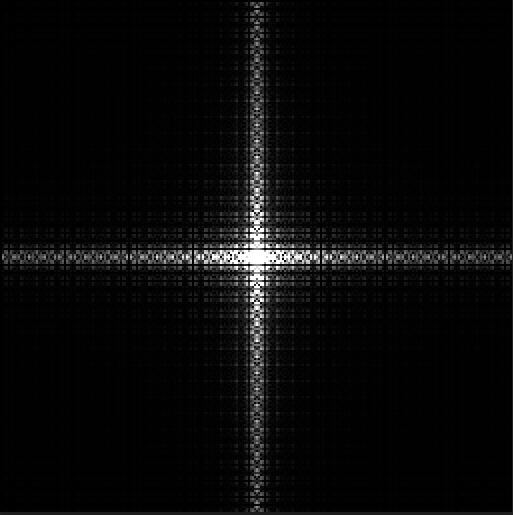
**Digital Image Processing (261453)**

**Computer Assignment 2**

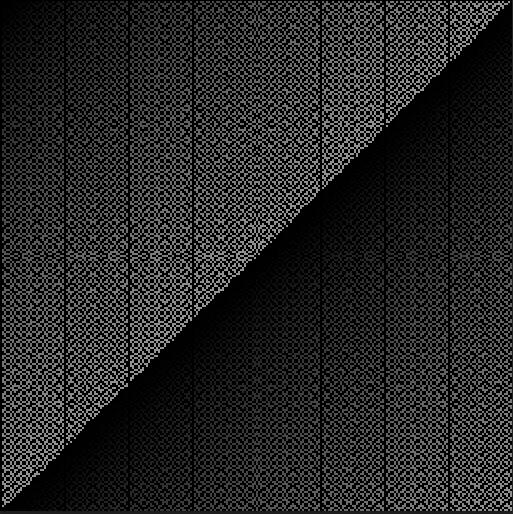
**1.  Properties of the Fourier Transform**

            1.1) ทำการแปลง Fourier Transform ของภาพ “[Cross.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Cross.pgm)” (200×200) เนื่องจากในการใช้ FFT จำเป็นต้องให้ภาพมีขนาดที่อยู่ในรูปของ 2*n* ดังนั้น อาจจะต้องทำการ Pad ก่อน และให้แสดงภาพผลลัพธ์ในรูปของ amplitude และ phase spectra

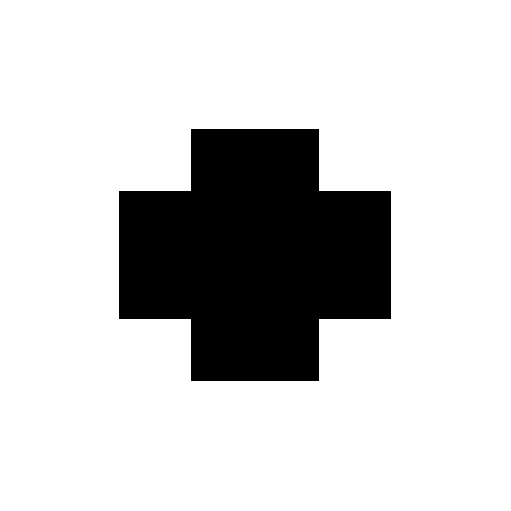
Amplitude

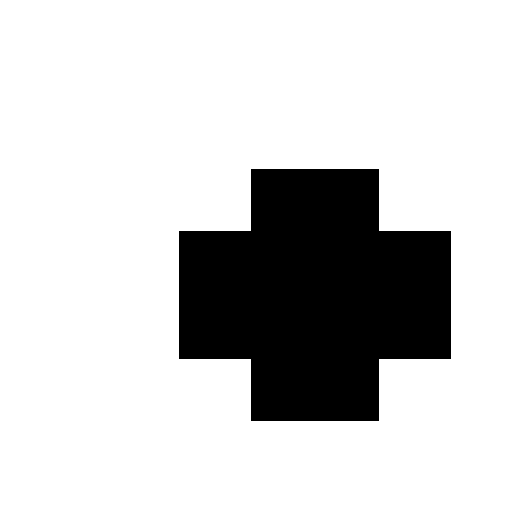


Phase spectra



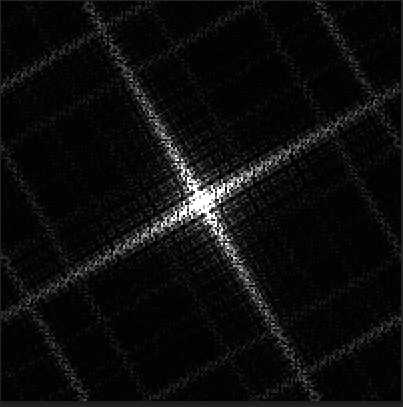
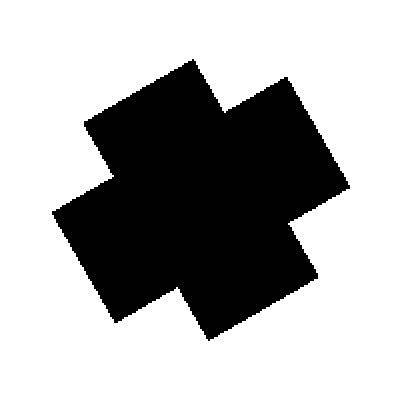
1.2) ให้ทำการคูณ phase spectrum ที่ได้ในข้อ 1.1 ด้วย  complex number ค่าหนึ่ง เพื่อที่ว่าเมื่อทำการ inverse Fourier transform แล้ว ภาพผลลัพธ์ที่ได้ ย้ายด้วยจำนวนในแกน *x* และ *y* เป็น (20,30)



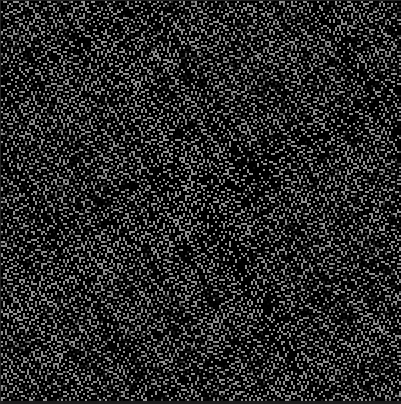


1.3) ทำการหมุนภาพ “[Cross.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Cross.pgm)” ไป 30 องศา และแสดงผลของการแปลงฟูเรียร์ ให้ทำการวิเคราะห์ว่าเกิดอะไรขึ้น

ภาพที่ได้จากการหมุน Amplitude



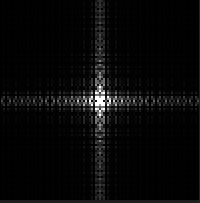
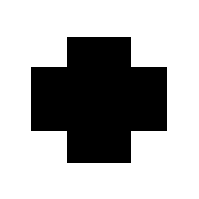
Phase spectra



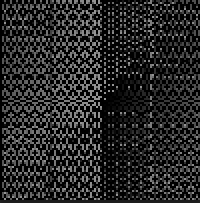
จากการหมุน จะสังเกตเห็นได้ว่าจะทำให้ Amplitude และ Phase ของรูปเปลี่ยนไป โดยที่ Amplitude นั้นหมุนไปตามแนวที่รูปเกิดการหมุนเป็นฯไปตามคุณสมบัติของ การหมุน

1.4) ทำการ Down-sample “[Cross.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Cross.pgm)” เพื่อให้รูปมีขนาด 100×100 หลังจากนั้นทำการแปลง Fourier Transform ให้แสดงภาพผลลัพธ์ในรูปของ amplitude และ phase spectra ให้ทำการวิเคราะห์ว่าเกิดอะไรขึ้น

ภาพที่ได้จากการหมุน Amplitude



Phase spectra



จะเห็นได้ว่าทั้ง Amplitude และ Spectrum ไม่มีการเปลี่ยนแปลงเลย เพราะ การ scaling ภาพ ไม่มีผลต่อการเปลี่ยนแปลง

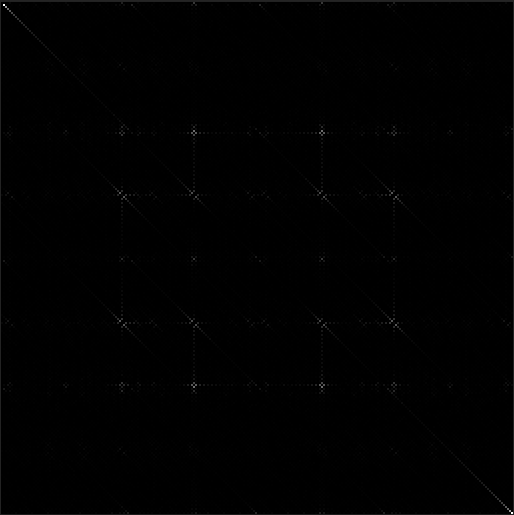
1.5) ใช้การแปลง  inverse Fourier Transform ของผลลัพธ์ที่ได้ในข้อ 1.1 โดยที่

                        1.5.1) ไม่ใช้ข้อมูล phase

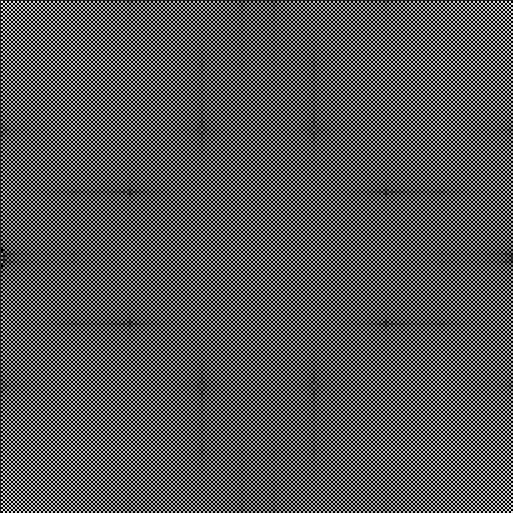
                        1.5.2) ไม่ใช้ข้อมูล amplitude

            ให้ทำการวิเคราะห์ว่าเกิดอะไรขึ้น

ใช้แต่ Phase



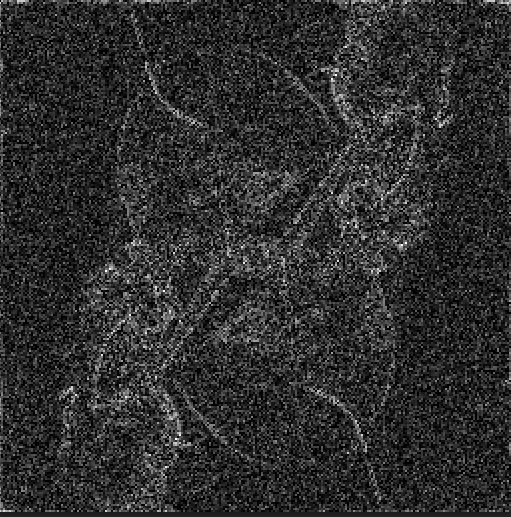
ใช้แต่ Amplitude



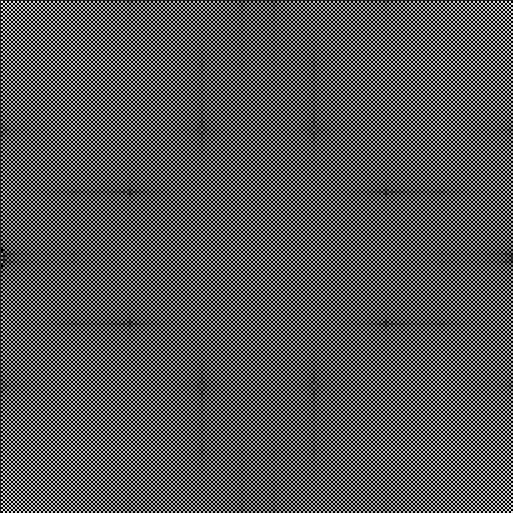
การที่ใช้แต่ข้อมูล Amplitude จะเห็นแต่ข้อมูลของความเข้มแสง โดยมีแค่ DC Component เท่านั้นที่เด่นชัด แต่ถ้าใช้ข้อมูล Phase จะทำให้สามารถเห็นขอบของวัตถุในภาพ ทำให้เราจึงทราบว่า ข้อมูล Phaseมีข้อมูลของขอบวัตถุประกอบอยู่ด้วย

1.6) ให้ทำการทดลองเช่นเดียวกับข้อง 1.5 ด้วยภาพ “[Lenna.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Lenna.pgm)” (256×256)

ภาพ Lenna.pgm ใช้แต่ Phase

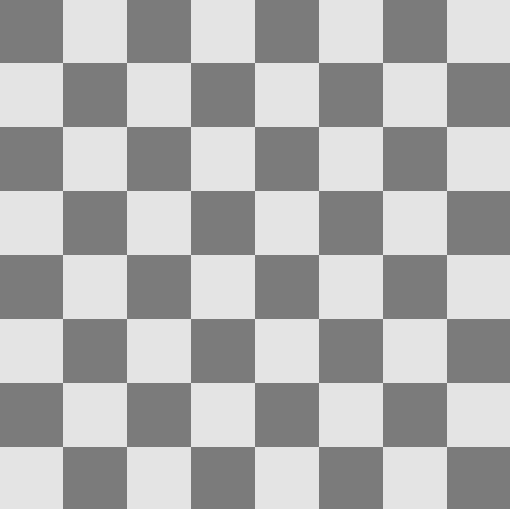
ใช้แต่ Amplitude



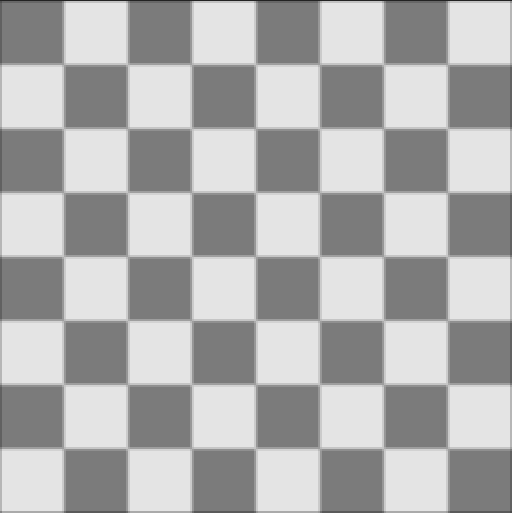
การที่ใช้แต่ข้อมูล Amplitude จะเห็นแต่ข้อมูลของความเข้มแสง โดยมีแค่ DC Component เท่านั้นที่เด่นชัด แต่ถ้าใช้ข้อมูล Phase จะทำให้สามารถเห็นขอบของวัตถุในภาพ ทำให้เราจึงทราบว่า ข้อมูล Phaseมีข้อมูลของขอบวัตถุประกอบอยู่ด้วย

1.7) ทำ convolution ภาพ “[Chess.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Chess.pgm)” (256×256) ด้วย mask หรือ kernel ขนาดเล็กอันหนึ่ง เพื่อทำการ Blur ภาพ และให้ทำการ filter ใน frequency  domain ด้วย Fourier transform ของ Kernel นั้นด้วย เพื่อทำการ Blur ภาพด้วย เปรียบเทียบผลที่ได้ทั้งสองแบบ ว่ามีความแตกต่างหรือหรือเหมือนกันอย่างไรภาพ

Chess.pgm



Blur in FFT Blur with Convolution

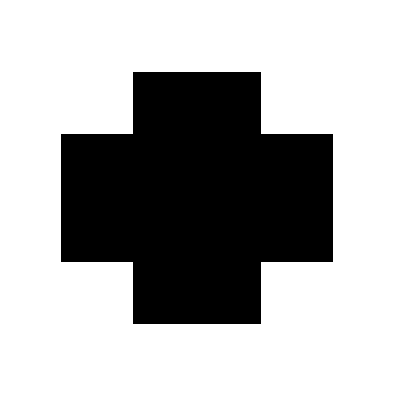


จากผลการทดลองเห็นว่า รูปที่ได้จากการเบลอด้วยทั้งสองวิธีนั้น ได้ผลลัพธ์ที่เหมือนกัน

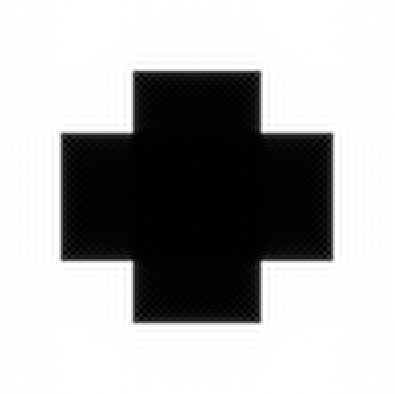
**2. Filter Design**

            2.1) ให้ใช้ ideal low-pass filter กับภาพ “[Cross.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Cross.pgm)” โดยที่เปลี่ยน cutoff frequency และให้ศึกษา ringing effect ที่เกิดขึ้น หลังจากนั้นให้ทำการทดลองซ้ำด้วย Non-ideal filter อื่น

Ideal low-pass filter



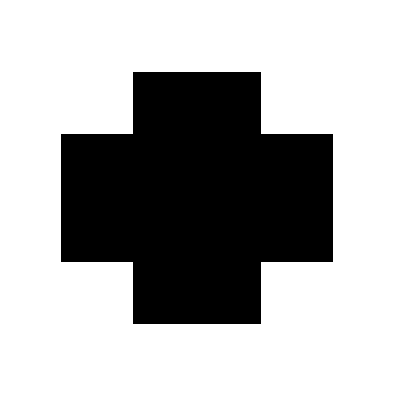
Original Ideal low pass filter with Cut off 20



Ideal low pass filter with Cut off 30 Ideal low pass filter with Cut off 50

จากการทดลองพบว่า เมื่อรูปต้นฉบับได้ถูกนำไปผ่าน Ideal low pass filter ที่ความถี่ Cut off ต่ำๆ ปรากฏว่าภาพที่ได้จะเบลอขึ้น ทำให้ขอบของวัตถุไม่ชัดเจน และยังทำให้เกิด Ringing Effect อีกด้วย แต่เมื่อนำไปผ่านความถี่ Cut off โดยเพิ่มค่าขึ้นเรื่อยปรากฏว่าภาพที่ได้นั้น ค่อยๆ ชัดขึ้นตามการเพิ่มค่าเลขความถี่ Cut off

Butterworth low-pass filter ที่ Order(n) = 2



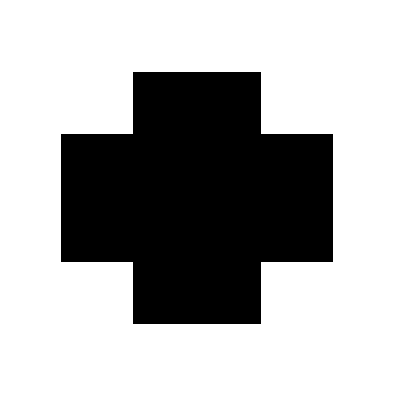
Original Cut off 20

Cut off 30 Cut off 50

จะพบว่าเมื่อใช้ Butter worth low-pass filter ภาพที่ได้นั้นคือ ภาพที่มีความเบลอ และไม่มีการเกิด Ringing Effect ถ้า N มีค่าเหมาะสม เมื่อมีการเพิ่ม Cut off ขึ้นเรื่อยๆ รูปที่ได้ก็จะมีความคล้ายคลึงกับต้นฉบับมากขึ้นเมื่อเพิ่มค่านี้เช่นกัน

Butterworth low-pass filter ที่เปลี่ยนค่า N แต่ cut off คงที่



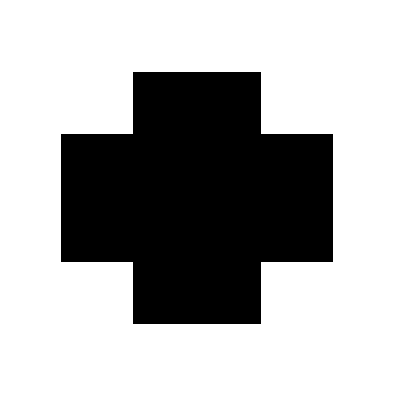
Original N = 2



N = 20 N = 50

จะเห็นว่า Butter worth low-pass filter ก็ทำให้เกิด Ringing Effect ได้ถ้าเราเพิ่มค่า N ขึ้นไปเรื่อย ดังนั้นเราควรจะเลือกค่า N ที่เหมาะสม ส่วนใหญ่นิยมใช้ N = 2

Gaussian Low-pass filter

 Original Cut off 20



Cut off 30 Cut off 50

จากการนำรูปภาพไปผ่าน Gaussian low-pass Filter แล้วพบว่าภาพที่ได้มีความเบลอเกิดขึ้นบนภาพ แต่ถ้าเพิ่มค่าความถี่ Cut off ขึ้นอีก เราจะพบว่ารูปภาพมีความใกล้เคียงกับรูปต้นฉบับไปเรื่อยๆตามค่าที่เพิ่มขึ้น และก็ยังพบอีกว่า ไม่มีการเกิด Ringing Effect ขึ้นมาเลย

2.2) ให้ใช้ low-pass filter หลายแบบ โดยที่เปลี่ยนค่าตัวแปรต่างๆ รวมถึงขนาดของ filter ด้วย กับภาพ “[Chess\_noise.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Chess_noise.pgm)” และ “[Lenna\_noise.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Lenna_noise.pgm)” เพื่อลด noise และให้เปรียบเทียบผลกับ Median filter ที่มีการเปลี่ยนขนาด และให้ใช้ RMS ในการคำนวณหาความแตกต่างระหว่างผลลัพธ์ที่ได้กับภาพที่ไม่มี Noise “[Chess.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Chess.pgm)” and “[Lenna.pgm](http://sansanee.cpe.eng.cmu.ac.th/DIPUnG/dataset/Lenna.pgm)”

ในการคำนวณ RMS ใช้สมการการคำนวณดังนี้

โดยที่

n และ m คือขนาดของด้านในแนวแกน x และแกน y ตามลำดับ

f1 และ f2 คือ ค่า Grey Scale ในตำแหน่งนั้นๆของรูปที่ 1 กับ รูปที่ 2 โดยที่ i และ j เป็นตำแหน่งของ pixel

ภาพ Lenna.pgm



ภาพ Lenna\_noise.pgm



ภาพ lenna\_noise ที่ผ่าน Ideal low-pass filter

Cut off 20 Cut off 30



Cut off 50

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Lenna.pgm | Lenna.pgm | 0.0 |
| Lenna.pgm | Lenna\_noise.pgm | 25.2807822355 |
| Lenna.pgm | Ideal low pass with cut off 20 | 17.6081566376 |
| Lenna.pgm | Ideal low pass with cut off 30 | 14.1999086404 |
| Lenna.pgm | Ideal low pass with cut off 50 | 12.6860422885 |

ภาพ lenna\_noise ที่ผ่าน Butter worth low-pass filter ที่ n = 2

Cut off 20 Cut off 30



Cut off 50

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Lenna.pgm | Lenna.pgm | 0.0 |
| Lenna.pgm | Lenna\_noise.pgm | 25.2807822355 |
| Lenna.pgm | butterworth with cut off 20 | 15.5516716847 |
| Lenna.pgm | butterworth with cut off 30 | 12.6251371773 |
| Lenna.pgm | butterworth with cut off 50 | 11.084632587 |

ภาพ lenna\_noise ที่ผ่าน Gaussian low-pass filter

Cut off 20 Cut off 30



Cut off 50

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Lenna.pgm | Lenna.pgm | 0.0 |
| Lenna.pgm | Lenna\_noise.pgm | 25.2807822355 |
| Lenna.pgm | Gaussian with cut off 20 | 14.4654091589 |
| Lenna.pgm | Gaussian with cut off 30 | 11.933575849 |
| Lenna.pgm | Gaussian with cut off 50 | 11.2171429091 |

ภาพ lenna\_noise ที่ผ่าน Median filter

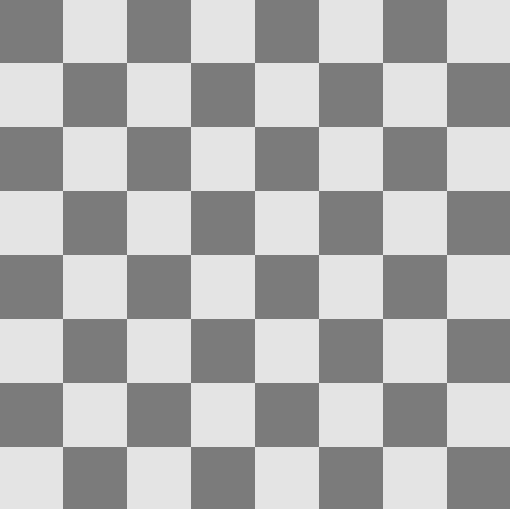


Median Filter

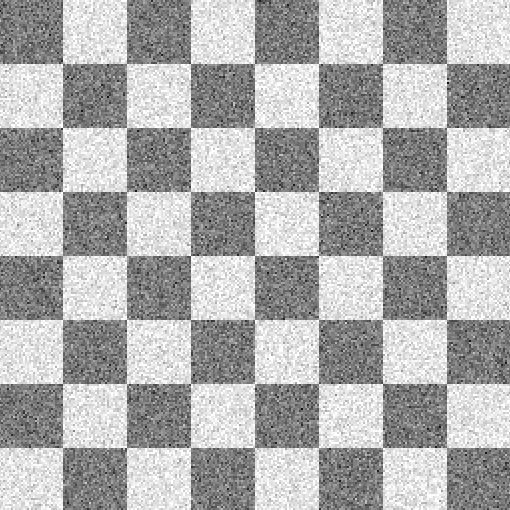
|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Lenna.pgm | Lenna.pgm | 0.0 |
| Lenna.pgm | Lenna\_noise.pgm | 25.2807822355 |
| Lenna.pgm | Median Filter | 13.5181113365 |

จากการทดลองเราได้ภาพที่ผ่าน Filter ต่างๆ และค่า RMS ซึ่งค่า RMS อาจจะสามารถบอกได้ว่ารูปภาพที่มีค่าน้อยกว่าค่า RMS ของ Lenna\_noise แสดงถึงตัว Filter ที่ใช้มีความสามารถในการลด noise ได้ และยิ่งถ้าความต่างของรูป (RMS) ที่ผ่าน Filter นั้นๆมีค่าเข้าใกล้ 0 ก็แสดงว่ามีความสามารถในการลด Noise ได้มาก ในการทดลองนี้พบว่า Gaussian with cutoff 50 ดีที่สุด(การทดลองนี้ไม่ได้ครอบคลุมทุก Cutoff อาจจะมีค่าที่ดีกว่านี้)

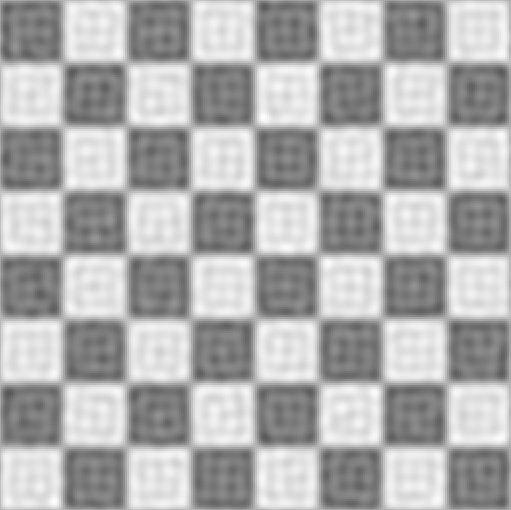
ภาพ Chess.pgm



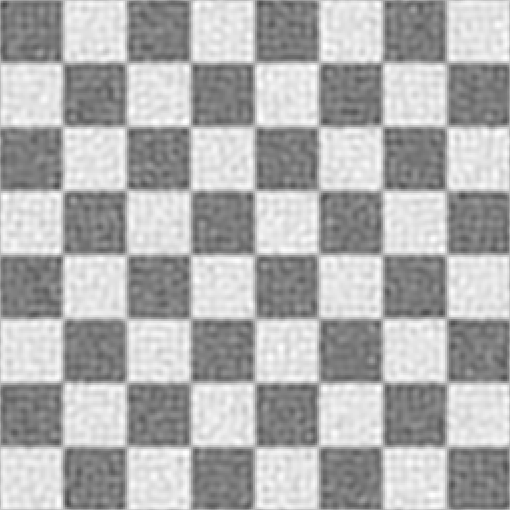
ภาพ Chess\_noise.pgm



ภาพ Chess\_noise ที่ผ่าน Ideal low-pass filter

Cut off 20 Cut off 30



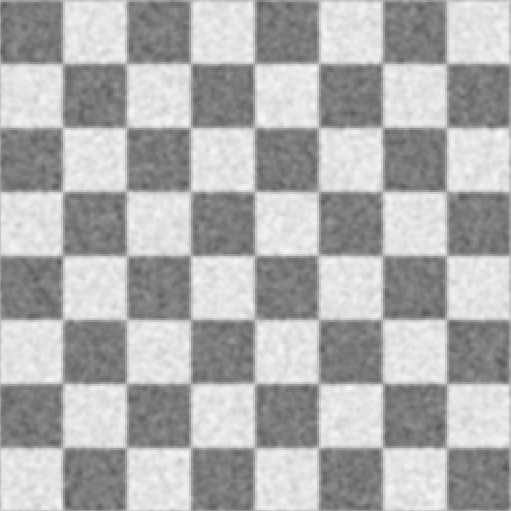
Cut off 50

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Chess.pgm | Chess.pgm | 0.0 |
| Chess.pgm | Chess\_noise.pgm | 24.5852060758 |
| Chess.pgm | Ideal low pass with cut off 20 | 26.131443233 |
| Chess.pgm | Ideal low pass with cut off 30 | 19.446980314 |
| Chess.pgm | Ideal low pass with cut off 50 | 16.8509834342 |

ภาพ Chess\_noise ที่ผ่าน Butter worth low-pass filter

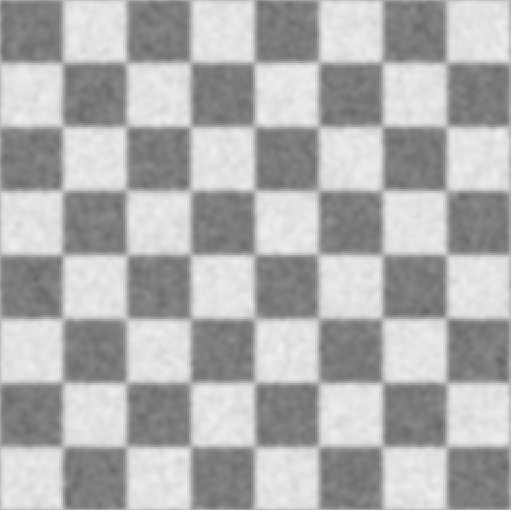
Cut off 20 Cut off 30



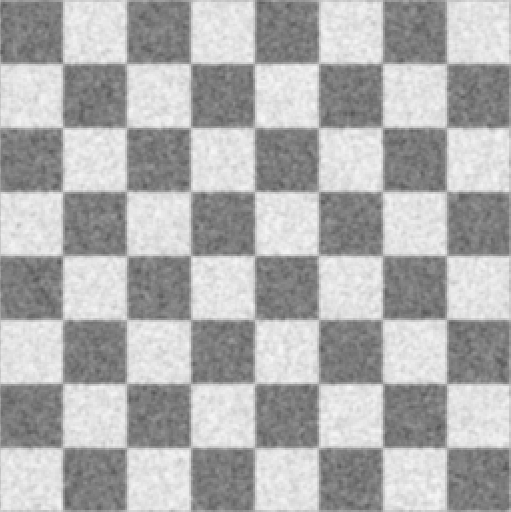
Cut off 50

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Chess.pgm | Chess.pgm | 0.0 |
| Chess.pgm | Chess\_noise.pgm | 24.5852060758 |
| Chess.pgm | Butterworth with cut off 20 | 22.0184176253 |
| Chess.pgm | Butterworth with cut off 30 | 18.0382847013 |
| Chess.pgm | Butterworth with cut off 50 | 14.6207029111 |

ภาพ Chess\_noise ที่ผ่าน Gaussian low-pass filter

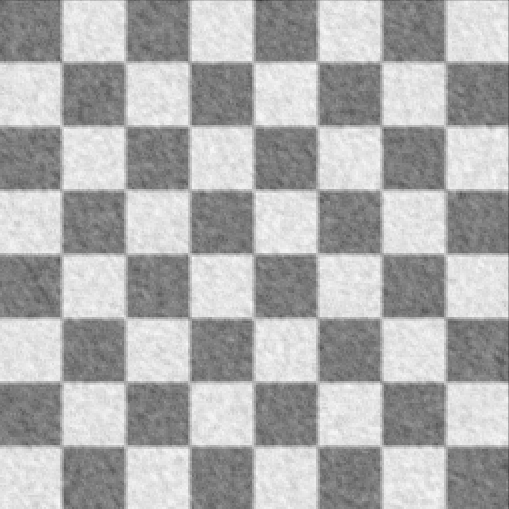
Cut off 20 Cut off 30



Cut off 50

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Chess.pgm | Chess.pgm | 0.0 |
| Chess.pgm | Chess\_noise.pgm | 24.5852060758 |
| Chess.pgm | Gaussian with cut off 20 | 20.7212630599 |
| Chess.pgm | Gaussian with cut off 30 | 17.0200391356 |
| Chess.pgm | Gaussian with cut off 50 | 14.0904350358 |

ภาพ Chess\_noise ที่ผ่าน Median filter



Median Filter

|  |  |  |
| --- | --- | --- |
| ภาพที่ 1 | เทียบกับภาพที่ 2 | RMS |
| Chess.pgm | Chess.pgm | 0.0 |
| Chess.pgm | Chess\_noise.pgm | 25.2807822355 |
| Chess.pgm | Median Filter | 17.9855237606 |

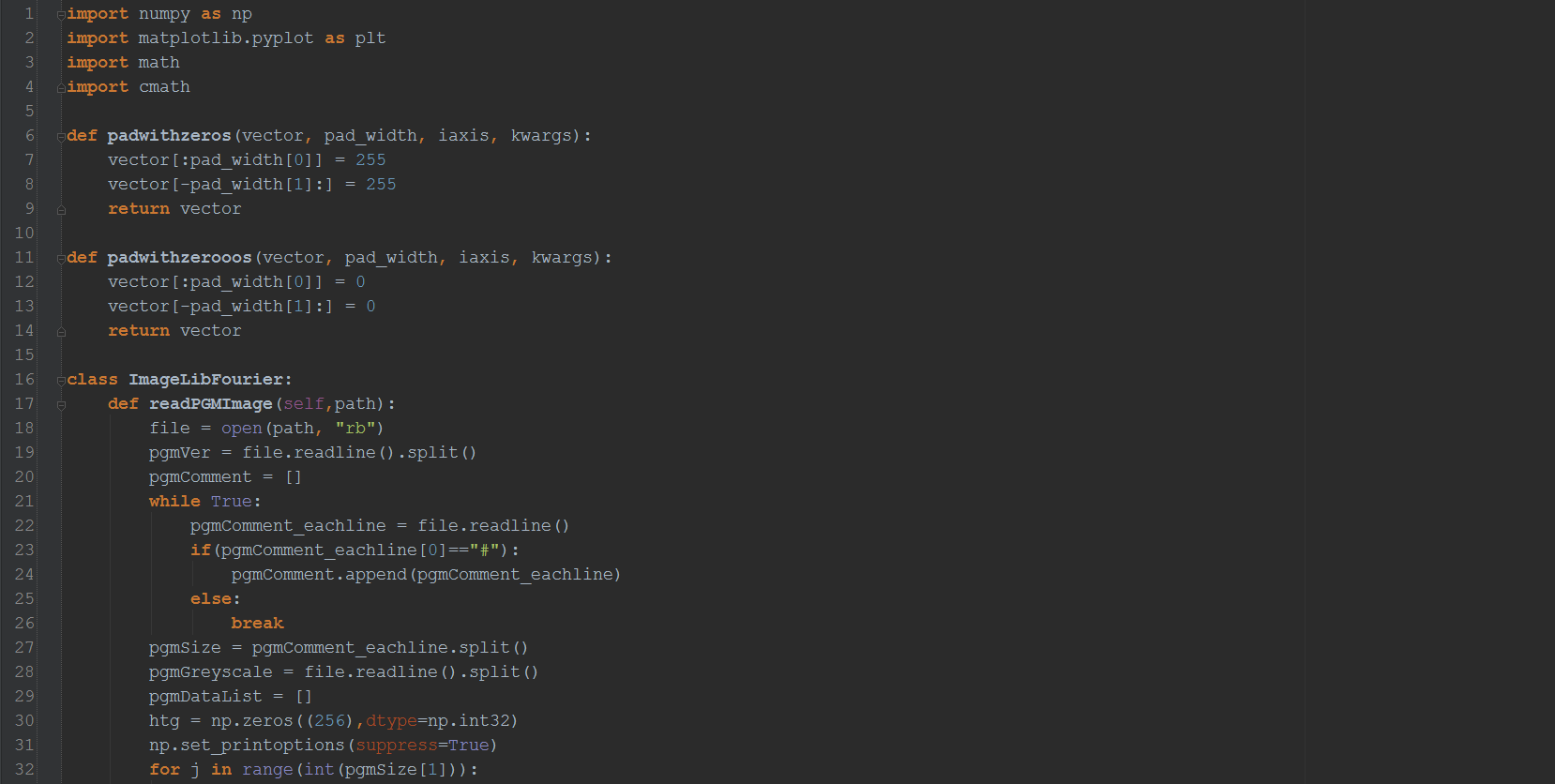
จากการทดลองเราได้ภาพที่ผ่าน Filter ต่างๆ และค่า RMS ซึ่งค่า RMS อาจจะสามารถบอกได้ว่ารูปภาพที่มีค่าน้อยกว่าค่า RMS ของ Chess\_noise แสดงถึงตัว Filter ที่ใช้มีความสามารถในการลด noise ได้ และยิ่งถ้าความต่างของรูป (RMS) ที่ผ่าน Filter นั้นๆมีค่าเข้าใกล้ 0 ก็แสดงว่ามีความสามารถในการลด Noise ได้มาก ในการทดลองนี้พบว่า Gaussian with cutoff 50 ดีที่สุด(การทดลองนี้ไม่ได้ครอบคลุมทุก Cutoff อาจจะมีค่าที่ดีกว่านี้)

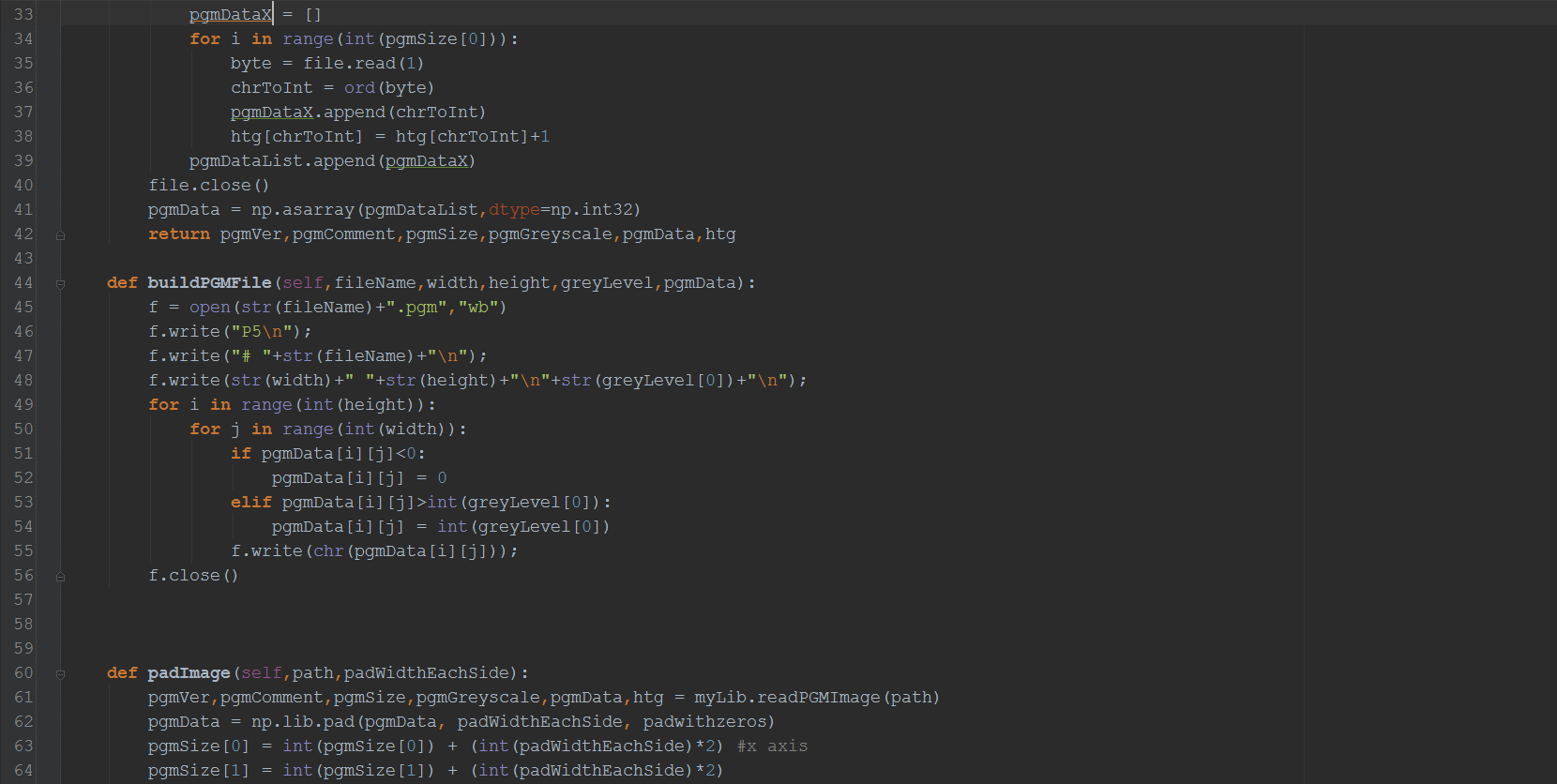
และเรายังพบอีกว่าในส่วนของ Ideal low pass ที่ cut off มีค่า 20 นั้นจะทำให้เกิด Ringing Effect อย่างเห็นได้ชัดและเมื่อสังเกตค่า RMS ของมัน ก็พบว่ามีค่ามากกว่า RMS ของ Chess\_noise.pgm เสียอีก

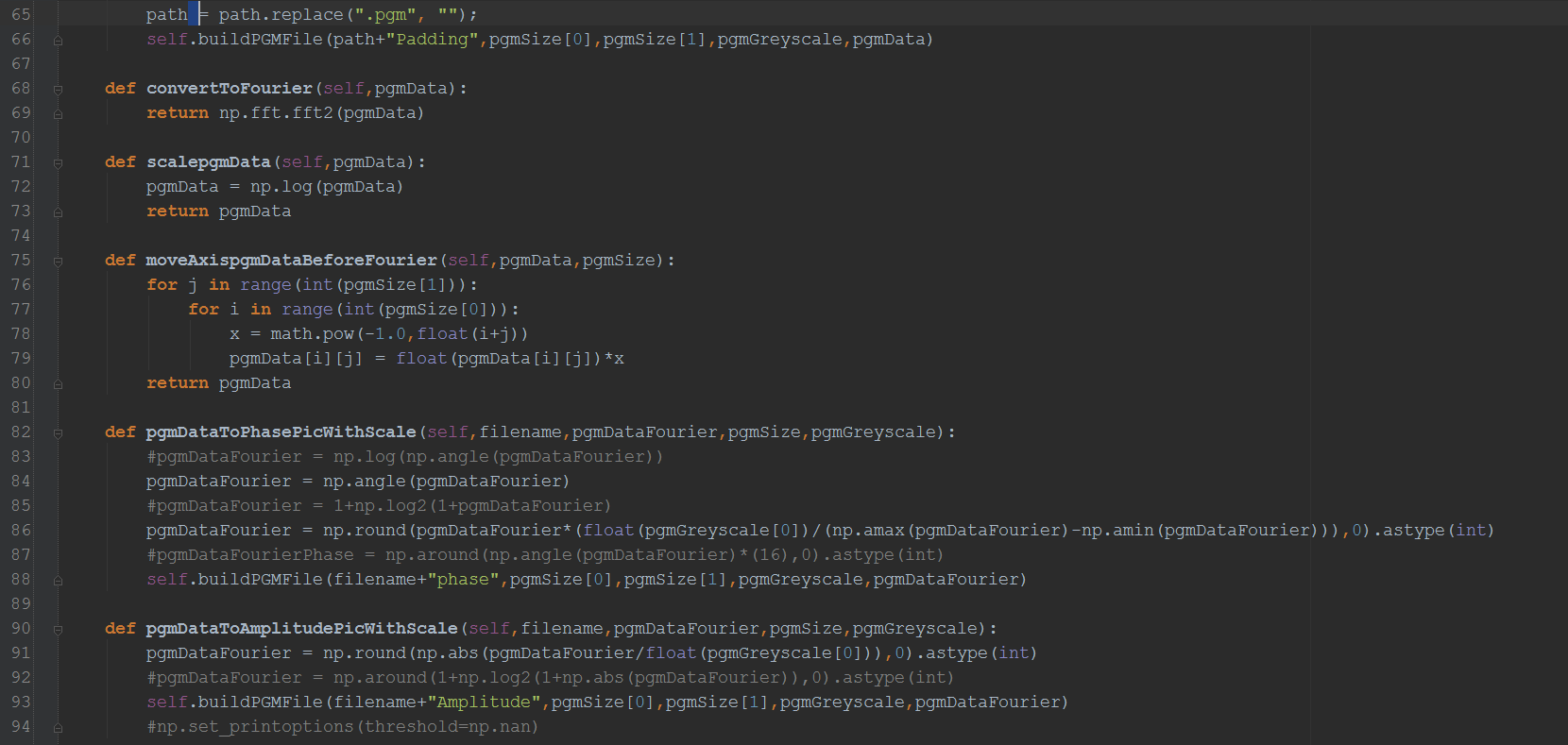
Link โค้ด 🡪 https://github.com/SupakornYu/ImageProcessingHomework-HW2/blob/master/img\_main2.py

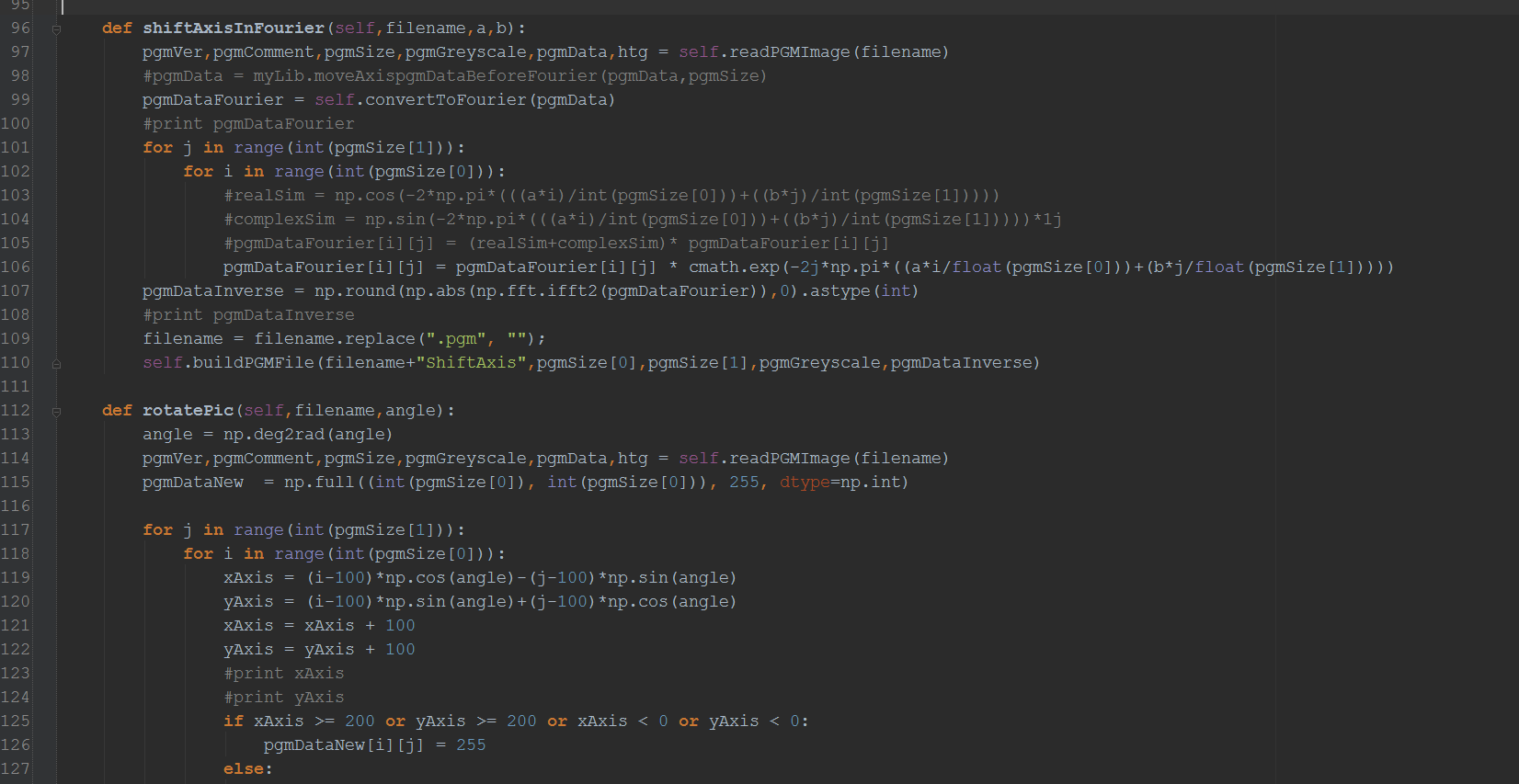
ใช้ภาษา Python ในการพัฒนา

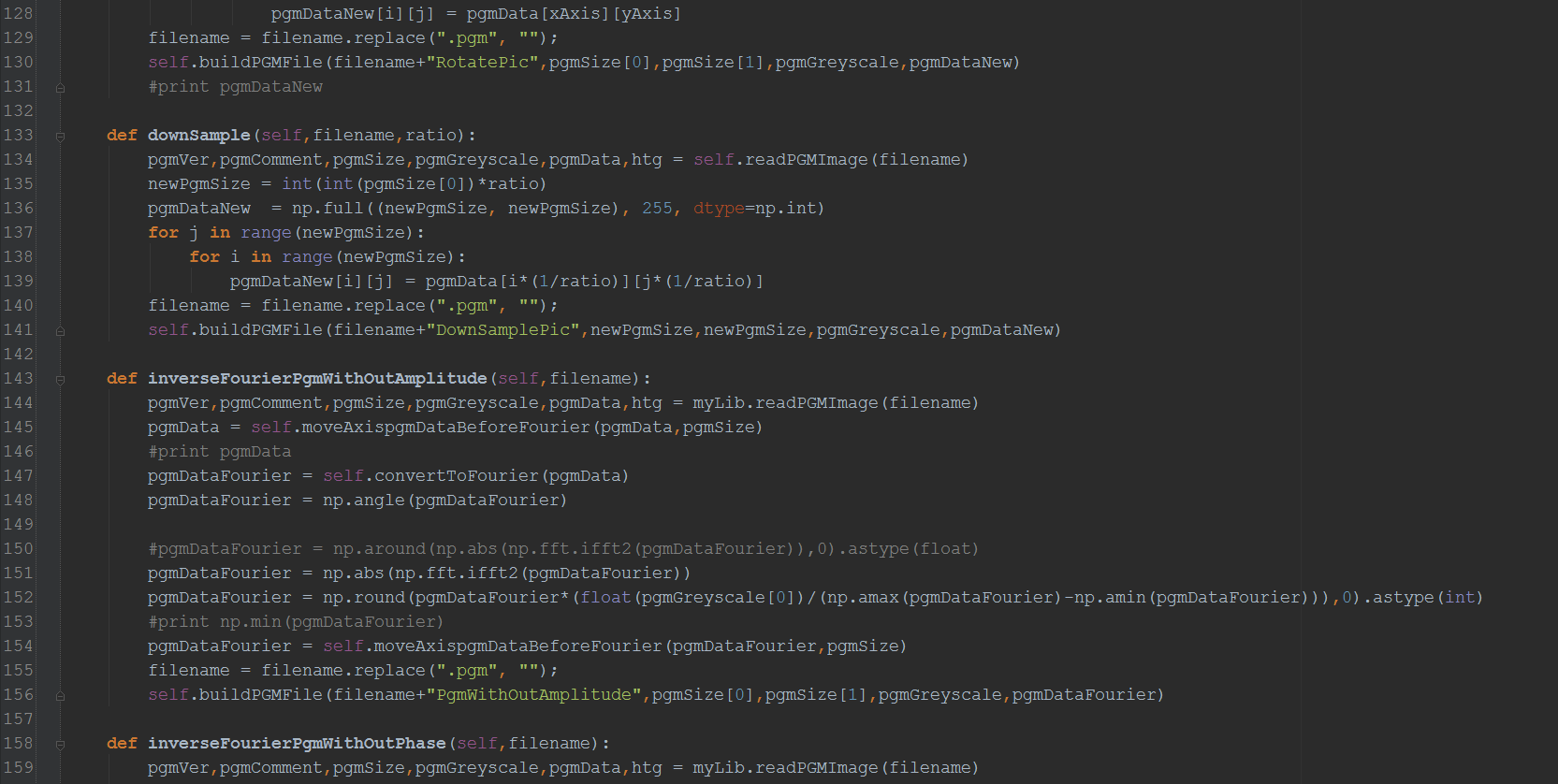
Library ที่เขียนไว้มีดังนี้

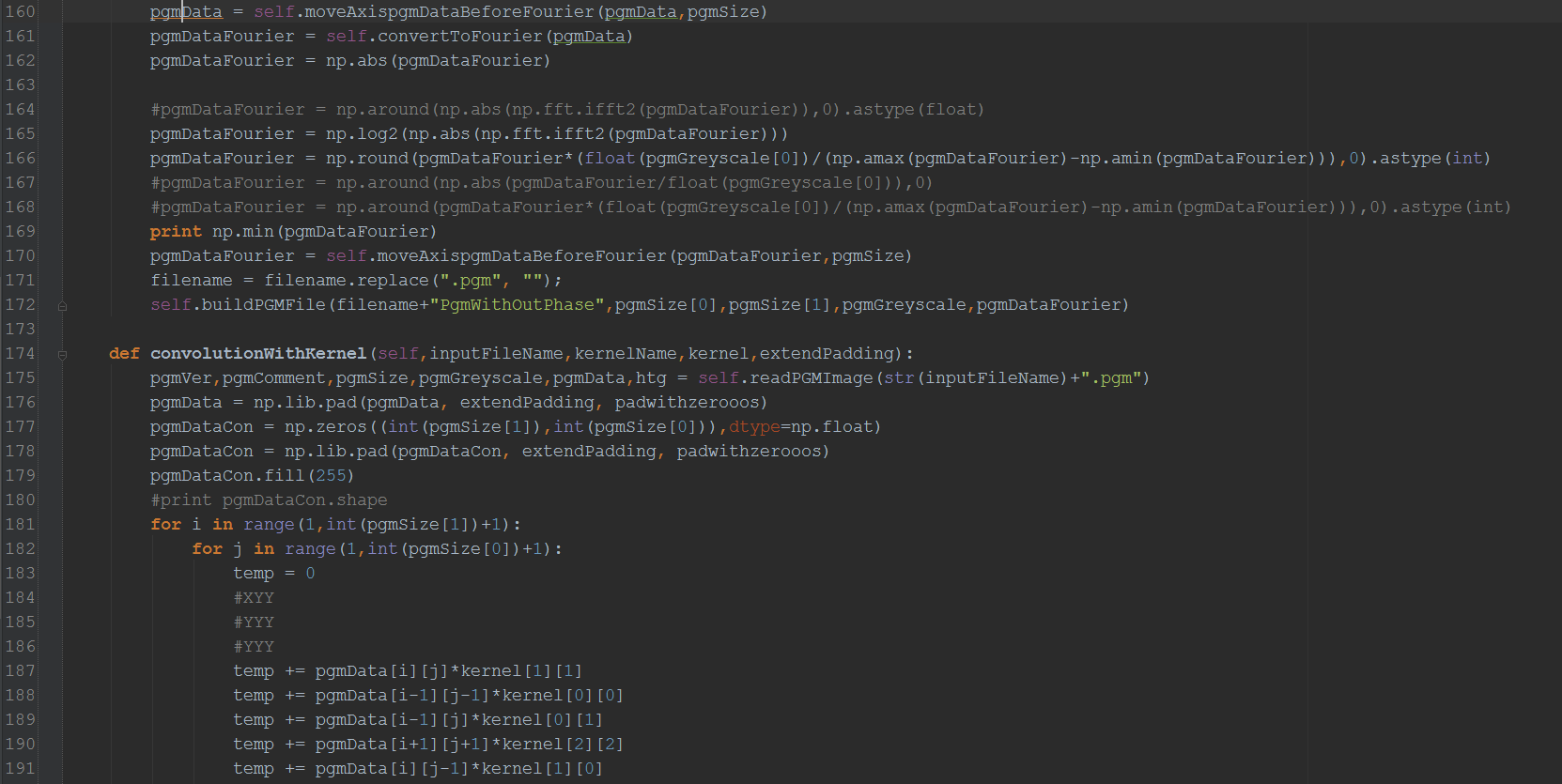


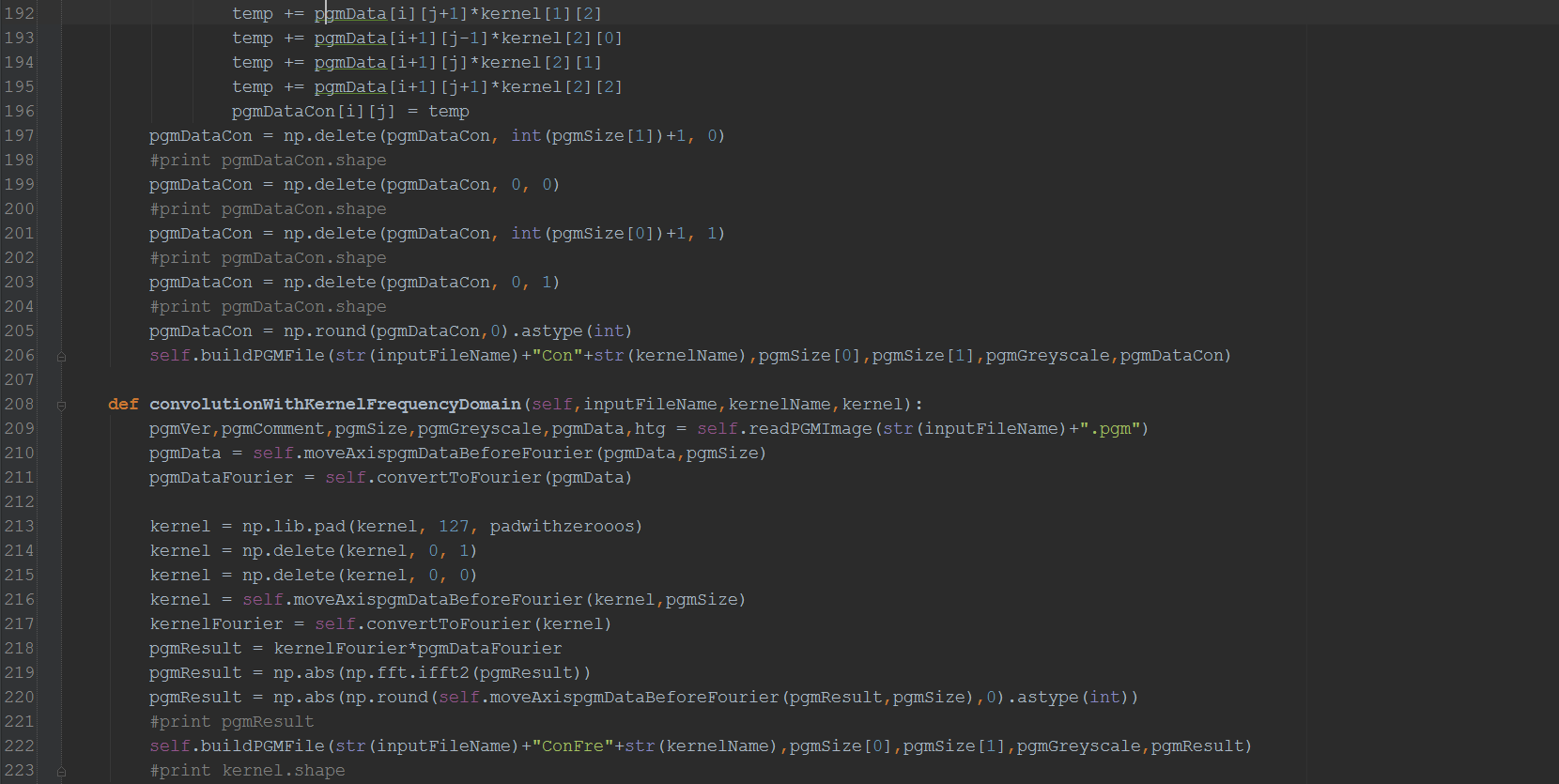


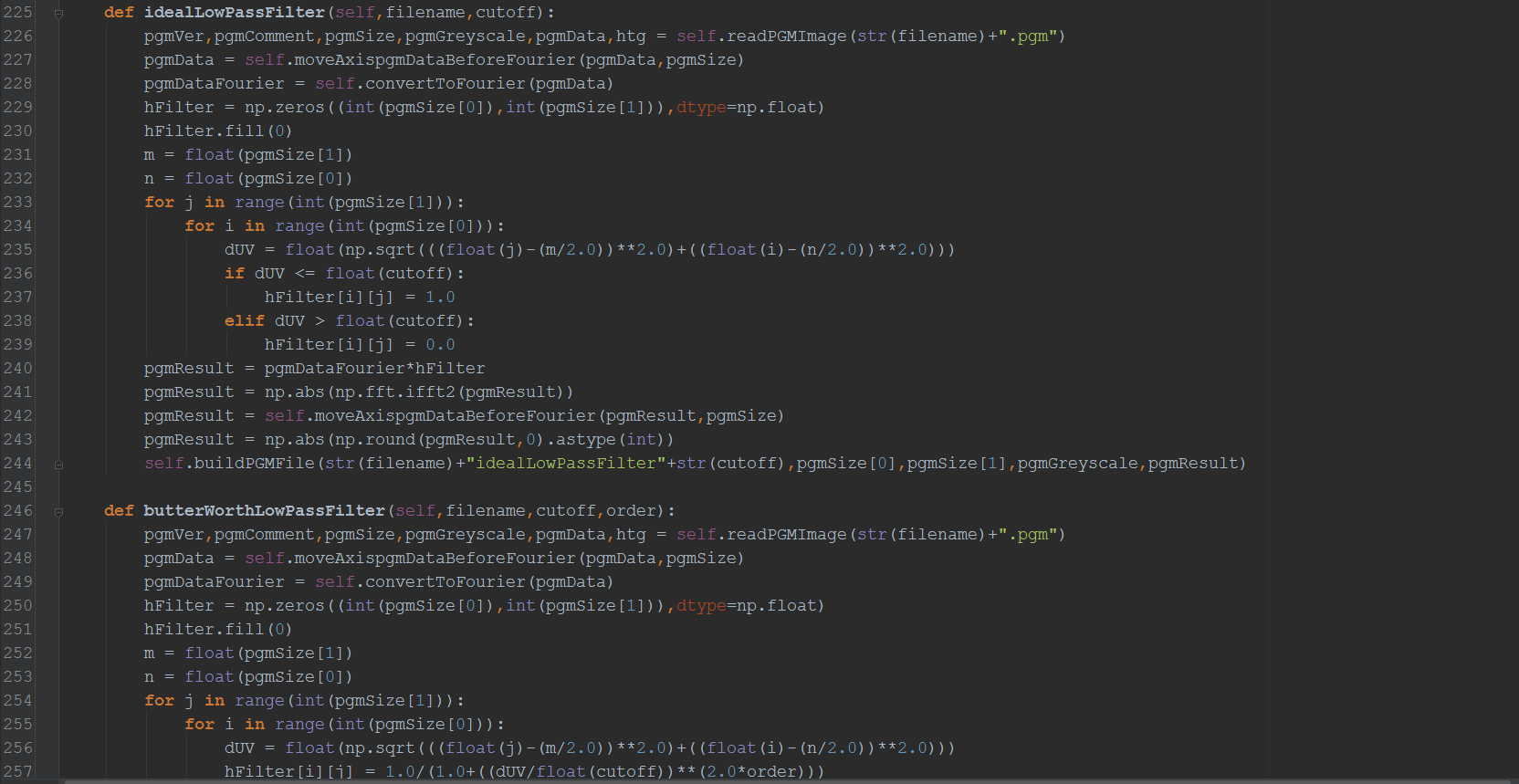


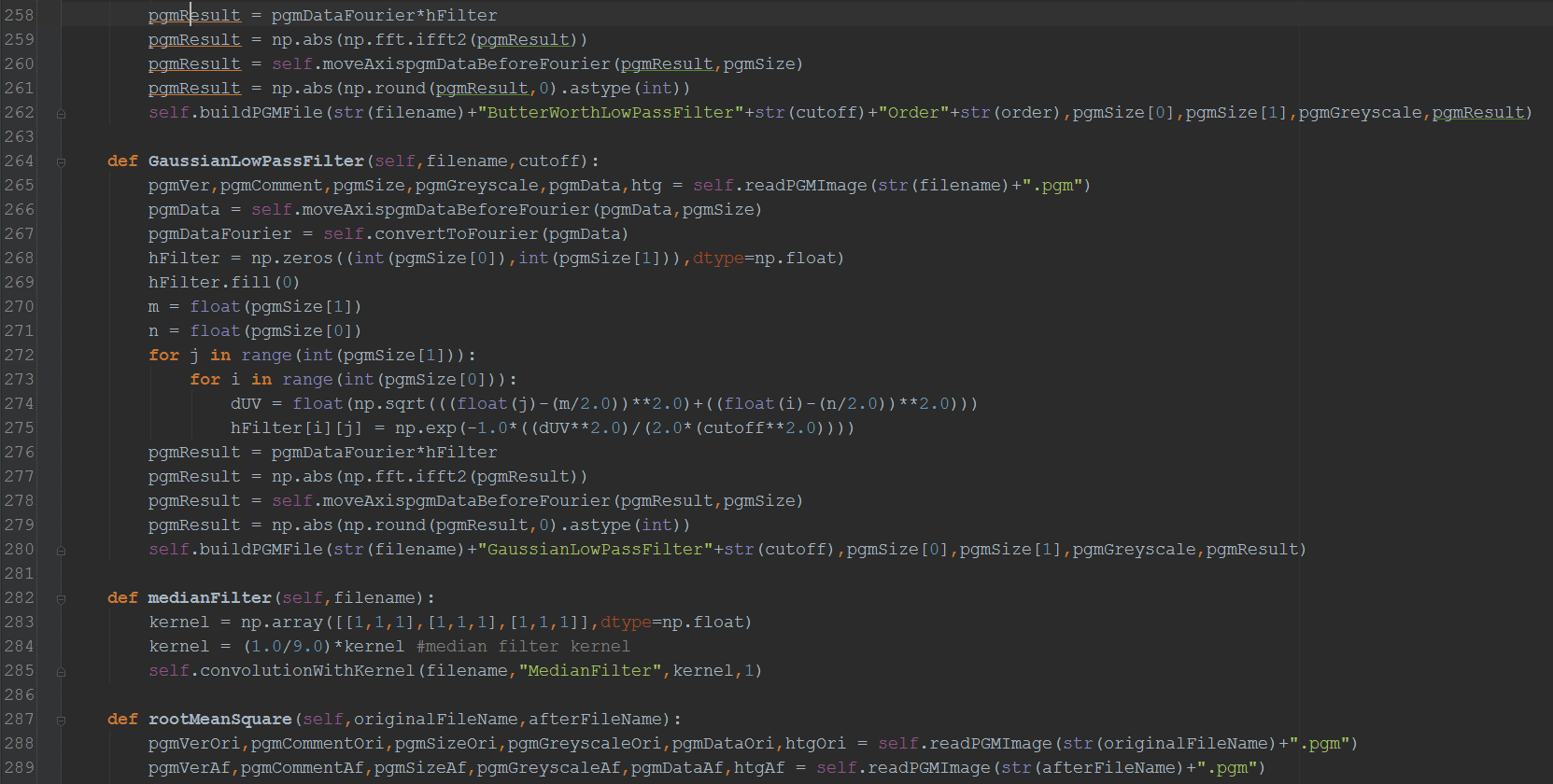


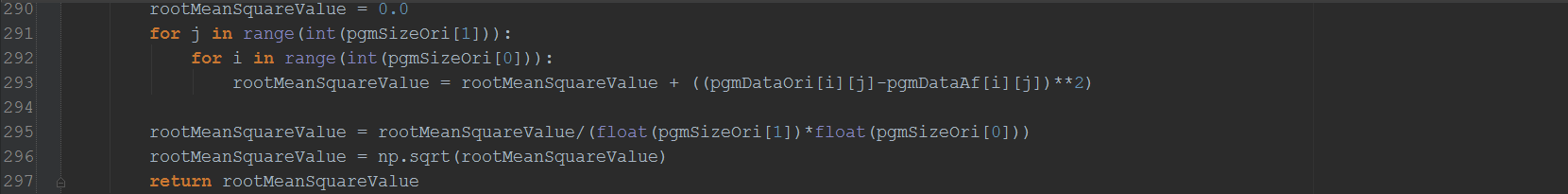












การเรียกใช้ Library ในแต่ล่ะข้อ

1.1)

myLib = ImageLibFourier()

myLib.padImage("Cross.pgm",28)

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage("CrossPadding.pgm")

pgmData = myLib.moveAxispgmDataBeforeFourier(pgmData,pgmSize)

pgmDataFourier = myLib.convertToFourier(pgmData)

myLib.pgmDataToPhasePicWithScale("CrossPadding",pgmDataFourier,pgmSize,pgmGreyscale)

myLib.pgmDataToAmplitudePicWithScale("CrossPadding",pgmDataFourier,pgmSize,pgmGreyscale)

1.2)

myLib = ImageLibFourier()

myLib.padImage("Cross.pgm",28)

myLib.shiftAxisInFourier("CrossPadding.pgm",20,30)

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage("CrossPaddingShiftAxis.pgm")

pgmData = myLib.moveAxispgmDataBeforeFourier(pgmData,pgmSize)

pgmDataFourier = myLib.convertToFourier(pgmData)

myLib.pgmDataToPhasePicWithScale("CrossPaddingShiftAxis",pgmDataFourier,pgmSize,pgmGreyscale)

myLib.pgmDataToAmplitudePicWithScale("CrossPaddingShiftAxis",pgmDataFourier,pgmSize,pgmGreyscale)

1.3)

myLib = ImageLibFourier()

myLib.rotatePic("Cross.pgm",-30)

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage("CrossRotatePic.pgm")

pgmData = myLib.moveAxispgmDataBeforeFourier(pgmData,pgmSize)

pgmDataFourier = myLib.convertToFourier(pgmData)

myLib.pgmDataToPhasePicWithScale("CrossRotatePic",pgmDataFourier,pgmSize,pgmGreyscale)

myLib.pgmDataToAmplitudePicWithScale("CrossRotatePic",pgmDataFourier,pgmSize,pgmGreyscale)

1.4)

myLib = ImageLibFourier()

myLib.downSample("Cross.pgm",0.5)

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage("CrossDownSamplePic.pgm")

pgmData = myLib.moveAxispgmDataBeforeFourier(pgmData,pgmSize)

pgmDataFourier = myLib.convertToFourier(pgmData)

myLib.pgmDataToPhasePicWithScale("CrossDownSamplePic",pgmDataFourier,pgmSize,pgmGreyscale)

myLib.pgmDataToAmplitudePicWithScale("CrossDownSamplePic",pgmDataFourier,pgmSize,pgmGreyscale)

1.5)

myLib = ImageLibFourier()

myLib.inverseFourierPgmWithOutAmplitude("CrossPadding.pgm")

myLib.inverseFourierPgmWithOutPhase("CrossPadding.pgm")

1.6)

myLib = ImageLibFourier()

myLib.inverseFourierPgmWithOutAmplitude("Lenna.pgm")

myLib.inverseFourierPgmWithOutPhase("Lenna.pgm")

1.7)

kernel = np.array([[1,2,1],[2,4,2],[1,2,1]],dtype=np.float)

kernel = (1.0/16.0)\*kernel

myLib = ImageLibFourier()

myLib.convolutionWithKernel("Chess","Blur",kernel,1)

myLib.convolutionWithKernelFrequencyDomain("Chess","BlurInFourier",kernel)

2.1)

myLib = ImageLibFourier()

myLib.idealLowPassFilter("Cross",20)

myLib.idealLowPassFilter("Cross",30)

myLib.idealLowPassFilter("Cross",50)

myLib.butterWorthLowPassFilter("Cross",20,2)

myLib.butterWorthLowPassFilter("Cross",30,2)

myLib.butterWorthLowPassFilter("Cross",50,2)

myLib.butterWorthLowPassFilter("Cross",20,20)

myLib.butterWorthLowPassFilter("Cross",20,5)

myLib.butterWorthLowPassFilter("Cross",20,100)

myLib.GaussianLowPassFilter("Cross",20)

myLib.GaussianLowPassFilter("Cross",30)

myLib.GaussianLowPassFilter("Cross",50)

2.2)

myLib = ImageLibFourier()

myLib.medianFilter("Lenna\_noise")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseConMedianFilter")

myLib.idealLowPassFilter("Lenna\_noise",20)

myLib.idealLowPassFilter("Lenna\_noise",30)

myLib.idealLowPassFilter("Lenna\_noise",50)

myLib.butterWorthLowPassFilter("Lenna\_noise",20,2)

myLib.butterWorthLowPassFilter("Lenna\_noise",30,2)

myLib.butterWorthLowPassFilter("Lenna\_noise",50,2)

myLib.GaussianLowPassFilter("Lenna\_noise",20)

myLib.GaussianLowPassFilter("Lenna\_noise",30)

myLib.GaussianLowPassFilter("Lenna\_noise",50)

print myLib.rootMeanSquare("Lenna","Lenna\_noise")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseidealLowPassFilter20")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseidealLowPassFilter30")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseidealLowPassFilter50")

print myLib.rootMeanSquare("Lenna","Lenna\_noise")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseButterWorthLowPassFilter20Order2")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseButterWorthLowPassFilter30Order2")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseButterWorthLowPassFilter50Order2")

print myLib.rootMeanSquare("Lenna","Lenna\_noise")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseGaussianLowPassFilter20")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseGaussianLowPassFilter30")

print myLib.rootMeanSquare("Lenna","Lenna\_noiseGaussianLowPassFilter50")

myLib.idealLowPassFilter("Chess\_noise",20)

myLib.idealLowPassFilter("Chess\_noise",30)

myLib.idealLowPassFilter("Chess\_noise",50)

myLib.butterWorthLowPassFilter("Chess\_noise",20,2)

myLib.butterWorthLowPassFilter("Chess\_noise",30,2)

myLib.butterWorthLowPassFilter("Chess\_noise",50,2)

myLib.GaussianLowPassFilter("Chess\_noise",20)

myLib.GaussianLowPassFilter("Chess\_noise",30)

myLib.GaussianLowPassFilter("Chess\_noise",50)

print myLib.rootMeanSquare("Chess","Chess\_noise")

print myLib.rootMeanSquare("Chess","Chess\_noiseidealLowPassFilter20")

print myLib.rootMeanSquare("Chess","Chess\_noiseidealLowPassFilter30")

print myLib.rootMeanSquare("Chess","Chess\_noiseidealLowPassFilter50")

print myLib.rootMeanSquare("Chess","Chess\_noise")

print myLib.rootMeanSquare("Chess","Chess\_noiseButterWorthLowPassFilter20Order2")

print myLib.rootMeanSquare("Chess","Chess\_noiseButterWorthLowPassFilter30Order2")

print myLib.rootMeanSquare("Chess","Chess\_noiseButterWorthLowPassFilter50Order2")

print myLib.rootMeanSquare("Chess","Chess\_noise")

print myLib.rootMeanSquare("Chess","Chess\_noiseGaussianLowPassFilter20")

print myLib.rootMeanSquare("Chess","Chess\_noiseGaussianLowPassFilter30")

print myLib.rootMeanSquare("Chess","Chess\_noiseGaussianLowPassFilter50")

myLib.medianFilter("Chess\_noise")

print myLib.rootMeanSquare("Chess","Chess\_noiseConMedianFilter")