Task4 图像滤波

一、背景知识

空间域：指图像平面本身，这类处理方法直接以图像中的像素操作为基础。分为灰度变换和空间滤波两类。

变换域：首先把一幅图像变换到变换域，在变换域中进行处理，然后通过反变换把处理结果返回到空间域。

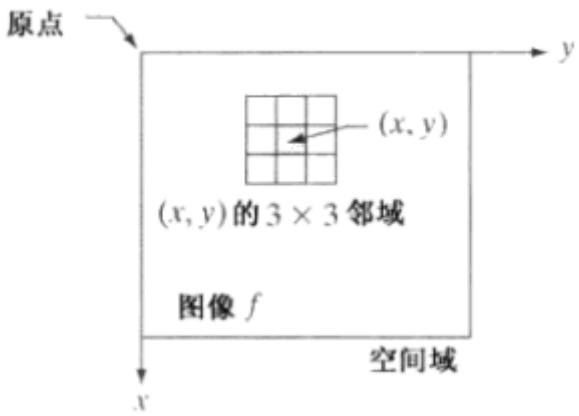
处理类型	操作
灰度变换	在图像的单个像素上操作，主要以对比度和阈值处理为目的
空间滤波	涉及到改善性能的操作

空间域的处理形式：

$$g(x,y) = T[f(x,y)]$$

其中，f(x,y)是输入图像，g(x,y)是处理后的图像，T是在点(x,y)的邻域上定义的关于f的一中算子。

下面给出处理步骤：



如图所示，(x,y)是图像中的一个任意位置，包含该点的小区域是点(x,y)的邻域，是一个中心在(x,y)的矩形。

空间滤波的过程：邻域原点从一个像素向另一个像素移动，对邻域中的像素应用算子T，并产生输出。

详细请见冈萨雷斯《数字图像处理》第三版63页。

二、空间滤波基础

图像的实质是一种二维信号，滤波是信号处理中的一个重要概念。在图像处理中，滤波是一种非常常见的技术，它们的原理非常简单，但是其思想却十分值得借鉴，滤波是很多图像算法的前置步骤或基础，掌握图像滤波对理解卷积神经网络也有一定帮助。

空间滤波器的别称：空间掩模、核、模板、窗口。

2.1 滤波分类

- **线性滤波**： 对邻域中的像素的计算为线性运算时，如利用窗口函数进行平滑加权求和的运算，或者某种卷积运算，都可以称为线性滤波。常见的线性滤波有：均值滤波、高斯滤波、盒子滤波、拉普拉斯滤波等等，通常线性滤波器之间只是模版系数不同。
- **非线性滤波**： 非线性滤波利用原始图像跟模版之间的一种逻辑关系得到结果，如最值滤波器，中值滤波器。比较常用的有中值滤波器和双边滤波器。

如图所示是一个3x3的线性滤波器的原理：

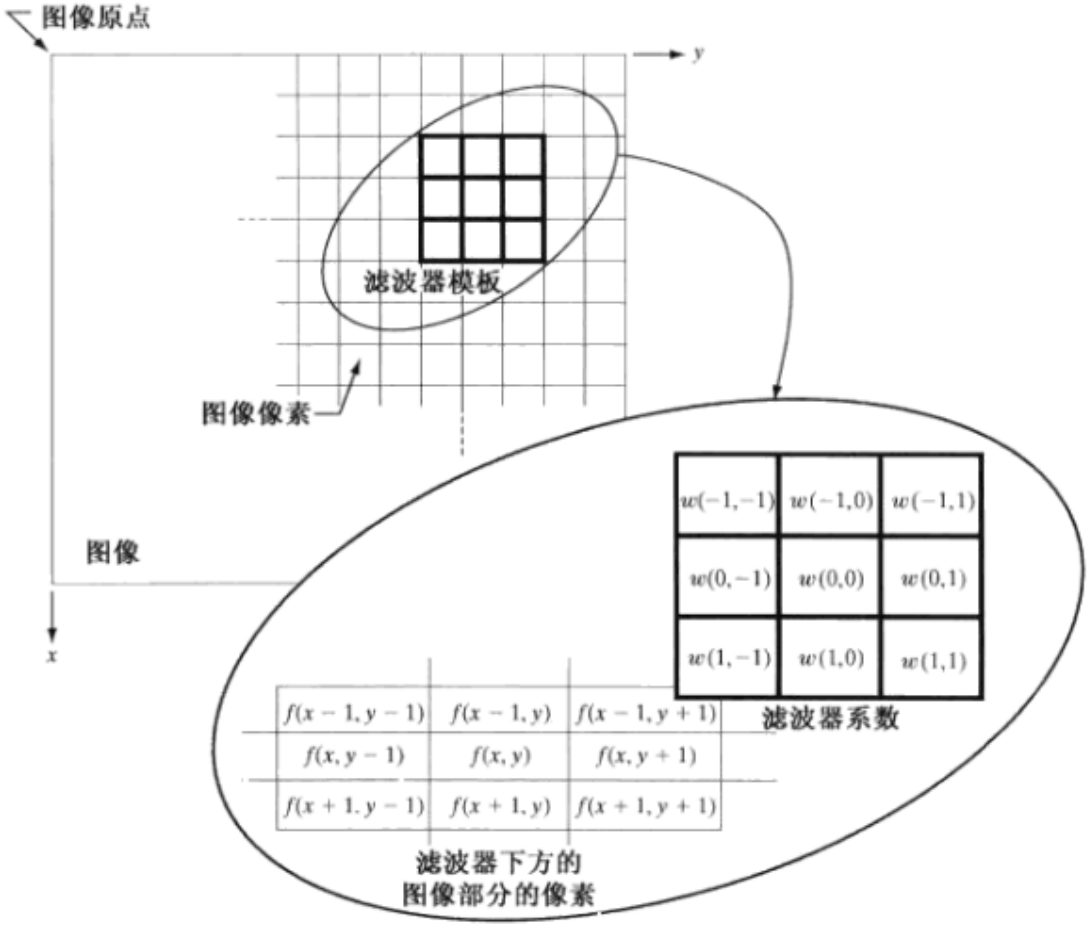


图 3.28　使用大小为 3×3 的滤波器模板的线性空间滤波的机理。表示滤波器模板系数的坐标所选择的形式简化了线性滤波的表达式

在图像中任意一个点(x,y)，滤波器的响应g(x,y)为：

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots + w(1,1)f(x+1,y+1)$$

一般来说，使用大小为mxn的滤波器对大小为MxN的图像进行线性空间滤波，可表示为：

$$g(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)$$

其中x和y是可变的，以便w中的每个像素可访问f中的每个像素。

2.2 空间相关和卷积

- **相关**：滤波器模板移过图像并计算每个位置乘积之和的处理。
- **卷积**：滤波器先旋转180°，再做相关运算。

三、线性滤波

3.1 方框（盒子）滤波

方框滤波是一种非常有用的线性滤波，也叫盒子滤波，均值滤波就是盒子滤波归一化的特殊情况。 **应用：** 可以说，一切需要求某个邻域内像素之和的场合，都有方框滤波的用武之地，比如：均值滤波、引导滤波、计算Haar特征等等。

优势： 就一个字：快！它可以使复杂度为O(MN)的求和，求方差等运算降低到O(1)或近似于O(1)的复杂度，也就是说与邻域尺寸无关了，有点类似积分图吧，但是比积分图更快（与它的实现方式有关）。

原理上，采用一个卷积核与图像进行卷积：

$$K = \alpha \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ & & \cdots\cdots\cdots & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

其中：

$$\alpha = \begin{cases} \frac{1}{ksize.width * ksize.height} & \text{when } \texttt{normalize=true} \\ 1 & \text{otherwise} \end{cases}$$

可见，归一化了就是均值滤波；不归一化则可以计算每个像素邻域上的各种积分特性，方差、协方差，平方和等等。

3.2 均值滤波

又称为平滑线性空间滤波器。使用滤波器模板确定的邻域内像素平均灰度值代替图像中每个像素的值，这种处理的结果降低了图像灰度的“尖锐”变化。常见的均值滤波应用就是**降低噪声**。然而，由于图像边缘也是由图像灰度尖锐变化带来的特性，因此均值滤波存在着不希望有的边缘模糊的负面效应。均值滤波的另一个应用是**去除图像中的不相关细节**，“不相关”指的是与滤波器模板尺寸相比较小的像素区域。（根据大佬的解释，这个又称为均值模糊，模糊图像以便得到感兴趣物体的粗略描述）。

均值滤波是方框滤波的特殊情况，均值滤波方法是：对待处理的当前像素，选择一个模板，该模板为其邻近的若干个像素组成，用模板的均值（方框滤波归一化）来替代原像素的值。公式表示为：

$$g(x,y) = 1/n \sum_{I \in Neighbour} I(x,y)$$

g(x,y)为该邻域的中心像素，n跟系数模版大小有关，一般3*3邻域的模板，n取为9，如：

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

当然，模板是可变的，一般取奇数，如5 * 5，7 * 7等等。

注：在实际处理过程中可对图像边界进行扩充，扩充为0或扩充为邻近的像素值。

即一般情况下，在图像中任意位置的灰度平均值是以(x,y)为中心的3x3邻域中的9各灰度值之和除以9。

均值滤波的缺陷：均值滤波本身存在着固有的缺陷，即它不能很好地保护图像细节，在图像去噪的同时也破坏了图像的细节部分，从而使图像变得模糊，不能很好地去除噪声点。特别是椒盐噪声。

> 椒盐噪声是一种随机出现的白点或者黑点，可能是亮的区域有黑色像素或是在暗的区域有白色像素（或是两者皆有），是脉冲噪声。常用的去除这种噪声的有效手段是使用中值滤波器。

3.3 高斯滤波

高斯滤波是一种线性平滑滤波器，对于服从正态分布的噪声有很好的抑制作用。在实际场景中，我们通常会假定图像包含的噪声为高斯白噪声，所以在许多实际应用的预处理部分，都会采用高斯滤波抑制噪声，如传统车牌识别等。

高斯滤波和均值滤波一样，都是利用一个掩膜和图像进行卷积求解。不同之处在于：均值滤波器的模板系数都是相同的为1，**而高斯滤波器的模板系数，则随着距离模板中心的增大而系数减小（服从二维高斯分布）**。所以，高斯滤波器相比于均值滤波器对图像个模糊程度较小，更能够保持图像的整体细节。

二维高斯分布公式：

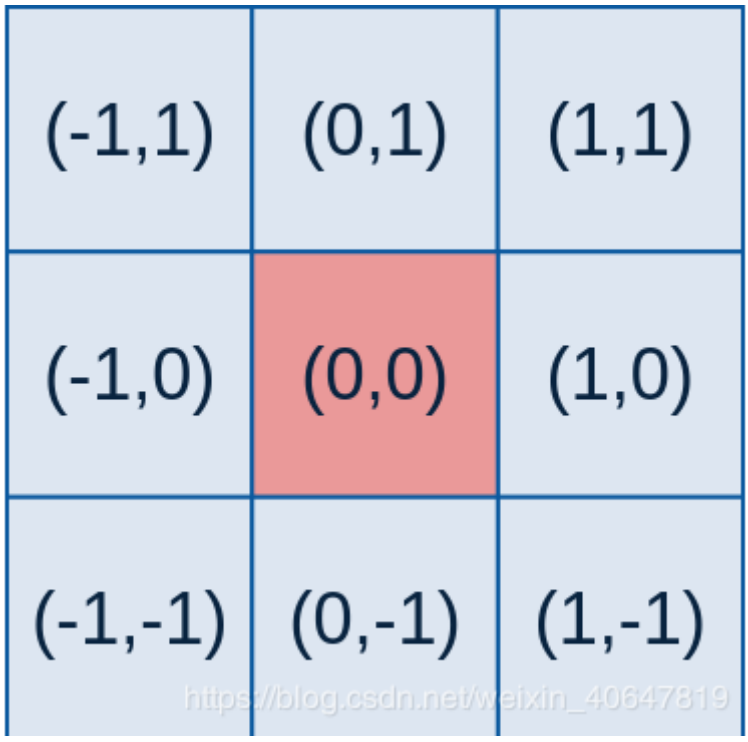
$$f(x,y)=\frac{1}{(\sqrt{2\pi}\sigma)^2}e^{-((x-ux)^2+(y-uy)^2)/2\sigma^2}$$

其中不必纠结于系数，因为它只是一个常数！并不会影响互相之间的比例关系，并且最终都要进行归一化，所以在实际计算时我们是忽略它而只计算后半部分：

$$f(x,y)=e^{-((x-ux)^2+(y-uy)^2)/2\sigma^2}$$

其中(x,y)为掩膜内任一点的坐标，(ux,uy)为掩膜内中心点的坐标，在图像处理中可认为是整数；σ是标准差。

例子：要产生一个3×3的高斯滤波器模板，以模板的中心位置为坐标原点进行取样。模板在各个位置的坐标，如下所示（x轴水平向右，y轴竖直向下）。



将各个位置的坐标带入到高斯函数中，得到的值就是模板的系数。 对于窗口模板的大小为(2k+1)×(2k+1)，模板中各个元素值的计算公式如下：

$$H_{i,j}=\frac{1}{2\pi\sigma^2}e^{-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}}$$

这样计算出来的模板有两种形式：小数和整数。

- 小数形式的模板，就是直接计算得到的值，没有经过任何的处理；
- 整数形式的，则需要进行归一化处理，将模板左上角的值归一化为1，具体介绍请看这篇博文。使用整数的模板时，需要在模板的前面加一个系数，系数为模板系数和的倒数。

生成高斯掩膜（小数形式）

知道了高斯分布原理，实现起来也就不困难了。

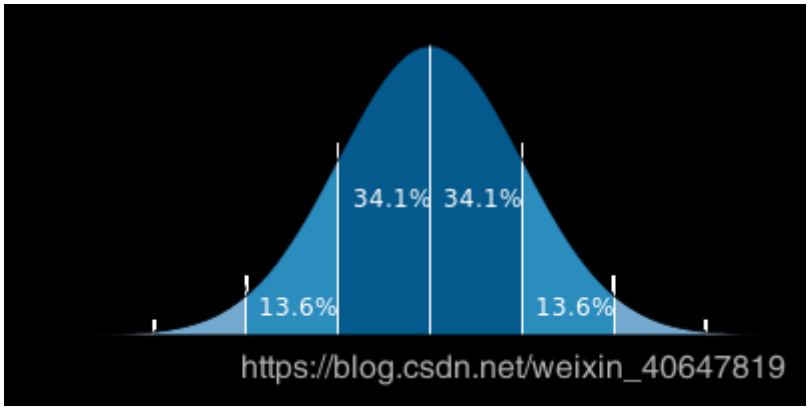
首先我们要确定我们生产掩模的尺寸wsize，然后设定高斯分布的标准差。生成的过程，我们首先根据模板的大小，找到模板的中心位置center。 然后就是遍历，根据高斯分布的函数，计算模板中每个系数的值。

最后模板的每个系数要除以所有系数的和。这样就得到了小数形式的模板。

σ的意义及选取

通过上述的实现过程，不难发现，高斯滤波器模板的生成最重要的参数就是高斯分布的标准差σ。标准差代表着数据的离散程度，如果σ较小，那么生成的模板的中心系数较大，而周围的系数较小，这样对图像的平滑效果就不是很明显；反之，σ较大，则生成的模板的各个系数相差就不是很大，比较类似均值模板，对图像的平滑效果比较明显。

来看下一维高斯分布的概率分布密度图：



于是我们有如下结论：

- σ越小分布越瘦高，σ越大分布越矮胖。
- σ越大，分布越分散，各部分比重差别不大，于是生成的模板各元素值差别不大，类似于平均模板；
- σ越小，分布越集中，中间部分所占比重远远高于其他部分，反映到高斯模板上就是中心元素值远远大于其他元素值，于是自然而然就相当于中间值得点运算。

关于σ和窗口大小的选择，可以参考这篇文章：https://www.cnblogs.com/shine-lee/p/9671253.html。

四、非线性滤波

又称为**统计排序滤波器**，这种滤波器的响应以滤波器包围的图像区域中所包含的像素的排序为急促，使用统计排序结果决定的值代替中心像素的值。

最知名的是中值滤波器，是将像素邻域内灰度的中值代替该像素的值。

实现过程（取自冈萨雷斯《数字图像处理》第三版96页）：

一个数值集合的中值ξ是这样的数值，即数值集合中有一半小于或等于ξ，还有一半大于或等于ξ。为了对一幅图像上的某个点进行中值滤波处理，首先将邻域内的像素分类排序，确定其中值，并将中值赋予滤波后图像中的相应像素点。例如，对于一个3×3 邻域，其中值是第 5 个最大的值，而在一个5×5 邻域中，中值就是第 13 个最大的值，等等。当一个邻域中的一些像素值相同时，所有相等的值都可以作为中值。假如，在一个3×3 邻域内有一系列像素值(10, 20, 20, 20, 15, 20, 20, 25, 100)，对这些值排序后为(10, 15, 20, 20, 20, 20, 20, 25, 100)，那么其中值就是 20。这样，中值滤波器的主要功能是使拥有不同灰度的点看起来更接近于它的相邻点。事实上，我们使用$m \times m$中值滤波器来去除那些相对于其邻域像素更亮或更暗并且其区域小于$m^2/2$（滤波器区域的一半）的孤立像素族。在这种情况下，“去除”的意思是强制为邻域的中值灰度。较大的族所受到的影响明显较小。

五、代码实现

给出python代码的实现

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('E:/PythonProgram/opencv_study/fig_transaction/opencv.PNG')
b, g, r = cv2.split(img)
img = cv2.merge([r, g, b]) # 这两行代码解决rgb排列的问题

# 1、均值滤波
blur = cv2.blur(img, (5,5))

# 2、高斯滤波
gblur = cv2.GaussianBlur(img,(5,5),0)

# 3、中值滤波
median = cv2.medianBlur(img, 5)

#4、盒子滤波
box = cv2.boxFilter(img, -1, (5,5))

plt.subplot(231),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(232),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.subplot(233),plt.imshow(gblur),plt.title('GaussianBlurred')
plt.xticks([]), plt.yticks([])
plt.subplot(234),plt.imshow(median),plt.title('MedianBlurred')
plt.xticks([]), plt.yticks([])
plt.subplot(235),plt.imshow(box),plt.title('BoxBlurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

得到的结果：

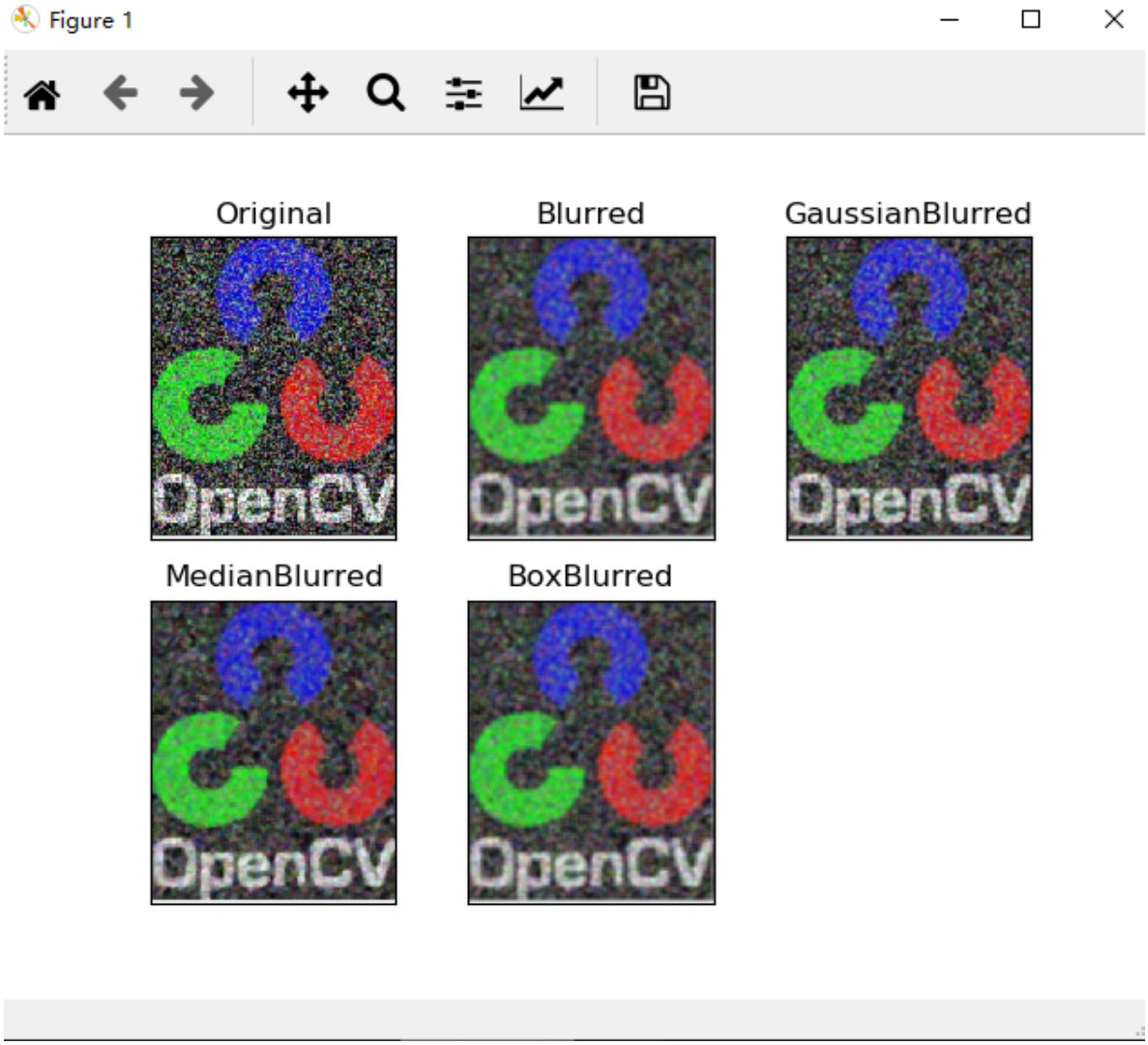


Figure 1
Original
Blurred
GaussianBlurred
MedianBlurred
BoxBlurred
OpenCV
OpenCV
OpenCV
OpenCV
OpenCV