第十周报告

1. 实验概览

本实验旨在理解理解图像特征提取中边缘检测的概念，理解 canny 边缘检测的具体步骤：灰度化图片、高斯滤波、基于不同梯度算子计算灰度值的梯度、实现非极大值抑制、基于双阈值算法检测实现边缘连接。实现 Canny 边缘检测算法，利用不同梯度算子和阈值并比较性能。

2. 实验环境

Docker: sjtumic/ee208

3. 解决思路

a) 灰度图读取

```
img = cv2.imread("./dataset/"+ str(i + 1) + ".jpg", cv2.IMREAD_GRAYSCALE)
```

b) 基于 opencv 实现高斯滤波

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

选择 3 * 3 高斯核进行滤波操作

c) 构建梯度算子

代码中构建有 Sobel 算子和 Prewitt 算子

```
def Sobel(img):
    height = len(img)
    width = len(img[0])
    p = numpy.zeros((height, width), dtype=int)
    q = numpy.zeros((height, width), dtype=int)
    g = numpy.zeros((height, width), dtype=int)
    t = numpy.zeros((height, width), dtype=float)
    for i in range(1, height - 1):
        for j in range(1, width - 1):
            p[i][j] = int(
                (img[i + 1, j + 1] + 2 * img[i + 1, j] + img[i + 1, j - 1])
                - (img[i - 1, j + 1] + 2 * img[i - 1, j] + img[i - 1, j - 1])
            )
            q[i][j] = int(
                (img[i - 1, j - 1] + 2 * img[i, j - 1] + img[i + 1, j - 1])
                - (img[i - 1, j + 1] + 2 * img[i, j + 1] + img[i + 1, j + 1])
            )
            g[i][j] = int(math.sqrt(p[i][j] * p[i][j] + q[i][j] * q[i][j]))
            if p[i][j] != 0:
                t[i][j] = math.atan(q[i][j] / p[i][j])
            else:
                t[i][j] = math.pi / 2
    return p, q, g, t
```

其中返回的 g 为灰度值图，t 为各个点出的梯度方向用以 NMS 使用

d) 实现非极大值抑制（NMS）

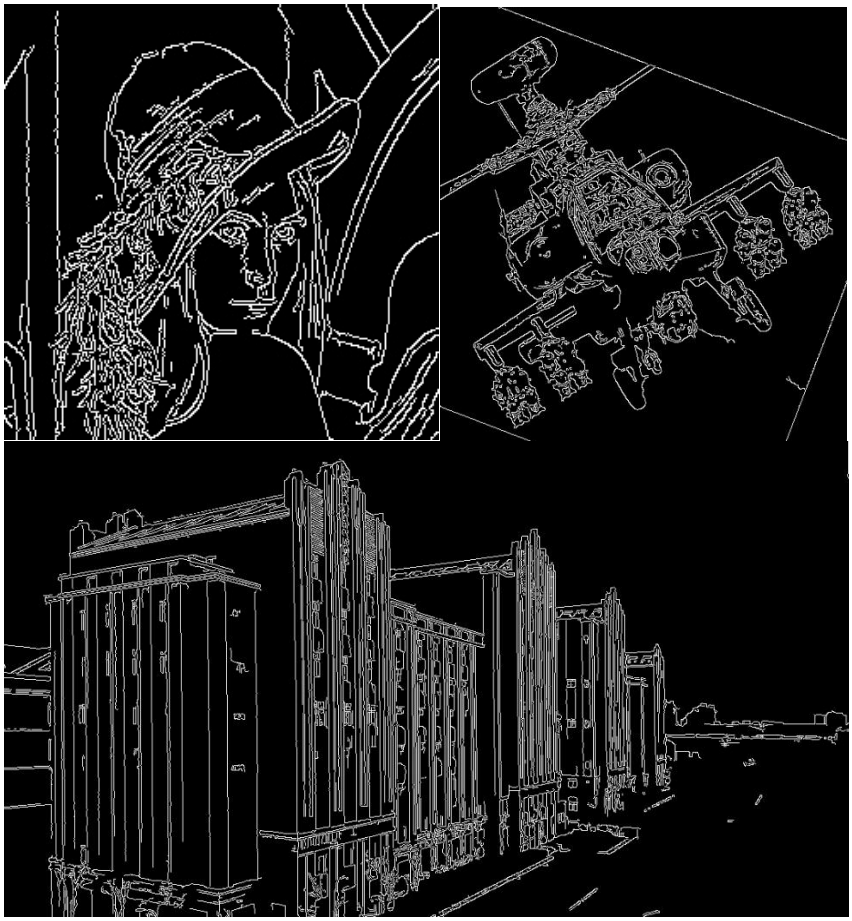
NMS 用于消除冗余边缘点。在具体实现中，根据梯度方向的情况分类， 计算得到 dtmp1 和 dtmp2 两个临时值并和目标像素点进行比较，舍弃不为边缘点的像素。（具体实现见附录代码中 NMS 函数）

e) 双阈值算法检测

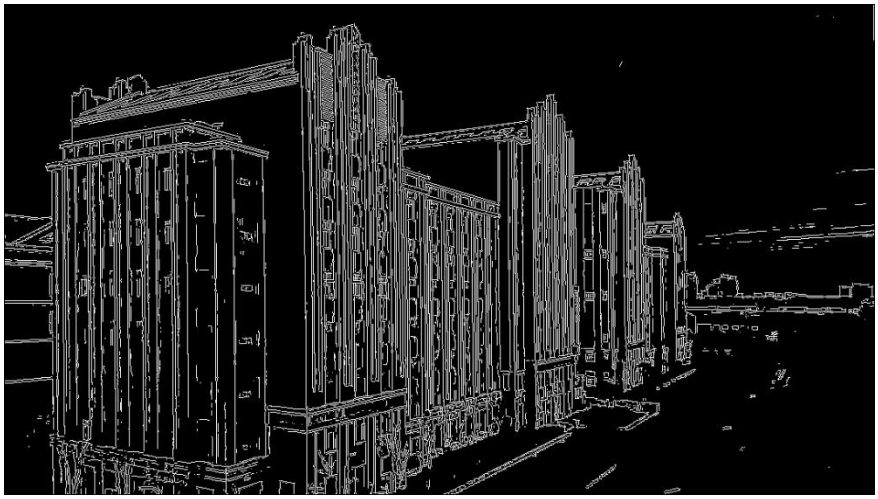
比较图像灰度值和两个阈值，高于 high 参考的直接连接，高于 low 低于 high 的像素点，查找附近是否有确定的边缘，如有则进行连接。（具体算法详见附录 threshhold 函数）

4. 代码运行结果

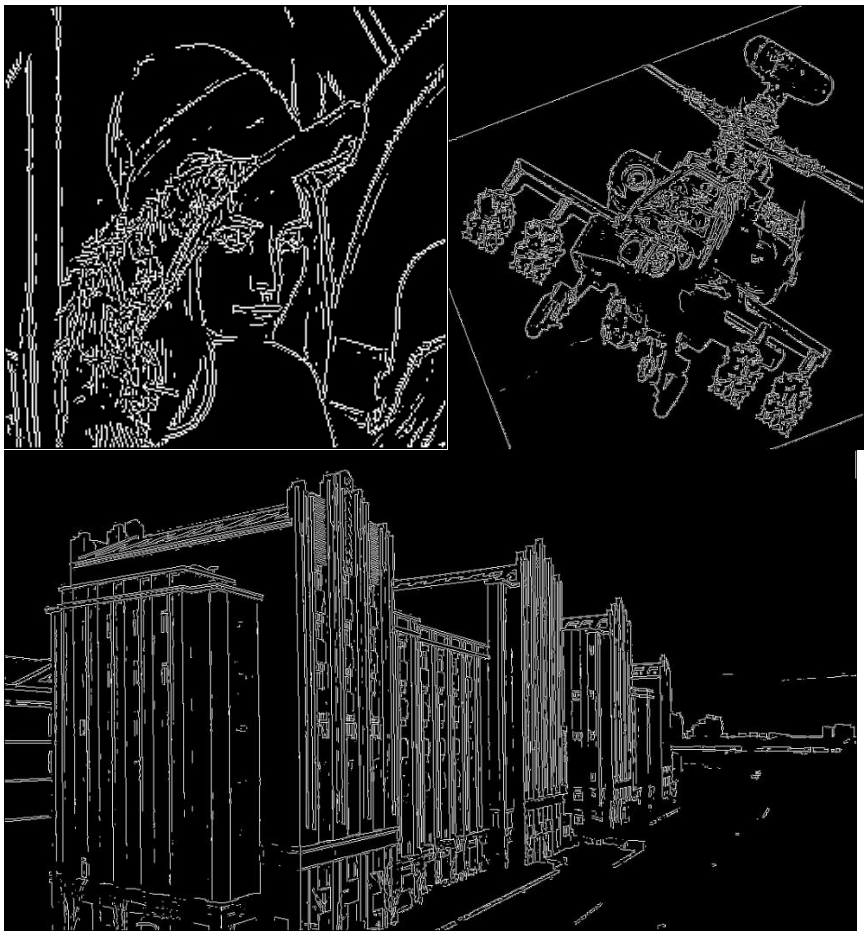
利用 opencv 自带 Canny 函数:



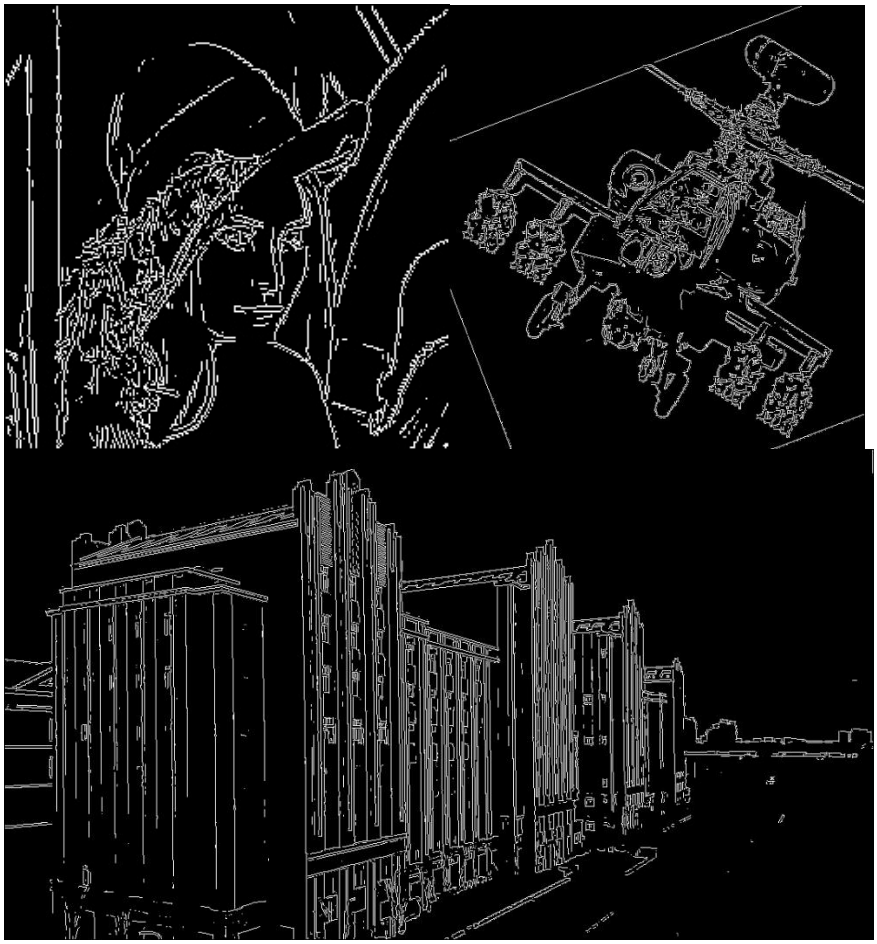
利用 Sobel 算子: （阈值 40, 120）:



Sobel 算子（阈值 64, 160）：



Sobel 算子（阈值 80, 200）：



Prewitt 算子（阈值 40, 120):

5. 分析与思考

- **阈值比较：**当阈值上升时，能够消除更多的假边缘，但同时造成整体检测出的边缘变得更加稀疏，出现边缘不闭合的情况
- **算子比较：**Sobel 算子和 Prewitt 算子基本都基于图像卷积的原理，Prewitt 算子更加倾向于检测垂直和水平边缘，Sobel 算子在处理弧形曲线时有更好的表现

代码详见文件夹中 canny.py