
  imshow(I);
  plot_count++;
end
[U V] = meshgrid(1:np2);
ONES = ones(np2, np2);
for n = start_n:1:(start_n+total_n-1)
  for i = start_i:1:(start_i+total_i-1)
    progress = (plot_count) ./ ((2*total_i)*(1+total_n));
    waitbar(progress, wb, strcat('Processing... (', int2str(int8(100 * progress)),
'%)'));
    H = ONES ./ (1 + (sqrt((U - np2/2).^2 + (V - np2/2).^2) ./ i).^(2*n));
    subplot(1+total_n, 2*total_i, plot_count);
    imshow(H);
    plot_count++;
    J = uint8(real(ifft2(ifftshift(F .* H))));
    fprintf("RMS error for J_%d_%d is %6.2f\n", n, i, sqrt(sum(sum(imsubtract(K,
J).^2))/prod(size(I))));
    subplot(1+total_n, 2*total_i, plot_count);
    imshow(J);
    plot_count++;
  end
end
close(wb);
```

2.2.3) Gaussian low-pass filter

```
wb = waitbar(0, 'Loading images...');
I = imread("Chess/Lenna_noise.pgm");
K = imread("Chess/Lenna.pgm");
np2 = 2.^nextpow2(max(size(I)));
F = fft2(I, np2, np2);
F = fftshift(F);
total_i = 9;
start_i = 30;
plot_count = 1;
total_col = 6;
total_row = 1 + (2*total_i) / total_col;
for i = 1:1:total_col
  subplot(total_row, total_col, plot_count);
  imshow(I);
  plot_count++;
end
[U V] = meshgrid(1:np2);
ONES = ones(np2, np2);
for i = start_i:1:(start_i+total_i-1)
  progress = (plot_count) ./ ((total_col)*(total_row));
  waitbar(progress, wb, strcat('Processing... (', int2str(int8(100 * progress)),
'%)'));
  H = exp(-((U - np2/2).^2 + (V - np2/2).^2) ./ (2 * i^2));
  subplot(total_row, total_col, plot_count);
  imshow(H);
  plot_count++;
  J = uint8(real(ifft2(ifftshift(F .* H))));
  fprintf("RMS error for J_%d is %6.2f\n", i, sqrt(sum(sum(imsubtract(K,
J).^2))/prod(size(I))));
  subplot(total_row, total_col, plot_count);
  imshow(J);
  plot_count++;
end
close(wb);
```