import numpy as np  
import cv2  
import copy  
  
  
def smoothFilter(img, imgS, filS):  
 temp = []  
 imgTempf = copy.deepcopy(img)  
 for i in range(imgS[0]):  
 for j in range(imgS[1]):  
 imgTempf = conv(img, imgTempf, imgS, filS, temp, i, j)  
  
 return imgTempf  
  
  
def conv(imgTempff, imgNew, imgSS, filSS, tempp, ii, jj):  
 for l in range(filSS[0]):  
 if ii + filSS[0] < imgSS[0]:  
 for m in range(filSS[1]):  
 if jj + filSS[1] < imgSS[1]:  
 tempp.append(imgTempff[ii + l][jj + m])  
 else:  
 break  
 else:  
 break  
  
 temppSum = sum(tempp)  
 cenPointx = int(filSS[0] / 2)  
 cenPointy = int(filSS[1] / 2)  
 num = int(temppSum / (filSS[0] \* filSS[1]))  
 if (ii + filSS[0] < imgSS[0]) and (jj + filSS[1] < imgSS[1]):  
 # print('imgTempf[', ii + 1, '][', jj + 1, ']: ', imgTempff[ii + 1][jj + 1], 'temppSum: ', temppSum)  
 imgNew[ii + cenPointx][jj + cenPointy] = num  
 tempp.clear()  
 temppSum = 0  
 return imgNew  
  
  
image = cv2.imread('lenna\_rgb.png')  
imgGray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
imgTemp = copy.deepcopy(imgGray)  
imgTempResult = copy.deepcopy(imgGray)  
arr = np.array(imgTemp)  
imgSize = arr.shape  
  
# print(imgTemp)  
print('imSize: ', imgSize)  
  
fNum = 5  
kernel = []  
for i in range(fNum):  
 kernel.append([1] \* fNum)  
  
fil = np.array(kernel)  
filSize = fil.shape  
print('kernel: ', kernel)  
print('imSize: ', filSize)  
  
imgTemp2 = smoothFilter(imgTemp, imgSize, filSize)  
for i in range(imgSize[0]):  
 for j in range(imgSize[1]):  
 imgTemp[i][j] = int(imgGray[i][j]) - int(imgTemp2[i][j])  
 if imgTemp[i][j] > 255:  
 imgTemp[i][j] = 255  
 elif imgTemp[i][j] < 0:  
 imgTemp[i][j] = 0  
 imgTempResult[i][j] = int(imgGray[i][j]) + int(imgTemp[i][j])  
 if imgTempResult[i][j] > 255:  
 imgTempResult[i][j] = 255  
 elif imgTempResult[i][j] < 0:  
 imgTempResult[i][j] = 0  
  
  
print('imSize: ', imgSize)  
print('end')  
  
cv2.imshow('Before', imgGray)  
cv2.imshow('Average', imgTemp2)  
cv2.imshow('Detail', imgTemp)  
cv2.imshow('Result', imgTempResult)  
cv2.waitKey(0)