### 数字图像处理大作业（二）

### sobel算子进行边缘检测

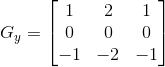
### 1、实验原理

Sobel算子其主要用于边缘检测，在技术上它是以离散型的差分算子，用来运算图像亮度函数的梯度的近似值， Sobel算子是典型的基于一阶导数的边缘检测算子，由于该算子中引入了类似局部平均的运算，因此对噪声具有平滑作用，能很好的消除噪声的影响。

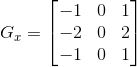
Sobel算子对于象素的位置的影响做了加权，与Prewitt算子、Roberts算子相比因此效果更好。

Sobel算子包含两组3x3的矩阵，分别为横向及纵向模板，将之与图像作平面卷积，即可分别得出横向及纵向的亮度差分近似值。实际使用中，常用如下两个模板来检测图像边缘。

检测水平边沿横向模板 ：



检测垂直平边沿纵向模板：



图像的每一个像素的横向及纵向梯度近似值可用以下的公式结合，来计算梯度的大小。è¿éåå¾çæè¿°

通常，为了提高效率使用不开平方的近似值：

https://img-my.csdn.net/uploads/201210/25/1351141094_3548.png

然后可用以下公式计算梯度方向。

https://img-blog.csdn.net/20170916212059963?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQveGRkd3o=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

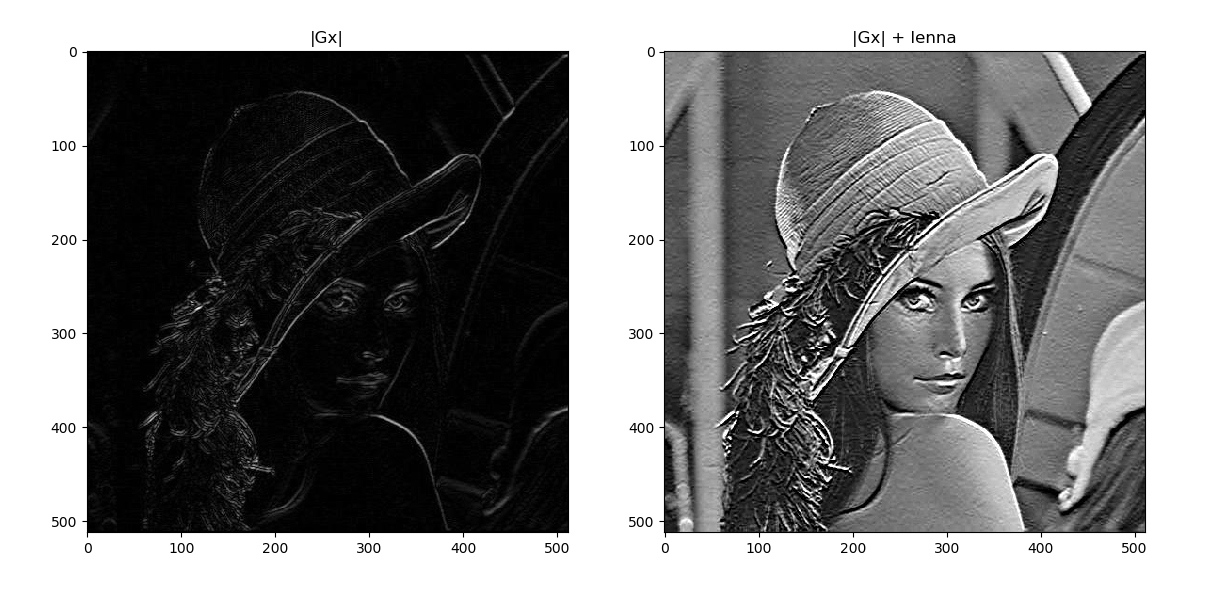
### 2、实验步骤

1、理解原理和算法

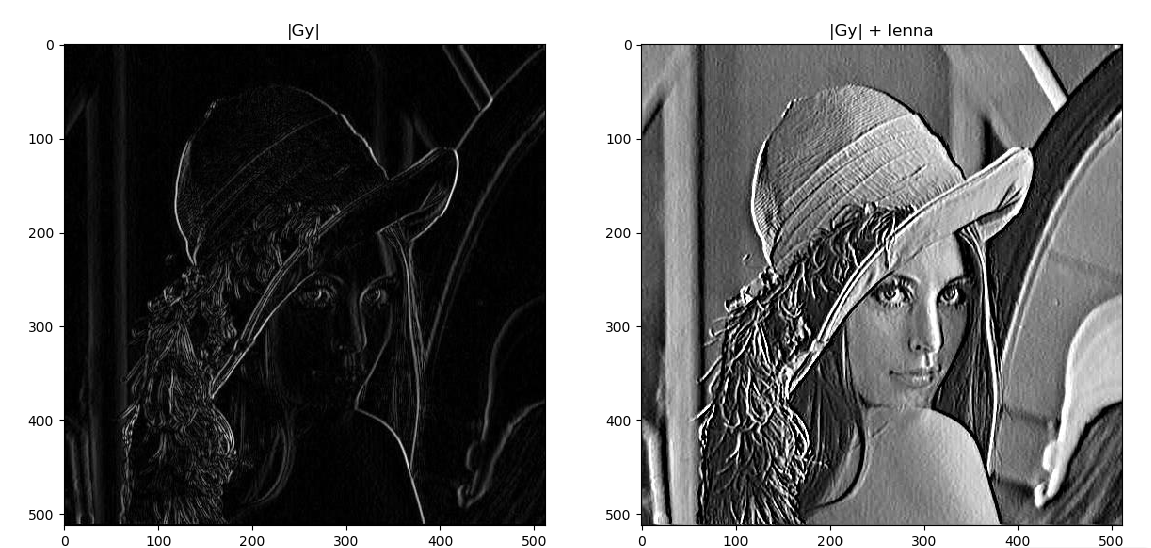
2、复用第一个大作业的函数，编码

### 3、实验结果和分析。

1、使用Gx模板检测水平边沿



2、使用Gy模板检测竖直边缘

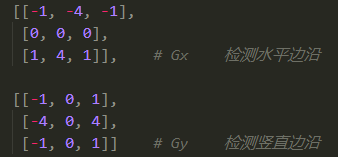


显然，检测竖直边缘算子对竖直边缘的检测更加清晰，很清楚的可以看到左边的细长的矩形边缘，而对水平边缘检测的图像就完全看不到。

3、梯度图像 **|Gx| + |Gy|**

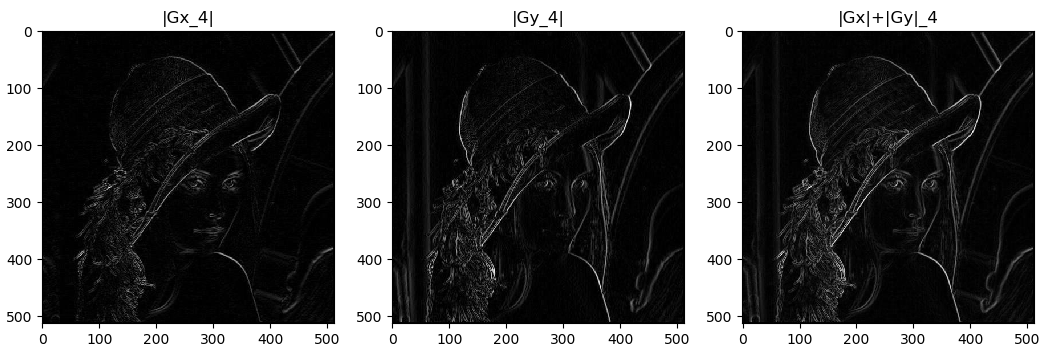
****

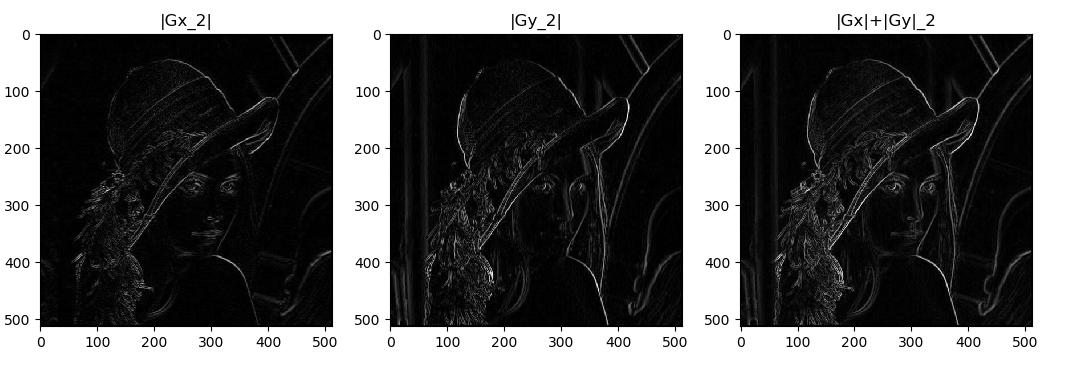
**4、更换算子参数如下**



得到图像如下：

通过比较两组不同的sobel算子我们发现，中间为4四的sobel算子，会比经典的sobel算子更加的锐化一些，这个比较好理解，因为相邻像素的差值随着中间值由2变为4而增加。





### Canny算子边缘检测

### 实验原理

Canny边缘检测于1986年由JOHN CANNY首次在论文《A Computational Approach to Edge Detection》中提出，就此拉开了Canny边缘检测算法的序幕。

Canny边缘检测是从不同视觉对象中提取有用的结构信息并大大减少要处理的数据量的一种技术，目前已广泛应用于各种计算机视觉系统。Canny发现，在不同视觉系统上对边缘检测的要求较为类似，因此，可以实现一种具有广泛应用意义的边缘检测技术。边缘检测的一般标准包括：

1)以低的错误率检测边缘，也即意味着需要尽可能准确的捕获图像中尽可能多的边缘。

2)检测到的边缘应精确定位在真实边缘的中心。

3)图像中给定的边缘应只被标记一次，并且在可能的情况下，图像的噪声不应产生假的边缘。

Canny边缘检测算法的处理流程：

1. 使用一个高斯滤波器平滑输入图像
2. 计算梯度幅值图像和角度图像
3. 对梯度图像应用非最大抑制
4. 用双阈值处理和连接分析来检测并连接边缘

### 2、实验过程

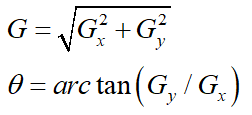
本次实验采用Python编程，利用numpy中 array做主要数据结构， matplotlib.pyplot进行可视化。

**高斯平滑滤波**

为了尽可能减少噪声对边缘检测结果的影响，所以必须滤除噪声以防止由噪声引起的错误检测。为了平滑图像，使用高斯滤波器与图像进行卷积，该步骤将平滑图像，以减少边缘检测器上明显的噪声影响。

**计算梯度强度和方向**

边缘检测的算子（如Roberts，Prewitt，Sobel等）返回水平Gx和垂直Gy方向的一阶导数值，由此便可以确定像素点的梯度G和方向theta 。

****

**非极大值抑制**

非极大值抑制是一种边缘稀疏技术，非极大值抑制的作用在于“瘦”边。对图像进行梯度计算后，仅仅基于梯度值提取的边缘仍然很模糊。对于标准3，对边缘有且应当只有一个准确的响应。而非极大值抑制则可以帮助将局部最大值之外的所有梯度值抑制为0，对梯度图像中每个像素进行非极大值抑制的算法是：

1) 将当前像素的梯度强度与沿正负梯度方向上的两个像素进行比较。

2) 如果当前像素的梯度强度与另外两个像素相比最大，则该像素点保留为边缘点，否则该像素点将被抑制。

**用双阈值算法检测和连接边缘**

对非极大值抑制图像作用两个阈值th1和th2，两者关系th1=0.4th2。我们把梯度值小于th1的像素的灰度值设为0，得到图像1。然后把梯度值小于th2的像素的灰度值设为0，得到图像2。由于图像2的阈值较高，去除大部分噪音，但同时也损失了有用的边缘信息。而图像1的阈值较低，保留了较多的信息，我们可以以图像2为基础，以图像1为补充来连结图像的边缘。

　链接边缘的具体步骤如下：

　对图像2进行扫描，当遇到一个非零灰度的像素p(x,y)时，跟踪以p(x,y)为开始点的轮廓线，直到轮廓线的终点q(x,y)。

　　考察图像1中与图像2中q(x,y)点位置对应的点s(x,y)的8邻近区域。如果在s(x,y)点的8邻近区域中有非零像素s(x,y)存在，则将其包括到图像2中，作为r(x,y)点。从r(x,y)开始，重复第一步，直到我们在图像1和图像2中都无法继续为止。

当完成对包含p(x,y)的轮廓线的连结之后，将这条轮廓线标记为已经访问。回到第一步，寻找下一条轮廓线。重复第一步、第二步、第三步，直到图像2中找不到新轮廓线为止。

　　至此，完成canny算子的边缘检测。

### 实验结果和分析

本实验中，设计很多的变量，下面列出不同的变量组合对实验结果的影响。

1. 高斯核模板：方差σ、模板大小n
2. 边缘检测算子的选取：sobel等
3. 双阈值处理高低阈值的选择
4. **高斯核模板**

在实验之前我们分析一下原始图像，由于lenna.jpg看起来并没有加噪声，猜想高斯滤波对最后结果的影响并不是很大。

我们可以使用前面章节讨论过的与Marr-Hildreth算法有关的方法确定模板大小n值，具体方法是：滤波器模板大小n是大于等于6σ的最小奇数，进而提高高斯模板的完美平滑能力。若实际考虑要求一个较小的滤波器模板，则折中方案是较小的n值，但是其平滑效果要差一些。

在**低阈值=50，高阈值=15**0的情况下，我使用了**σ=1**的模板，比较由于模板大小的不同**（n=3、5、7）**而导致的最后结果的差异：

|  |  |  |  |
| --- | --- | --- | --- |
| 模板大小n | 3 | 5 | 7 |
| 非极大值抑制后幅值图像最大值 | 752 | 662 | 650 |

下面三张图片分别是n=3、5、7的输出结果。

我们发现，以下三张图片都可以较好的发现边缘，但是n=7是提现的任务细节更少，分割出来的边缘更加准确，这是因为高斯滤波让图像更加平滑，幅值图像的各个像素点的值整体降低，在最大阈值不变的情况下，被舍弃的点变多。



当我们保持 **n = 6σ +1 的关系时，**进一步验证不同高斯核对结果的影响：

**σ=1，n=7 σ=2，n=13**





显然，模板变大，导致幅值图像的各个像素点的值整体降低，细节提现更少，同时还会导致很多有限的边缘像素被舍弃，这不是我们希望的。因此，在之后的实验中，我们采**用σ=1，n=7的模板**

1. **高低阈值的选择**

这里我主要进行了两组实验，一个是比较不同高低阈值比的影响，一个是在阈值比确定的情况下，阈值大小对实验结果的影响。

低阈值TL = 50 高阈值TH = 150



低阈值TL = 50 高阈值TH = 100



这里可以看出，低阈值：高阈值 = 1:2 会带来更多的细节，但也有可能是因为高阈值比较低而导致的被保留的像素点较多。

我们在**保持阈值比1:3，非极大值抑制后幅度图像最大值为650**的情况下，测试 **22：65（高阈值=0.1倍最大幅值），32：96（高阈值=0.15倍最大幅值）， 65：195（高阈值=3倍最大幅值）**

**22：65（高阈值=0.1倍最大幅值）**



**32：96（高阈值=0.15倍最大幅值）**



**65：195（高阈值=3倍最大幅值）**



**当高阈值=0.1倍最大幅值时，细节保留太多，当高阈值=0.3倍最大幅值，有效边缘被舍弃太多，高阈值=0.15倍最大幅值时，是一个很好的折中。**

1. **双阈值处理和连接分析**

**在这一步有一个比较麻烦的地方，我们要判断gNL里面的点p是否能够8邻接到gNH，不能简单地判断p周围八个点是否有gNH里面的点，这样会导致边缘的缺失。**

**我采用了广度优先遍历（BFS）的算法，看gNL里面的点是否能够和gNH 8邻接，这样仅仅需要将gNL遍历一遍就可以完成，时间复杂度只有O（n）。**

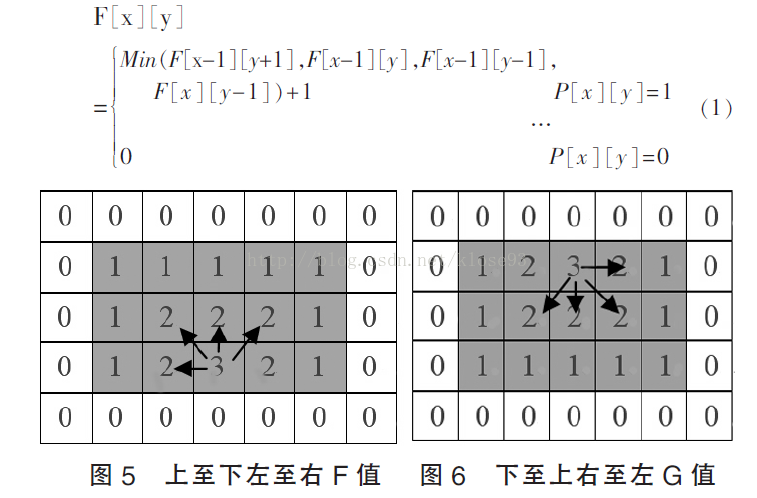
1. **边缘细化算法**

我在canny算法四步结束后，有添加一个边缘细化的步骤，这样可以保证，呈现的边缘仅仅是一个像素值，舍弃多余没用的像素，对于lenna图，这个并不明显，因为在非极大值抑制阶段就已经有了很好的结果，但是对于其他要求更高的情况，比如医学图像处理，边缘宽度只有一个像素是非常重要的一个性质。

**边缘细化算法：**

本算法基于逐层剥取理念，快速细化算法改进了反复扫描的过程，顺序、逆序搜索两次，确定每个点的层数，通过层数可判断该点属于边界点还是骨架， 因而优化算法的效率，大量减少程序运算时间。

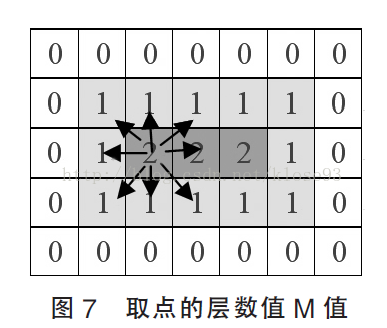
第一步，从上到下，从左到右依次扫描每一个点，若当前点（x，y）为黑点时，需判断其右上、上、左上和左四点的情况。（x-1，y+1） （x-1，y） （x-1，y-1） （x，y-1） 这四点可以理解为当前点是被该四点包围，该点层数即四点层数最小值多一层；如果当前点（x，y）为白色，层数值赋为0。



第二步，自下而上，自右而左，与第一步相似的判断每个点的状态，若当前点是黑点，于是判断其左下、下、右下和右四点的情况（图6）。（x+1，y-1） （x+1，y） （x+1，y+1） （x，y+1）四点包围了当前点，同样取这四点层数的最小值加1 作为该点层数值；如果当前点（x，y）为白色，层数值赋为0。

与此同时，既然已经求得每一个点上包围层数值和下包围层数值，每个点的实际层数其实就是两种层数值中最小值（图7）。

**M［x］［y］=Min(F［x］［y］，G［x］［y］)**



第三步，从上到下，从左到右依次扫描每一个点，此时需要判断该点8 方向（图7）范围内所有点层数的情况。如果当前点的层数值是周围相邻点中最大，该点即保留；如果当前点的层数值并非最大，该点即可删去。