林禹臣 5140309507

November 29, 2015（Week 11）

电类工程导论C 实验报告

——Canny 边缘检测算法

目录页

1. 引言

在这次实验中，我主要完成的是自己实现了一遍 Canny 边缘检测算法，加深了对Canny 算法的认识，同时也在实验中了解了如何处理 Canny 算法中的细节问题。

2. 实验环境

Mac OS X 10.11.1 + opencv 2.4.12 + numpy1.10.1 + Python 2.7.10 (64bit)

3. 预备知识

计算梯度的几种卷积算子：

最简单的一种算子是：

$$s_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, s_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

但是更为常用的还是 Sobel 算子：

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Canny 算法的两个核心过程，一个就是对非极大值的梯度强度进行删除，但是这里要注意的一点就是，如果我们真的把所有的非极大值点删除的话，会导致最后的曲线出现非常细的情况，导致有些部分即使采用了双阈值的方法但是还是不能连成线，所以这里要进行宽松处理，之后会提到有两种方式来进行宽松处理。

另外一个过程就是边缘追踪函数，这是一个栈的过程，所以可以利用递归的方法来进行编写代码。首先我们把点分为三类：

第一类是**一定是边缘的点**（大于高阈值的点），

第二类是**可能是边缘的点**（高于低阈值但是低于高阈值的点），

第三类是**一定不是边缘的点**（低于低阈值的点）。

当我们发现第一类点并不能围城一个封闭曲线时，我们需要让一部分第二类点成为第一类点，他们需要满足的条件就是他们紧邻着第一类点。

4. 实验过程

4.1.去除噪音

这里最简单的方法就是利用高斯模糊的方法，高斯模糊的方法可以利用正态分布的特性对周围的邻接点进行加权平均，从而消除了噪声的干扰。

这里利用 OpenCV 自带的方法即可，非常方便。

```
img = cv2.GaussianBlur(img,(3,3),0)
```

第一个参数是需要处理的像素矩阵，第二个参数是所要研究的边界，最后一个是 sigma标准差，所以这里如果 sigma=0，指的是一种均匀的分布情况。也可以取1.5作为标准差。

4.2.梯度相关量计算

梯度的计算因算子不同而有很多差异。比如我一开始选择的算子是最简单的那个算子，所以我计算的过程，首先构建算子矩阵，然后调用 filter2D 函数来进行卷积操作，这时一定要注意的是要指定 anchor 也就是锚点，否则计算肯

定是错的，而且还要注意到除以2的操作，这个操作并不在算子之中，需要自己处理。代码如下：

```
sx = np.array([[-1,1],[-1,1]])
sy = np.array([[1,1],[-1,-1]])
P = cv2.filter2D(img,cv2.CV_32F,sx,anchor=(0,0))/2
Q = cv2.filter2D(img,cv2.CV_32F,sy,anchor=(0,0))/2
M = np.sqrt(P*P+Q*Q)
```

这里 sx 和 sy 是两个算子矩阵，P 是套用了 sx 的卷积值，Q 是套用了 sy 的卷积值。最后我们要算幅值可以让

$$M[i,j]=\sqrt{P[i,j]^2+Q[i,j]^2}$$

但是我最后采用的是 Sobel 算子，所以采用了另外一种计算方式：

首先利用 cv2.sobel 函数计算 dx，dy，然后把他们转回 int8，这样自动避免了负数和超过255的数的问题。然后我们可以把 P 和 Q 的模做一个算数平均数，近似的来计算幅值（因为我们对具体的幅值大小精度并不关心，只要满足单调性即可。）所以此时，

$$M(x,y)=\frac{1}{2}|P(x,y)|+\frac{1}{2}|Q(x,y)|$$

所以我们可以利用加权和函数来处理这个过程。代码如下：

```
#Sobel operator
dx = cv2.Sobel(img,cv2.CV_16S,1,0)
dy = cv2.Sobel(img,cv2.CV_16S,0,1)
P = cv2.convertScaleAbs(dx)   # convert 2 uint8
Q = cv2.convertScaleAbs(dy)
M = cv2.addWeighted(P,0.5,Q,0.5,0)
```

此时我们已经得到了 P、Q、M，还差一个就是梯度方向的角度Sita 矩阵。我们可以利用 arctan 函数来计算。

```
Sita = np.zeros(img.shape)

for x in range(1,img.shape[0]-1):
    for y in range(1,img.shape[1]-1):
        if(P[x][y] == 0):
            Sita[x][y] = np.sign(Q[x][y]) * PI/2
        else:
            Sita[x][y] = math.atan(Q[x][y]/P[x][y])
```

目前位置，我们完成了所有 Canny 算法的准备工作，下面开始第一个核心过程。

4.3.非极大值抑制的处理

4.3.1.GetPoint(C, sita)函数

这个函数根据给定的中心点C，和一个梯度方向 Sita，来判断此梯度线与九宫格的交点位置。此函数只返回在第二、三象限的交点，另外一个交点可以由对称性得到。代码简单但是比较长，主要是分类讨论，这里不贴了，主要是几何知识，利用对称性等条件即可，注意边界的情况处理。

4.3.2.InsertValue(P, M)函数

这个函数用来插值。因为我们的交点非常有可能不是格点，所以我们需要对这个点进行插值，插值的过程其实就是取离这个点最近的两个点的幅值的加权平均数，权重就是距离的比值。代码如下：

```python
#P is Point , M is Matrix
def isInteger(num):
    return math.trunc(num)==num
def InsertValue(P,M):
    x,y = P
    if(isInteger(x) and isInteger(y)):
        return M[x][y]
    elif(isInteger(x) and not isInteger(y)):
        y1 = math.trunc(y)
        y2 = y1+1
        dy = y-y1
        return M[x][y1]*dy + M[x][y2]*(1-dy)
    else:
        x1 = math.trunc(x)
        x2 = x1+1
        dx = x-x1
        return M[x1][y]*dx + M[x2][y]*(1-dx)
```

主要分三类讨论。

4.3.3.极大值判断 & 抑制极大值（放宽条件）

这个是核心过程中的核心过程。首先要注意到的是，我们必须新创建一个矩阵N，用来存储中间过程，并且最后生成边缘图像。如果直接在 M 上处理，会导致每次改变 M 会对之后的幅值判断造成影响，形成双线现象。（王伟涛同学在群里提问时的那个问题就是这个原因）。

对于每一个像素点来说，如果本身这个点的幅值就是0，那么直接跳过不处理，因为梯度为0的点永远不会是边界点，对应的 N 中元素也是0，减少了大量运算。

如果不是0，我们首先利用 GetPoint 函数来取出它所在梯度线与九宫格的其中一个交点，dtmp1，然后利用对称性得到 dtmp2。

接下来判断 (x,y) 是不是 dtmp1,dtmp2中的极大值就可以了，如果是，则它可能将来是边缘点，所以把它在N 中对应的位置先暂时置为110（110只是一个flag，取任何一个不是0或255的数都可以。）

否则，让 N中的对应位置仍然是0。

但是，正如之前所说，我们这里需要放宽条件，否则会出现曲线太细，曲线不完整等等问题。那么我们在比较时，比较的是 M[x][y]和插值的0.85倍即可，也就是我们放宽了15%的条件，让曲线更加粗壮。

代码如下：

```
#surpress the non-extreme-value
for x in range(1,img.shape[0]-1):
    for y in range(1,img.shape[1]-1):
        if(M[x][y]==0):
            N[x][y]=0
            continue
        dtmp1 = GetPoint((x,y),Sita[x][y])
        dtmp2 = (2*x-dtmp1[0],2*y-dtmp1[1])
        if((M[x][y] < 0.85*InsertValue(dtmp1,M)) or (M[x][y] < 0.85*InsertValue(dtmp2,M))):
            N[x][y] = 0
        else:
            N[x][y] = 110 # (x,y) is a possible edge point
```

4.4.阈值自动生成

我们确实可以手动指定高低两个阈值，但是如果利用比例来自动确定，我们就不用关心具体的幅值大小了，而且这个比例一般来说是固定的在0.79左右，二级阈值是一级阈值的一半或者三分之位置。

为了实现这个，我们首先要计算灰度梯度强度直方图，从而确定到底有多少个可能是边缘的点，然后根据这个数值乘以高阈值比例算出具体的阈值，以

此算出低阈值数值。具体过程很简单，只是步骤比较多而已。代码就不在这里贴了。

4.5.边缘跟踪迭代

这里是第二个核心过程。终于要进行对边缘的构建了。这里有也有两个步骤，第一步，对每个高于高阈值的点，把 N 中对应位置设为255，并且以此为起点，进行跟踪迭代，让他周围的每个高于低阈值的点都不是孤立点，这个过程可以保证最后形成的边缘是一个闭合曲线。

代码如下：

```
for x in range(img.shape[0]):
    for y in range(img.shape[1]):
        if((N[x][y]==110) and (M[x][y]>=highThresold)):
            N[x][y] = 255 #white
            Trace(x,y,lowThresold,N,M)
```

其中 Trace 为：

```
def Trace(ori_x,ori_y,lowThresold,res,MM):
    dx = [1,1,0,-1,-1,-1,0,1]
    dy = [0,1,1,1,0,-1,-1,-1]
    for i in range(8):
        x = ori_x+dx[i]
        y = ori_y+dy[i]
        if(res[x][y]==110 and MM[x][y]>=lowThresold):
            res[x][y]=255
            Trace(x,y,lowThresold,res,MM)
```

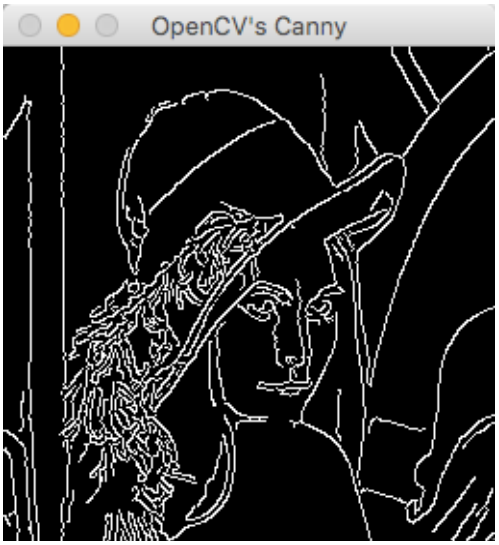
第二步，就是把所有没有用到的『可能是边缘的点』删除。即把此时仍然在 N 中值为110的点置为0。代码如下：

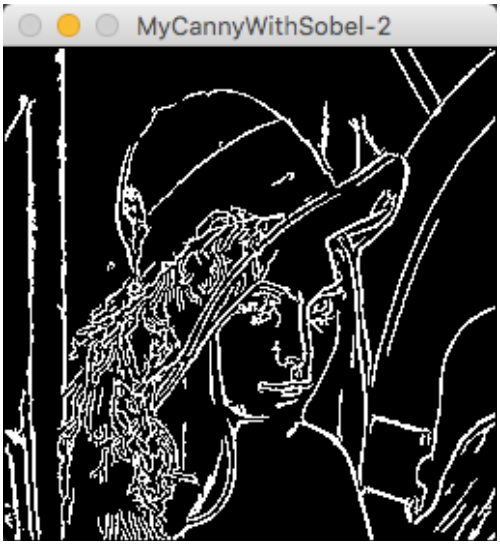
```
#clear 110
for x in range(img.shape[0]):
    for y in range(img.shape[1]):
        if(N[x][y]!=255):
            N[x][y] = 0 #black
```

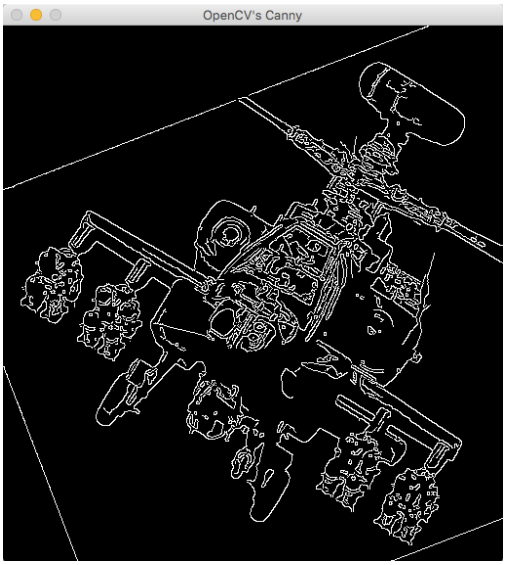
此时 N 即是最终的边缘曲线图片，输出即可。

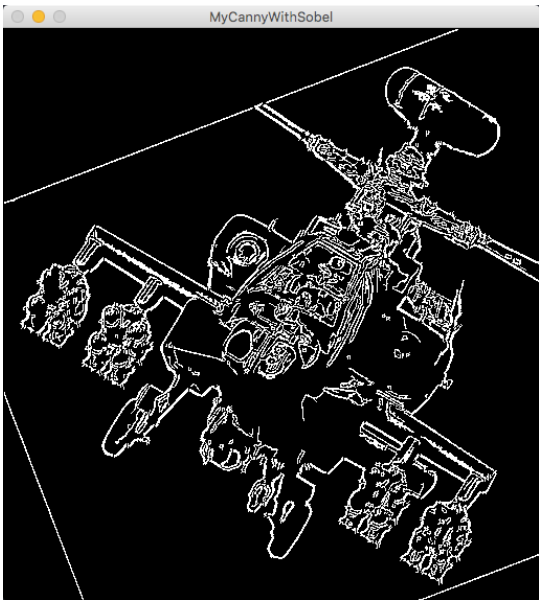
5. 结果展示

从左至右依次为，原图、OpenCV 自带函数图、我自己的代码实现的图。

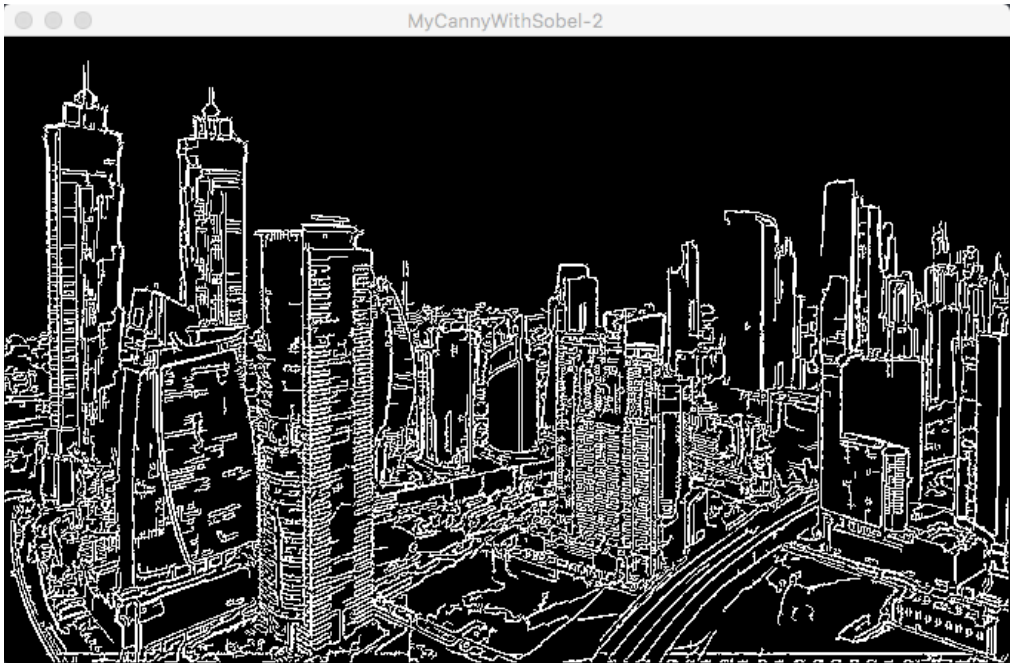




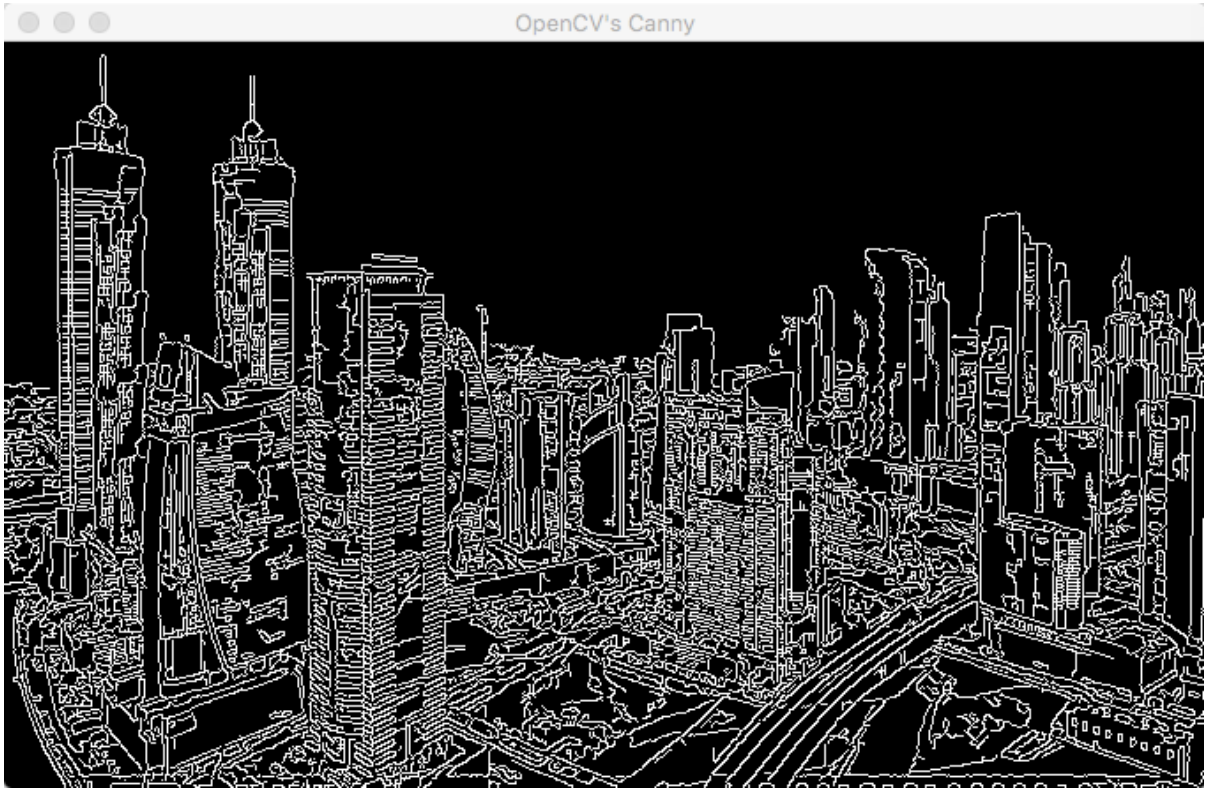




OpenCV 实现的



我的代码实现的：



6. 遇到的困难和总结

遇到的最大困难就是在使用 Sobel 算子时发现很多边缘线非常非常细，甚至很多点是孤立的，一开始以为是 Sobel 算子本身的问题。所以猜测OpenCV的官方 canny 函数中肯定不是 Sobel 算子，但是查源码一看，确实是 Sobel 算子。那是怎么回事呢?

仔细阅读源码发现，它在非极值抑制的那一步并没有采用 PPT 中的算法，算插值。他把角度近似分类成了几个固定的角，比如 45度、90度、135度等等，这样肯定对应格点，可以直接比较。这样的操作有两个好处，第一个好处是加快了运算速率，避免了找点、插值等等步骤，只需要对角度进行分布估计即可。第二个好处，它使得『非极值』这个条件被放宽了，所以会有更多的点参与到边缘点的待选点中来，从而使得线条更粗，更加闭合、美观。

根据这个思想，我采取了另一种放宽条件的做法，那就是判断时比较的是0.85倍的插值和中心点，从而放宽了条件。

另一个小困难就是递归层数可能超过最大限制，要进行调节:

```
import sys
sys.setrecursionlimit(1000000)
```

7. 参考

http://blog.csdn.net/sunny2038/article/details/9170013

http://blog.csdn.net/likezhaobin/article/details/6892629

http://blog.csdn.net/abc20002929/article/details/37833849

非常感谢助教的及时帮助和何老师的指导。

林禹臣 5140309507