图像增强的问题咨询：

# Laplace 增强后的效果不佳，卷积后对于超出灰度范围的哪种方式处理能够得出较好的图像：

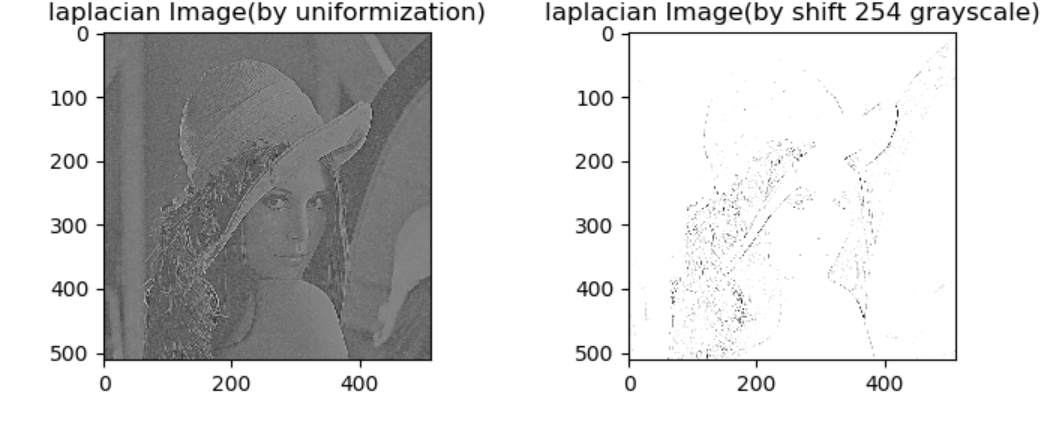
使用[-1, -1, -1]

[-1, 9, -1]

[-1, -1, -1]

对lena的图片进行卷积操作，得出的像素级范围是：[-420 , 811],超出8bit的灰度级范围，采用归一化处理，压缩至0~255之间和使用增加254个灰度，然后将超出范围的设置为边界值（小于0的设置为0，大于255的设置为255）两种方式，得出下面两幅图。效果都不是很好，请问卷积后，如何处理超出范围的灰度，能够得出较好的图像。

Ps:类似的问题，包括roberts，sobel等算法在执行时，abs后的结果，也可能超过255，如何处理？



# 代数运算—加法：去除叠加性噪声

通过加法去除叠加性噪声，根据定义，需要几个前提条件：

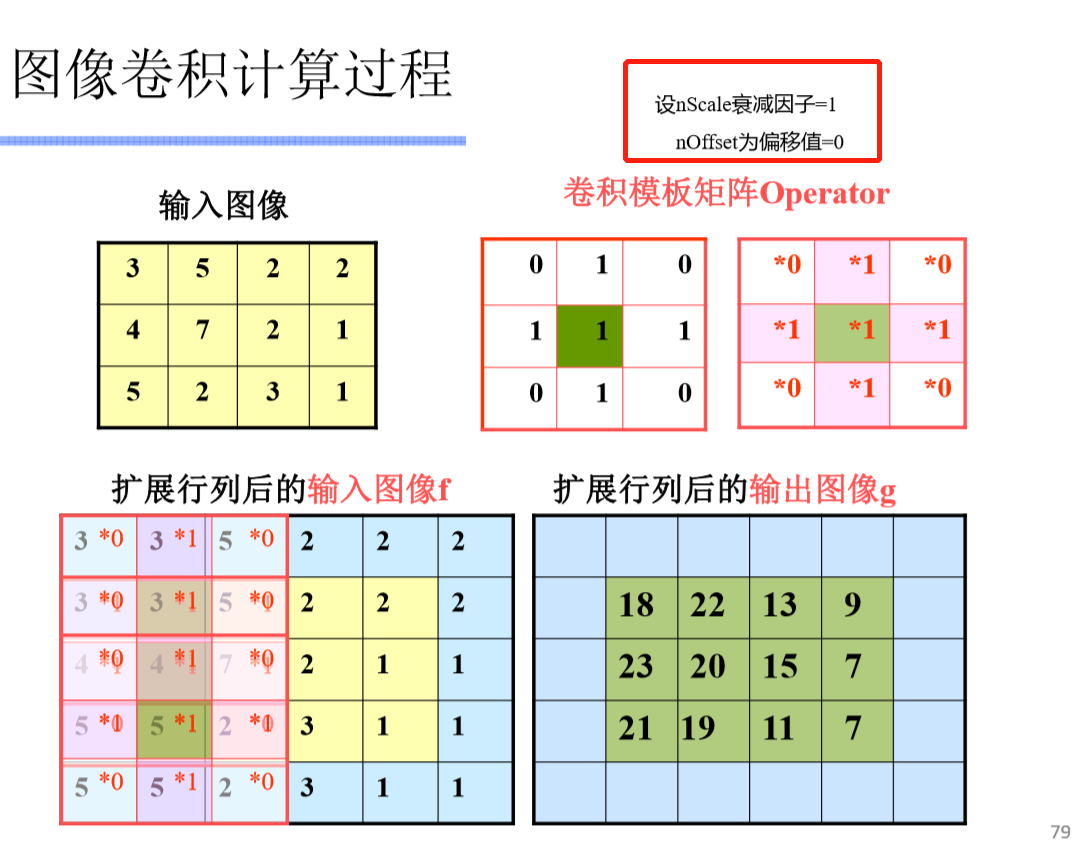
1）原图f(x,y)要基本不变化，2）噪声h(x,y)要求数学期望为0，3）要有一定数量N张这样的随机图片。

按照这个条件存在2个问题：

1）一个禁止不变的原图f(x,y)，这种场景很难找（ppt中的星系照片算一个，还有一类是超高速照相机，瞬时拍摄N张照片），适用面比较窄；2）如何确定噪声的数学期望时0？如果无法确定噪声的数学期望时0，那就无法确定是否可用加法去除叠加性噪声。

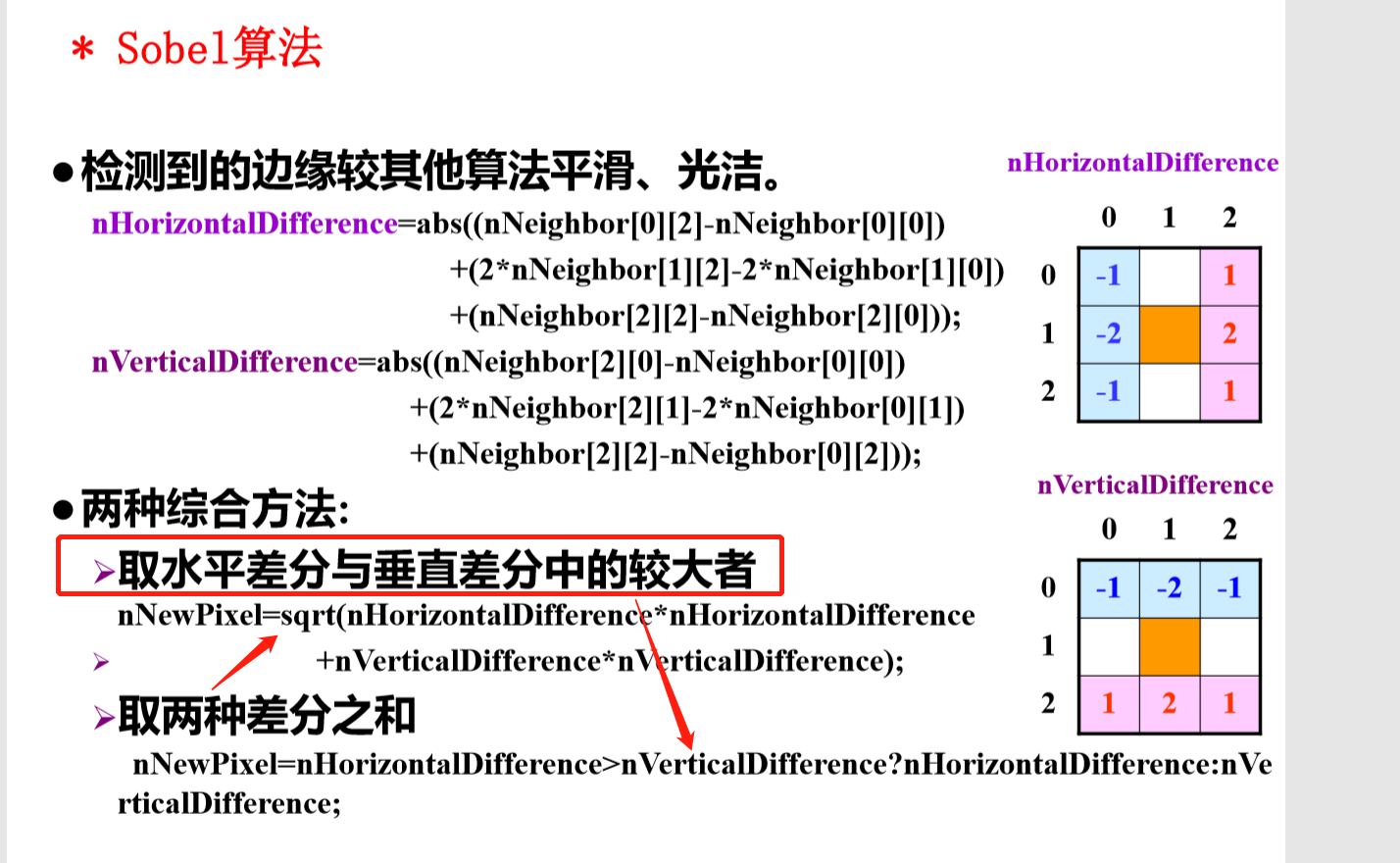
# 卷积计算中的衰减因子和偏移值是什么？

Ppt中卷积模板可以理解，但右上角的衰减因子和偏移值是什么意思？

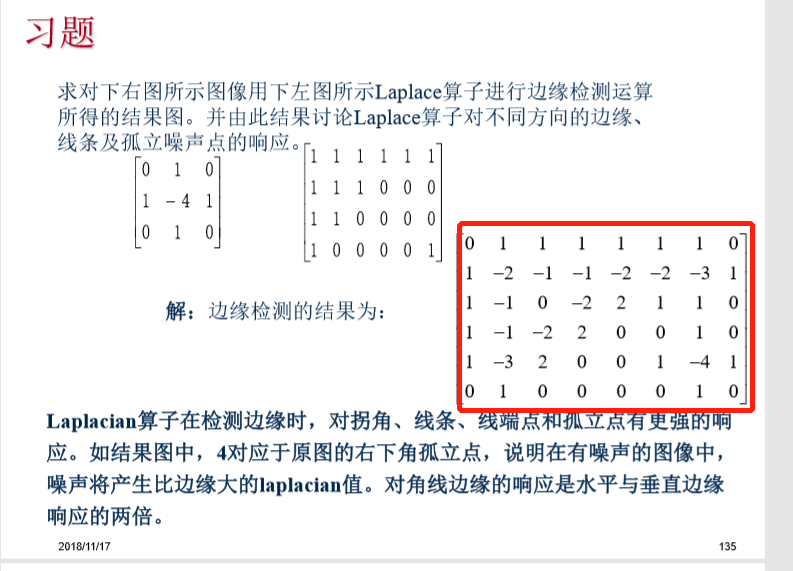


# Sobel算法综合方法描述有误：

Ppt中的sobel算法章节，综合方法：文字与公式关系好像反了？



# P135习题解不准确：



用十字星的Lapace算子对图像边缘检测，为了计算方便，一般会对图像进行扩边，扩边的方式为边复制和四周加0边。根据算法，计算出来的结果矩阵的shape应该与原矩阵一致。从现在给的结果看，应该是加了0边，其计算结果应该是：

[-2. -1. -1. -2. -2. -3.]

[-1. 0. -2. 2. 1. 1.]

[-1. -2. 2. 0. 0. 1.]

[-3. 2. 0. 0. 1. -4.]

Ps：如何确定是选择复制边方式还是加0边的方式扩充边？

#2019年10月3日  
#第三章ppt，p135 计算。  
import numpy as np  
  
##滤波卷积操作：  
# 定义函数指定一个原图，使用规定的3\*3模板进行滤过操作。  
# 采用复制边方式进行处理。  
def fn\_filter33 (orgImg, filterImg, method="sum", edgeModel="copy"):  
 rows,cols = orgImg.shape  
 #为原图扩展边，以便于计算，扩展方式为边复制.  
 extentImg = np.zeros((rows+2, cols+2), dtype=np.uint8)  
 extentImg[1:rows+1, 1:cols+1] = orgImg #中间部分用原图填充  
  
 if(edgeModel == "copy"):  
 extentImg[0, 1:cols+1]=orgImg[0, :] #第一行复制img的第一行  
 extentImg[rows+1, 1:cols+1]=orgImg[rows-1, :]#最后一行复制img的最后一行  
 extentImg[1:rows+1, 0]=orgImg[:, 0] #第一列复制过来  
 extentImg[1:rows+1, cols+1]=orgImg[:, cols-1] #最后一列  
 #补齐四个角  
 extentImg[0, 0]=orgImg[0, 0]  
 extentImg[0, cols+1] = orgImg[0, cols-1]  
 extentImg[rows+1, 0] = orgImg[rows-1, 0]  
 extentImg[rows+1, cols+1] = orgImg[rows-1, cols-1]  
 else:#添加0边  
 extentImg[0,:] = np.array([0]\*(cols+2))  
 extentImg[rows+1, :] = extentImg[0,:]  
 extentImg[:, 0] = np.array([0]\*(rows+2)).reshape((rows+2, 1))[:,0]  
 extentImg[:, cols+1] =extentImg[:, 0]  
  
 meanImg = np.zeros(orgImg.shape)  
   
 if method == "sum" :  
 for i in np.arange(1, rows+1):  
 for j in np.arange(1, cols+1):  
 rect = filterImg\*extentImg[i-1:i+2, j-1:j+2] #i+2的原因是切片时截至数不包括在里面  
 meanImg[i-1, j-1] = np.sum(rect) #np.uint8(abs(np.sum(rect))) #均值： 如果是十字星型模板，则不能使用mean方法，需要计算sum，除以十字星模板的sum  
 return meanImg  
   
#3\*3的模板滤波器  
templateMean = np.array([[0, 1, 0],  
 [1, -4, 1],  
 [0, 1, 0]])  
originalImg = np.array([[1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 0, 0, 0],  
 [1, 1, 0, 0, 0, 0],  
 [1, 0, 0, 0, 0, 1]])  
  
print("copy edge image ")  
copysEdgeImg = fn\_filter33(originalImg, templateMean, method="sum")  
print(copyEdgeImg)  
print("add zeros edge image")  
addZerosImg = fn\_filter33(originalImg, templateMean, edgeModel="zero")  
print(addZerosImg)