2019 年秋季　图像处理与分析 编程作业 02

【问题 1 图像二维卷积函数】

函数说明:

输入:原始图像矩阵 f，卷积核矩阵 w，填充方式 padding

输出：卷积运算后得到的图像矩阵 y

运行流程:

（1）读取图像矩阵 f 以及卷积核 w，获取图像尺寸[m n]及卷积核尺寸[p q]；

（2）为了使得最终图像大小不变，初始化一个更大的零矩阵 x，其尺寸在 f 基础上扩充为（m+p-1）*（n+q-1）。同时将原图像矩阵值利用 for 循环复制到 x 正中间，边缘等待填充；

（3）if 语句判断：若 padding='zero'，则不操作，边缘皆为 0；若 padding='replicate'，分别对上、下、左、右以及四个角这 8 个矩阵进行赋值，完成填充；

（4）将卷积核矩阵 w 上下、左右翻转得到新的核 new_w；

（5）利用 for 循环，将新的核 new_w 在扩充图像 x 上滑动，作矩阵的对应元素相乘，结果存储在输出矩阵 y 中；

（6）对输出矩阵 y 中元素进行处理，留下 0-255 的 uint8 型灰度值。

详细代码：

```python
import numpy as np
#卷积函数定义，缺省填充方式为zero
def twodConv(f,w,padding='zero'):
    [m,n] = f.shape
    [p,q] = w.shape
    x = np.zeros((m + p - 1, n + q - 1))      #初始化图像填充矩阵
    y = np.zeros((m,n))                       #初始化m*n输出矩阵
    for i in range(0, m):
        for j in range(0, n):
            x[i + p//2][j + q//2] = f[i][j]  #在新矩阵内部的m*n矩阵中复制原图像
    if padding=='zero':
        pass
    if padding=='replicate':#填充方式为复制
        del_h=p//2          #需要填充p行，图片上下各有p//2行
        del_w=q//2          #需要填充q列，图片左右各有q//2列
        for i in range(del_h):
            x[i,del_w:del_w+n] = f[0,:]             #填充图片正上方矩阵（复制第一行）
            x[m+del_h+i,del_w:del_w+n] = f[-1,:]    #填充图片正下方矩阵（复制最后一行）
        for j in range(del_w):
            x[del_h:m+del_h,j]=f[:,0]               #填充图片正左方矩阵（复制第一列）
            x[del_h:m+del_h,n+del_w+j]=f[:,-1]      #填充图片正右方矩阵（复制最后一列）

        x[0:del_h,0:del_w] = f[0,0]               #左上角小矩阵填充
        x[0:del_h,n+del_w:n+q] = f[0,-1]          #右上角小矩阵填充
        x[m+del_h:m+p,0:del_w] = f[-1,0]          #左下角小矩阵填充
        x[m+del_h:m+p,n+del_w:n+q] = f[-1,-1]     #右下角小矩阵填充
    #将卷积核矩阵左右、上下翻转
    new_w=w.reshape(w.size)
    new_w=new_w[::-1]
    new_w=new_w.reshape(w.shape)
    #翻转后的卷积核在图像上扫描
    for i in range(0,m):
        for j in range(0,n):
            y[i,j]=np.sum(x[i:i+p,j:j+q]*new_w)

    # 去掉矩阵乘法后的小于0的和大于255的原值,重置为0和255
    y = y.clip(0, 255)
    y = np.rint(y).astype('uint8')

    return y
```

【问题 2　归一化二维高斯滤波核函数】

函数说明:

输入:高斯核的标准差 sig，高斯核的长（宽）度 m

输出：m*m 的高斯滤波核矩阵

运行流程:

（1）判断 m 的大小：若 m 为默认值（此处设为 0），则按公式根据 sig 计算 m。若 m 小于公式规范值，则提示重新输入并终止所有程序;

（2）初始化高斯核矩阵为 m*m 的零矩阵 kernel，计算矩阵的中心位置 center=m//2;

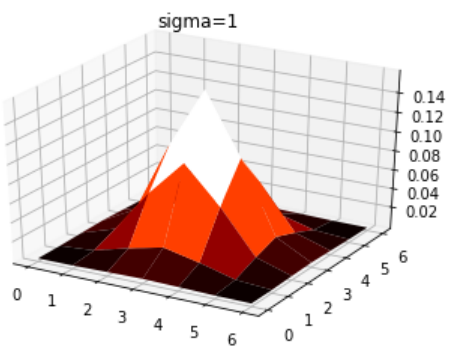
（3）利用 for 循环，根据核矩阵中每一点与中心的距离（i-center,j-center）、利用高斯公式计算每一个元素的数值 kernel[i , j]。同时计算所有元素值总和 sum;

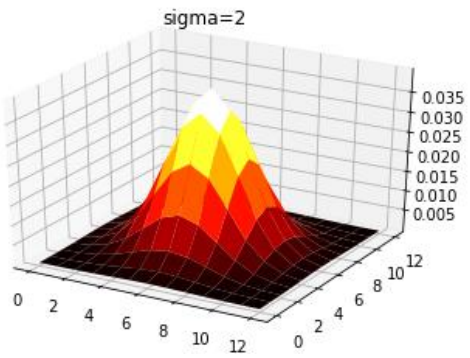
（4）对矩阵进行归一化——所有元素除以元素总和 sum，得到最终的高斯核。

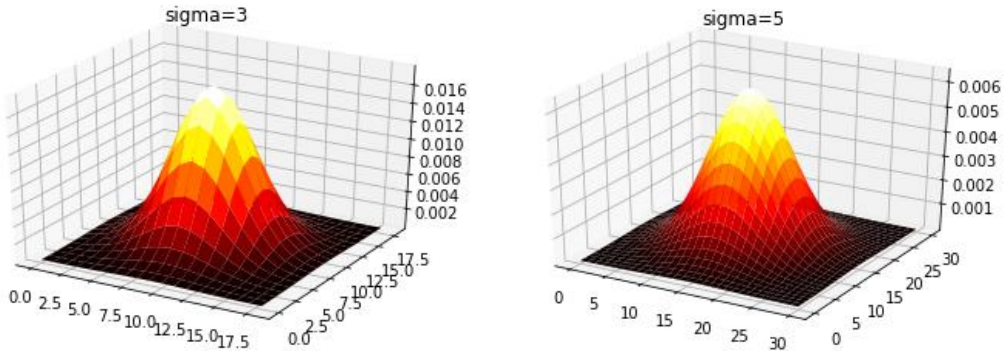
详细代码:

```
import sys
import numpy as np
def gaussKernel(sig,m=0):
    if m==0:#当m没有输入时默认值为0，此时根据公式计算m大小
        m=round(3*sig)*2+1
    if m<(round(3*sig)*2+1):#如果m输入值过小，则程序运行终止，提示重新输入m值
        print("输入的m值过小，请重新输入m")
        sys.exit(0)
        return 0

    kernel=np.zeros((m,m))#一个m*m的零矩阵作为高斯核的初始化矩阵
    center=m//2           #计算中心位置
    s=sig**2
    sum=0
    for i in range(m):
        for j in range(m):
            x, y = i-center, j-center                #核内元素到中心点的距离
            kernel[i, j] = np.exp(-(x**2+y**2)/(2*s))#高斯公式计算滤波核元素值
            sum += kernel[i, j]                      #高斯核所有元素总和
    kernel = kernel/sum    #归一化处理
    return kernel
```

为了比较清晰地看出这个函数生成的高斯滤波核实际分布情况，这里列出了 4 种标准差以及尺寸对应的高斯核矩阵的 3 维图像，如下所示:







【问题 3　灰度图像的高斯滤波】

（1）分别取 sigma=1,2,3,5，对 4 张灰度图像进行高斯卷积：（这里选择了 replicate 填充）

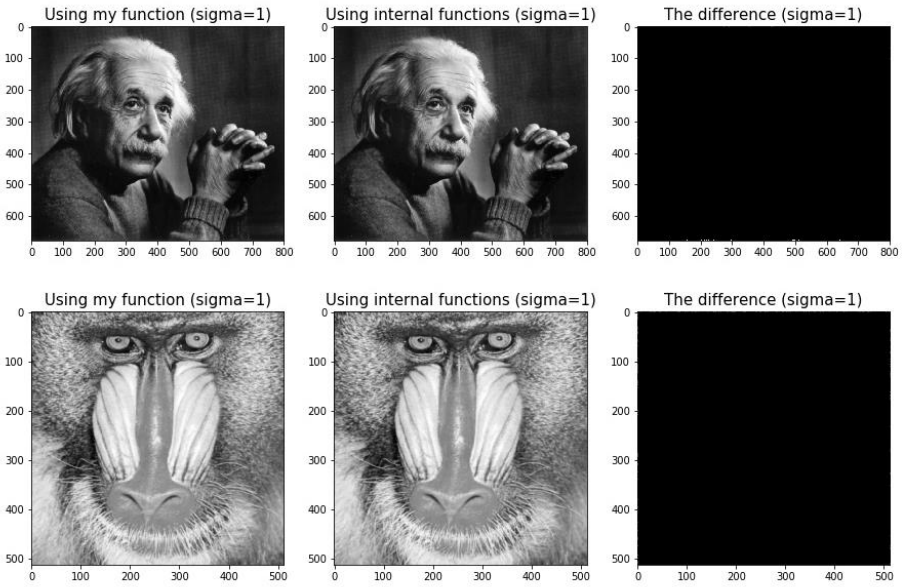
可以看出，随着高斯滤波核的标准差增大，滤波的效果愈发明显，图像细节被柔化、模糊。适当的高斯滤波可以使得图像中一些无规则的噪声被滤除，但过度的滤波会使得原图像本身信息受到影响。

（2）基于 sig=1 的高斯核，比较自编函数与内部原有函数的效果：

调用 opencv 中现有的一维高斯核生成函数 getGaussianKernel（kernelsize，sigma），稍作组合得到现成的二维高斯核生成函数；利用 opencv 中的 filter2d 函数实现二维卷积运算。部分代码如下：

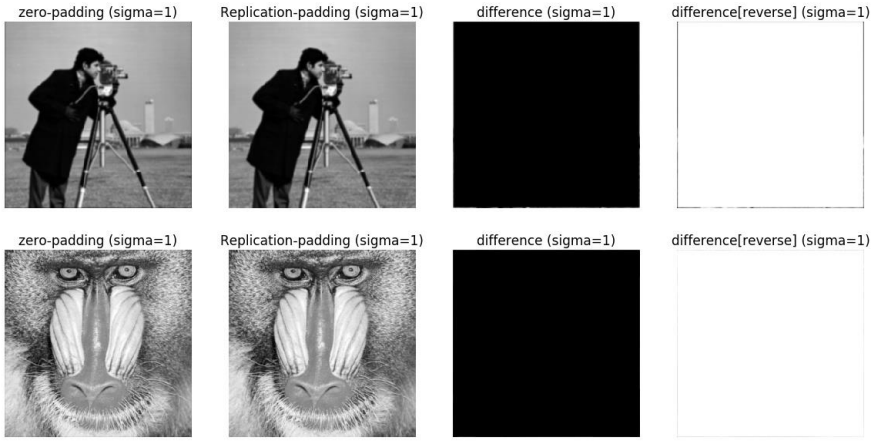
```
#利用自带函数进行高斯核卷积操作，其中sigma=1
def gaussian_kernel_2d_opencv(kernel_size = 3,sigma = 0):
    kx = cv2.getGaussianKernel(kernel_size,sigma)
    ky = cv2.getGaussianKernel(kernel_size,sigma)
    return np.multiply(kx,np.transpose(ky)) #利用opencv库自带一维高斯核函数构造二维高斯核函数
wb=gaussian_kernel_2d_opencv(7,1)           #生成与w1相同尺寸、相同标准差的高斯核

img_outI = cv2.filter2D(img2, -1, wb)       #调用opencv中卷积函数-第一张图
img_difI=abs(img_out5-img_outI)
img_outII = cv2.filter2D(img4, -1, wb)      #调用opencv中卷积函数-第二张图
img_difII=abs(img_out13-img_outII)
```

根据上述函数以及前面自己编写的高斯核生成函数 gaussKernel（sig，m）、二维卷积函数 twodConv（f,w,padding），对图像分别进行高斯滤波，并对图像结果进行差值比较（差值后取了绝对值，消除了负值）。如下图所示，可以看出两者几乎没有差别，说明自行编写的函数比较理想。



（3）其他参数不变，比较复制和补零两种方式：

这里参数取值为 sig=1，m=7，分别作出如下图：



为了能够看得更清楚，对差值图像取了反，可以看出两种方式得到的图像整体几乎没有差别，但是在图像的外边界存在差异，能够看到清晰的边界。这是因为输出图像边缘的点其灰度值的卷积计算中包含了填充的像素点，而图像内部的点卷积并不会受影响。