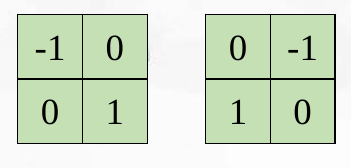
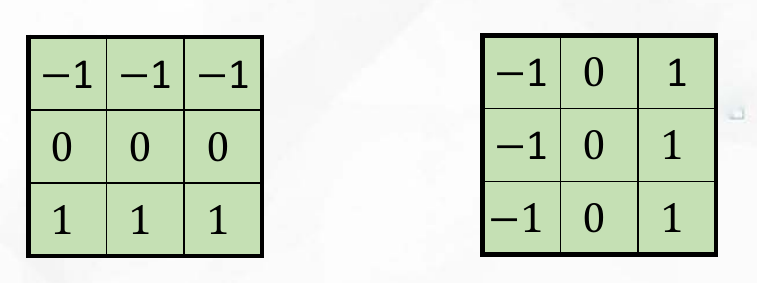
# 第三次作业-王奥博-1120152035

简单解释Roberts, Prewitt, Sobel, lapacian四种算子的编码实现过程,以Roberts为例,对一副图片,只需对图片中的每一个像素点经过对应算子的处理即可.

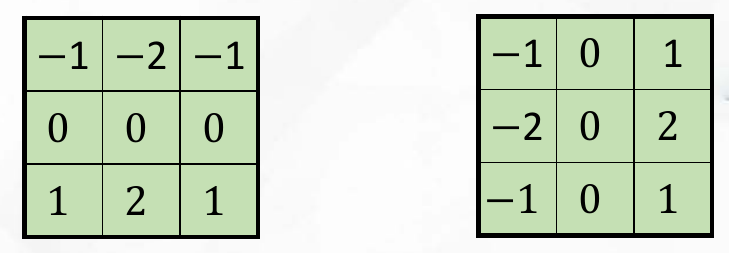
Roberts算子:



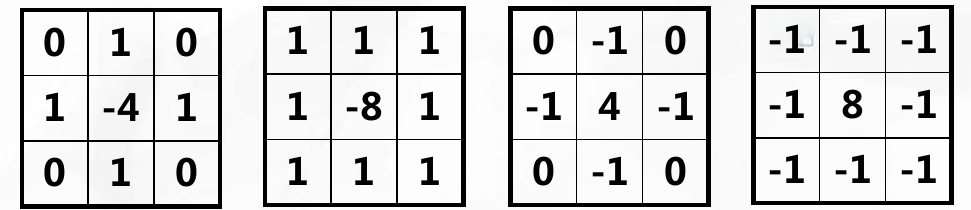
Prewitt算子(两个不同的算子完成了横向和纵向两个方向上的处理):



Sobel算子(同样,Sobel的两个算子分别代表两个方向)



Lapacian算子(lapacian的四个算子代表的两个方向以及黑白反色四种情况):



对代码解释如下:

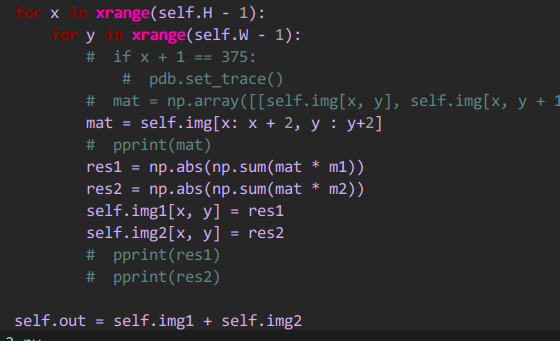
以Roberts算子为例,首先需要把图像转为灰度图:



设置算子矩阵:



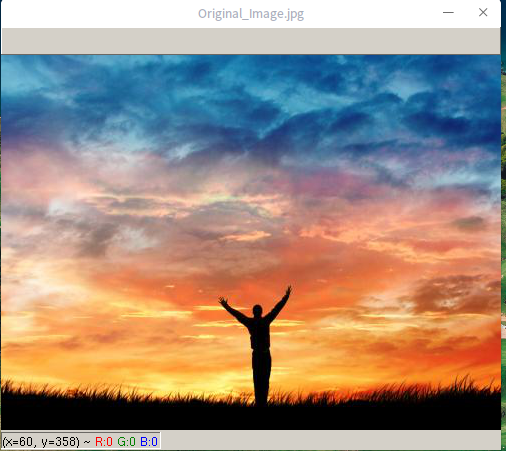
处理图像中的每一个像素点:



完成了Roberts的编程实现,Prewitt算子,Sobel算子,lapacian算子的编程实现与Roberts实现类似,只需简单改变算子矩阵即可,不再复述.

对四种算子测试如下:

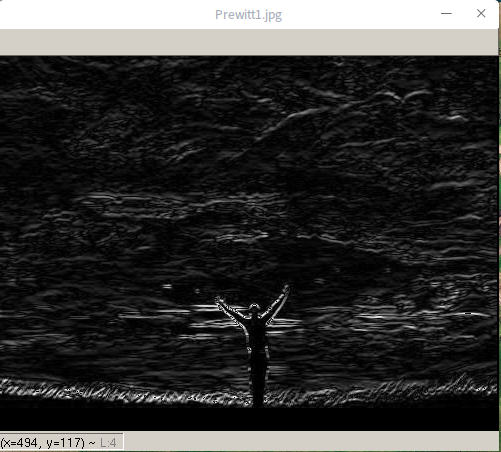
原图:



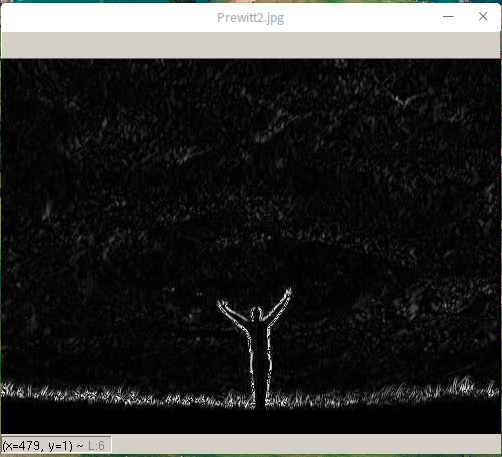
Roberts:



Prewitt横向:



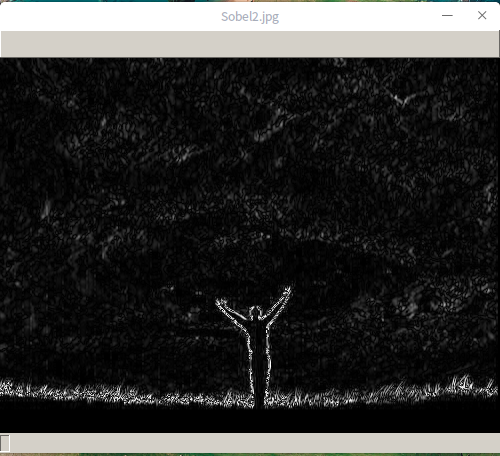
Prewitt纵向:



Sobel横向:

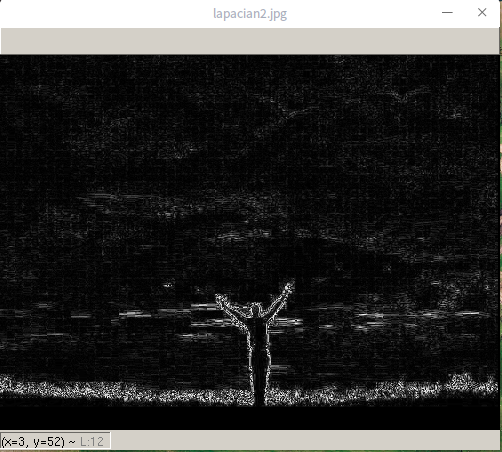


Sobel纵向:

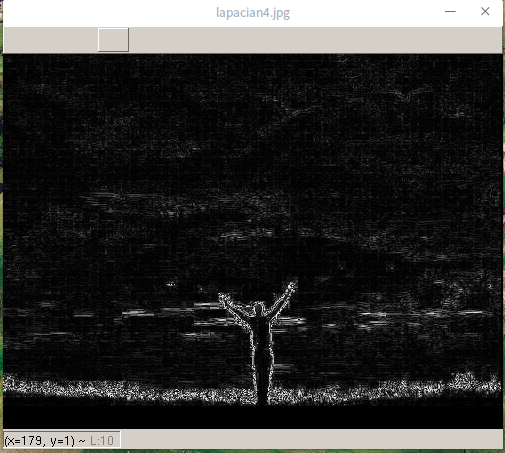


Lapcian:









经过多组测试,可以得出四种算子的特点:

Roberts:

* 优点:
  + Roberts算子对边缘检测较为准确
  + 运算速度快
* 缺点:
  + 当图像中有较多噪点时,处理效果较差
  + 处理后的边界较粗

Prewitt:

* 优点:
  + 与Roberts相比,Prewitt有一定的降噪功能
* 缺点
  + Prewitt降噪是通过像素平均实现的,但这样做使Prewitt对边界检测的准确性不如Roberts

Sobel:

* 优点:
  + 与Prewitt类似,有一定的降噪能力
  + 在不同方向上的检测幅度是一样的
* 缺点:
  + 提取的图像轮廓较粗

Lapcian:

* 优点:
  + 与前几种相比,lapcian的降噪效果最好
* 缺点:
  + 运算较为复杂,运行时间长

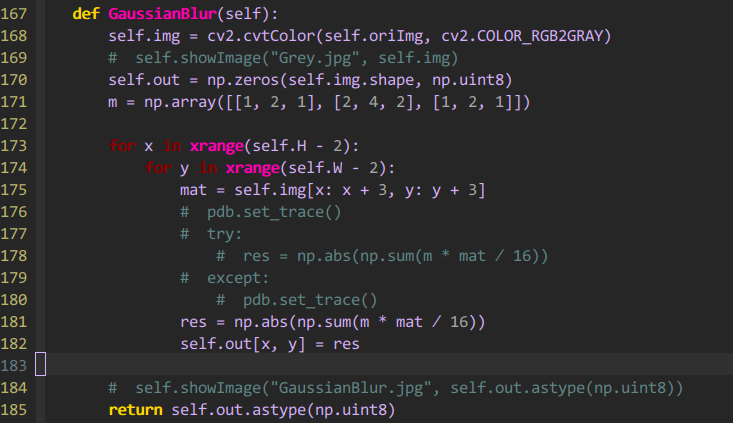
可见,上述几种方法都有明显的缺点,因此引入canny算子,作为一种较为优秀的算法,canny的实现较为复杂,可以分为一下几个步骤:

1. 对图像做高斯模糊处理
2. 用一阶导的有限差分计算梯度的幅值以及方向
3. 对梯度的幅值进行非极大值抑制
4. 用双阈值算法检测和连接边缘

对各个步骤解释如下:

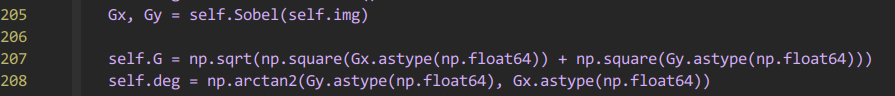
* 高斯模糊:

该步骤主要是为了降噪.通过对原图的灰度图进行高斯模糊后的图像与原图相比有些轻微的模糊,因此单个的噪点几乎全被消除,具体实现如下:



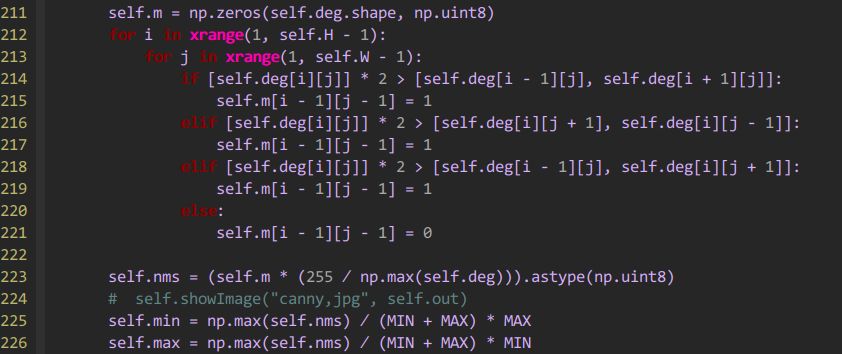
* 计算梯度的幅值与方向:

利用Sobel两个方向上的算子配合numpy的arctan2函数,可以轻松实现计算赋值与角度,具体实现代码如下:



* 对梯度幅值进行非极大性限制:

上述步骤已经获得了全局的梯度,但还没有获得边缘,保留局部梯度最大的点即可或的边缘,因此进行抑制非极大值限制,代码如下:



这里听从了@凌轶群的建议,使用矩阵保存了图片的梯度信息,如此通过一步简单的相乘即可得到结果.

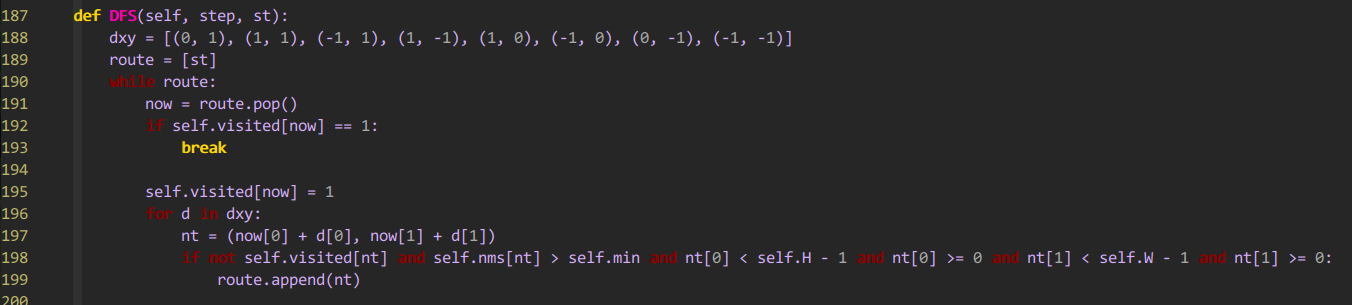
* 用双阈值法检测和连接边缘

双阈法的主要问题是最大阈值与最小阈值的范围较难规定,经过多次实验,得出了当最大阈值:最小阈值在2和3之间是效果较好.在函数的实现过程中,添加了用户控制阈值的接口,只需简单的提供比例参数即可,具体实现如下:





通过DFS搜索保证了结果的准确性



完整代码如下: