浙江大学

计算机视觉(本科)作业报告

作业名称： Candy Edge Detection

姓　　名：

学　　号：

电子邮箱：

联系电话：

导　　师：

2019 年 11 月 30 日

Candy Edge Detection

一、 作业已实现的功能简述及运行简要说明

1.功能简述

本次实验的要求是对输入的一张彩色图片实现 Canny Edge 检测算法。通过①将图片转换为灰度图像，②用高斯滤波器平滑图像，③用一阶偏导有限差分计算梯度幅值和方向，④对梯度幅值进行非极大值抑制（NMS），⑤用双阈值算法检测和连接，⑥最终边缘结果图（作为 mask）覆盖在原彩色图像这六个步骤提取出图像的彩色边缘图。

其中，在本实验过程中，没有调用 OPENCV 或其他 SDK 里于 Canny 边缘检测相关的函数；图像梯度计算、非极大值抑制、双阈值三个关键步骤都由自己写函数实现。对双阈值变化的不同效果的实验在实验结果中显示。

在本次实验中，最后可以得到中间的处理结果及最终的检测结果，输出成图像文件，包括梯度图、Non-max Supression(NMS)后的图，双阈值结果图和彩色边缘图。

2.运行简要说明：

打开 Pycharm 运行程序，可以在运行程序过程中显示视频，并在程序文件夹下生成 8 个 BMP 文件，分别为梯度图、Non-max Supression(NMS)后的图，双阈值结果图（三组）和彩色边缘图（三组）。其中三组图片可以展现双阈值变化的不同效果。

二、 作业的开发与运行环境

Windows 10

Python 3.7.4

Opencv 4.1.1

三、 系统或算法的基本思路、原理、及流程或步骤等

程序分为以下六个步骤：

1. 将图片转换为灰度图像

读入图像，得到原始图像矩阵，将图像转换为灰度图像，并记录图像大小。

2.用高斯滤波器平滑图像

3.计算梯度幅值和方向

运用 Sobel 算子计算梯度幅值和方向角（自己实现算法）。

4.对梯度幅值进行非极大值抑制（NMS）

方向角离散化，抑制，得到新幅值图。

5.用双阈值算法检测和连接

取高低两个阈值作用于新幅值图，得到高阈值边缘图和低阈值边缘图，连接高阈值边缘图，出现断点时，在低阈值边缘图的 8 邻点域搜寻边缘点连接。

6. 提取图像的彩色边缘图。

最终边缘结果图（作为 mask）覆盖在原彩色图像这六个步骤提取出图像的彩色边缘图。

四、 具体如何实现，例如关键（伪）代码、主要用到函数与算法等

1. 将图片转换为灰度图像

（1）首先读入图像，得到原始图像矩阵，这里运用了 cv2. imread(filename, flags)函数，其中 filename 为导入图片的路径文件名。

（2）将图像转换为灰度图像，这里运用了 cv2.cvtColor(src, code, dst, dstCn)函数，其中 src 为需要转换的图像，code 为颜色映射类型，在程序中我运用到的实例为：cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)。

（3）记录图像大小，即用 h,w 来代表图像大小。

2. 用高斯滤波器平滑图像

这里用到了 cv2.GaussianBlur(src, ksize, sigmaX, dst=None, sigmaY=None, borderType=None)函数，其中 src 是需要进行高斯平滑的图像，ksize 是高斯核的大小，即（width, height），两者都是正奇数（如果设为 0 的话可以根据 sigma 得到），sigmaX 为 X 方向的高斯核标准差，sigmaY 为 Y 方向的高斯核标准差。在这里我的应用实例为 cv2.GaussianBlur(gray, (3, 3), 0)。

3. 计算梯度幅值和方向

这里我采用 Sobel 算子计算梯度幅值和方向角，Sobel 算子具体结构如下：

$$kernal\_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad kernal\_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

（1）使用一阶优先差分计算偏导数阵列 P 和 Q（Gx 和 Gy）。其中 Gx 和 Gy 的计算方式即为用上面的 kernal_x 和 kernal_y 矩阵分别对灰度图像进行卷积，得到的结果即为 Gx 和 Gy，其中对应的代码如下：

```
# 运用 sobel 算子计算梯度幅值和方向角
Gx = gray.copy()
Gy = gray.copy()
kernal_x = np.array(([-1, 0, 1], [-2, 0, 2], [-1, 0, 1]), dtype='float64')
kernal_y = np.array(([1, 2, 1], [0, 0, 0], [-1, -2, -1]), dtype='float64')
```

```
for i in range(1,h-1):
    for j in range(1,w-1):
        Gx[i,j] = np.sum(kernal_x * gray[i - 1:i + 2, j - 1: j + 2])
        Gy[i,j] = np.sum(kernal_y * gray[i - 1:i + 2, j - 1:j + 2])
```

（2）计算梯度幅值

$$M[i,j]=\sqrt{G_x[i,j]^2+G_y[i,j]^2}$$

如上图所示，梯度幅值为刚刚所计算出来的 Gx 和 Gy 每个对应元素的平方和开根号，这个公式用代码表示出来，如下所示：

```
# 梯度幅值
M = np.sqrt(Gx ** 2 + Gy ** 2)
```

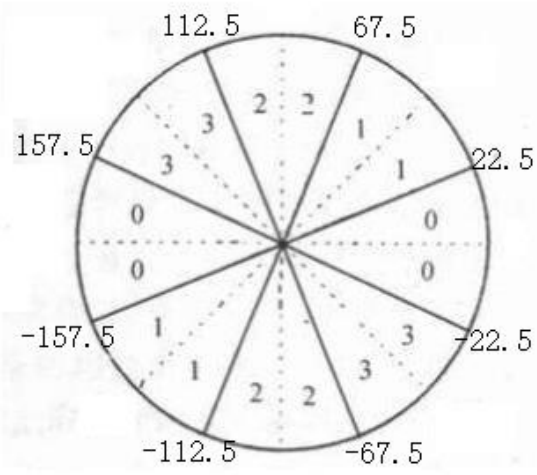
（3）计算方向角

这里采用 numpy.arctan2(x1, x2, *args, **kwargs)函数，在这里我们给定 x1，x2（在实例中即为 Gy，Gx）,会返回以弧度为单位的矩阵，其每个值的范围为[-Π,Π]。

应用实例为 theta = np.arctan2(Gy, Gx)。

4.对梯度幅值进行非极大值抑制（NMS）

（1）首先，进行方向角离散化。将梯度角离散为圆周的四个扇区之一。离散后的区域如图所示，分别为 0，1，2，3 四个区域。



（2）抑制，得到新幅值图。

对梯度图像中每个像素进行非极大值抑制的算法是：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，如上图所示，我们所比较的正负梯度方向上的两个

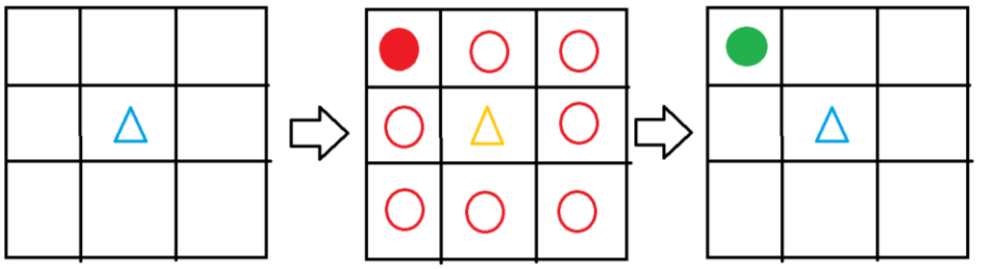
像素相对于当前像素的位置在同一区域内是相同的。如果当前像素的梯度强度与另外两个像素相比最大，则该像素点保留为边缘点，否则该像素点将被抑制。具体的代码示例如下：

```
for i in range(1, M.shape[0] - 1):
    for j in range(1, M.shape[1] - 1):
        angle[i, j] = round(math.degrees(theta[i, j]), 1)
        if ((angle[i, j] >= -22.5 and angle[i, j] <= 22.5) or angle[i, j] >= 157.5 or angle[i, j] <= -157.5):  # 0
            pixel1 = M[i, j + 1]
            pixel2 = M[i, j - 1]
        elif ((angle[i, j] > 22.5 and angle[i, j] <= 67.5) or (
                angle[i, j] > -157.5 and angle[i, j] <= -112.5)):  # 1
            pixel1 = M[i + 1, j - 1]
            pixel2 = M[i - 1, j + 1]
        elif ((angle[i, j] > 67.5 and angle[i, j] <= 112.5) or (
                angle[i, j] > -112.5 and angle[i, j] <= -67.5)):  # 2
            pixel1 = M[i + 1, j]
            pixel2 = M[i - 1, j]
        elif ((angle[i, j] > 112.5 and angle[i, j] < 157.5) or (angle[i, j] > -67.5 and angle[i, j] < -22.5)):  # 3
            pixel1 = M[i + 1, j + 1]
            pixel2 = M[i - 1, j - 1]
        if (M[i, j] != max(M[i, j], pixel1, pixel2)):
            N[i, j] = 0
```

5.用双阈值算法检测和连接

（1）取高低两个阈值（T2，T1）作用于新幅值图 N[i,j]，得到高阈值边缘图和低阈值边缘图。其具体做法为：对于高阈值边缘图来说，即像素值大于高阈值 T2 的像素点被赋值为 255，其它像素点都被赋值为 0，由此得到高阈值边缘图；对于低阈值边缘图来说，即像素值大于低阈值 T1 的像素点被赋值为 255，其它像素点都被赋值为 0，由此得到低阈值边缘图。

（2）连接高阈值边缘图，出现断点时，在低阈值边缘图的 8 邻点域搜寻边缘点链接。其具体做法为：从上到下、从左到右扫描高阈值边缘图，当扫描到像素值为 255 的像素点时，找寻这个像素点在低阈值边缘图对应位置的 8 邻点域中是否有像素值为 255 的像素点，如果有，则将高阈值边缘图对应位置像素点赋值为 255。其大致流程如图所示：



高阈值边缘图　　低阈值边缘图
如图，左上角的值为255　　高阈值边缘图
将与左图中像素点对应点的像素值改为255

其具体的代码实现如下：

```
for i in range(1, h - 1):
    for j in range(1, w - 1):
        if high_t[i, j] == 255:
            for x in range(i - 1, i + 2):
                for y in range(j - 1, j + 2):
                    if low_t[x, y] == 255:
                        high_t[x, y] = 255
```

对于双阈值变化的不同效果分析在实验结果分析部分。

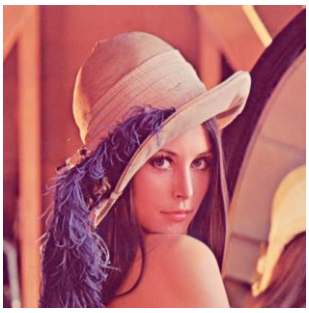
6. 提取图像的彩色边缘图。

最终边缘结果图（作为 mask）覆盖在原彩色图像这六个步骤提取出图像的彩色边缘图。这里我用到了 cv2.add(src1, src2, dst=None, mask=None, dtype=None)函数，这里的 src1 指需要相加的图像 1，src2 指需要相加的图像 2，mask 即掩膜，这里我的实际用法为：final = cv2.add(img, np.zeros(np.shape(img), dtype=np.uint8), mask=mask)，可以实现彩色边缘图像的提取。

五、　实验结果与分析

1. 总体结果分析（仅选取一组双阈值结果图和彩色边缘图）

我所选用的三张图像如下：



（1）lena 图像

梯度图

NMS 之后的图

双阈值结果图

彩色边缘图

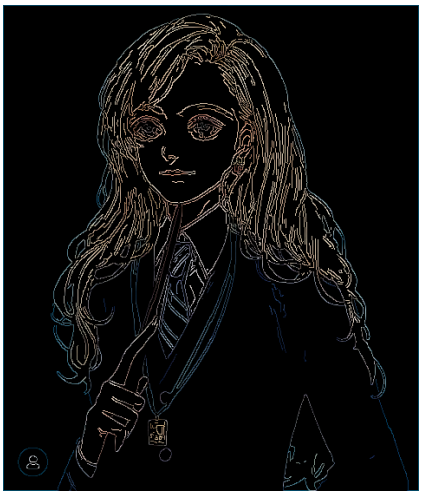
（2）动漫女孩图像

梯度图

NMS 之后的图

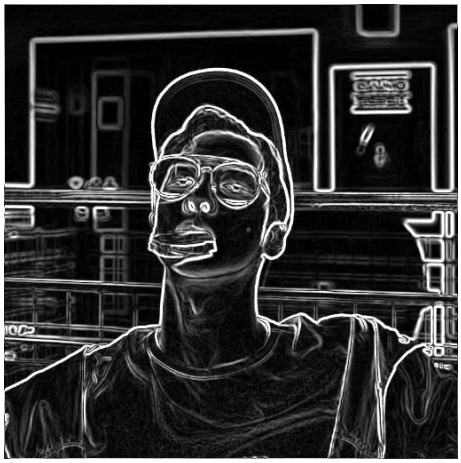
双阈值结果图

彩色边缘图

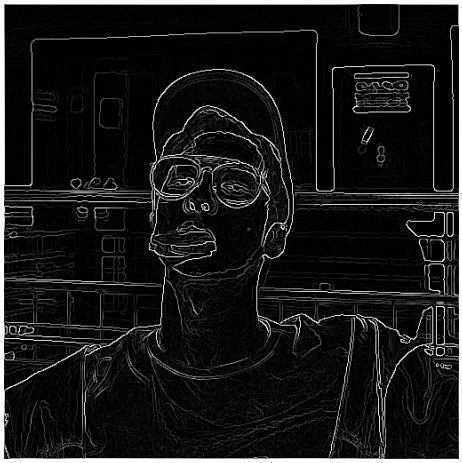


（3）男明星图像

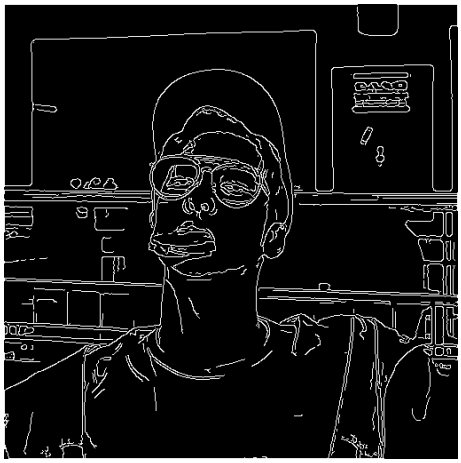
梯度图



NMS 之后的图



双阈值结果图



彩色边缘图

由上面的结果来看，本次实验的边缘检测取得了一些成果，但是和 opencv 自带的 canny 函数相比还是有很大的不足。

2. 对双阈值变化的不同效果进行实验。

这里采取 lena 图像高阈值为 100，低阈值为 50 作为对照组，其双阈值结果图和彩色边缘图如下所示：

双阈值结果图

彩色边缘图

高阈值为 200，低阈值为 50 的双阈值结果图和彩色边缘图如下所示：

双阈值结果图

彩色边缘图

高阈值为 100，低阈值为 20 的双阈值结果图和彩色边缘图如下所示：

双阈值结果图

彩色边缘图

可以看到，当低阈值降低时，会出现一些假边缘；而当高阈值过高时，则会丢失一些必要的边缘信息。由此可见，选择恰当的高阈值和低阈值是非常必要的，如果选择合适，则可以获得较为准确的边缘图。

六、 结论与心得体会

在完成本次实验的过程中主要遇到了两个问题，接下来将主要介绍这两个问题及其解决方案。

1. 第一个问题出现在计算梯度幅值和方向角的地方，用课件中所给出的方法（如图所示）

步2a. 使用一阶有限差分计算偏导数阵列P与Q：

$$G_x[i,j] \approx (S[i,j+1] - S[i,j] + S[i+1,j+1] - S[i+1,j]) / 2$$
$$G_y[i,j] \approx (S[i,j] - S[i+1,j] + S[i,j+1] - S[i+1,j+1]) / 2$$

得出来的结果会比较浅淡，梯度图的边缘不明显（如下图所示），所以最后我选择用 Sobel 算子来来实现这个操作，改用 Sobel 算子之后得出来的结果清晰了许多（可以和上面的进行对比）

2. 第二个问题出现在非极大值抑制的部分，由于刚开始梯度方向出现了一些错误（与正确的梯度方向垂直了），导致 NMS 后出现的图像中有很多的点，而不能连成直线，在这个问题上花费了比较长的时间。如下图所示。后来梯度方向纠正过来之后显示出了正确的图像。

通过本次实验，我了解了 Canny 边缘检测算法，在动手做的过程发现并解决问题，在不断的思考和改正中完成了本次作业，同时巩固了 Python 的语法及运用，感觉收获很大，受益匪浅。

七、 参考文献

灰度图像转换： https://www.jianshu.com/p/fd77d0d4201f

掩膜处理： https://blog.csdn.net/lucky__ing/article/details/78559916