****

《计算机视觉实验报告》

（实验四）

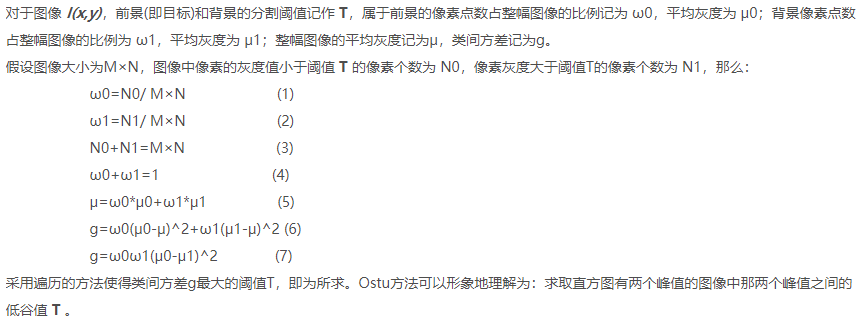
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学院名称 | | ： | | 数据科学与计算机学院 | | | | | | | | | |
| 专业班级 | | ： | | 17级软件工程5班 | | | | | | | | | |
| 学生姓名 | | ： | | 张淇 | | | | | | | | | |
| 学 号 | | ： | | 17343153 | | | | | | | | | |
| 时 间 | | ： | | 2019 | | 年 | 10 | 月 | | 27 | | 日 | |
|  | | |  | |  | | | | 成绩 | | : | |  | |
| 实验二 | | ： | | 图像分割、基本OCR | | | | | | | | | |

# 实验内容

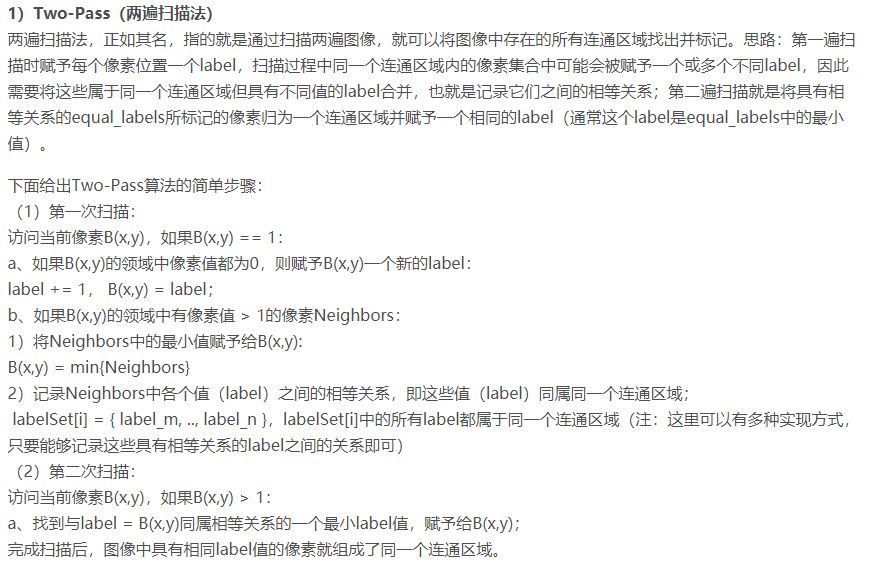
1. 把图像的前后景分割。
2. 把给定图像中的数字切割出来。
3. 把给定图像的标尺OCR。

# 实验原理

1. 大津算法：



1. Two-Pass搜索连通块：



# 实验器材

Ubuntu18.04、g++ 7.4.0

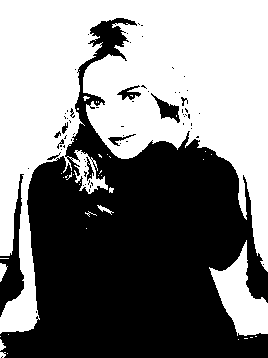
# 实验过程与结果

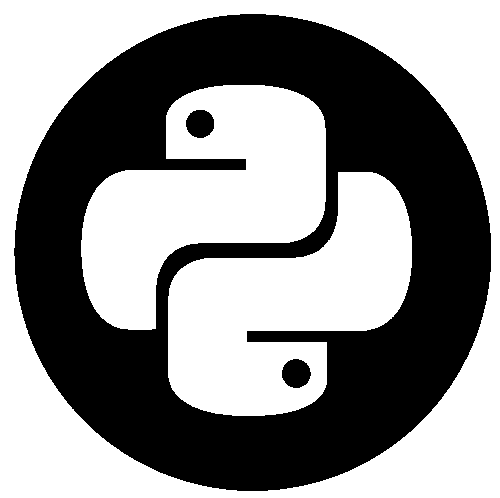
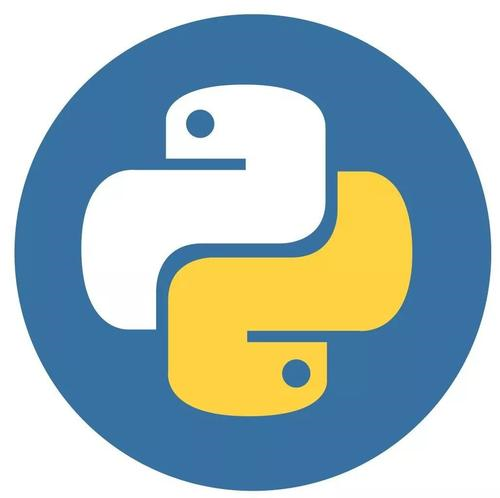
1. 任务一：将图像的前后景分离

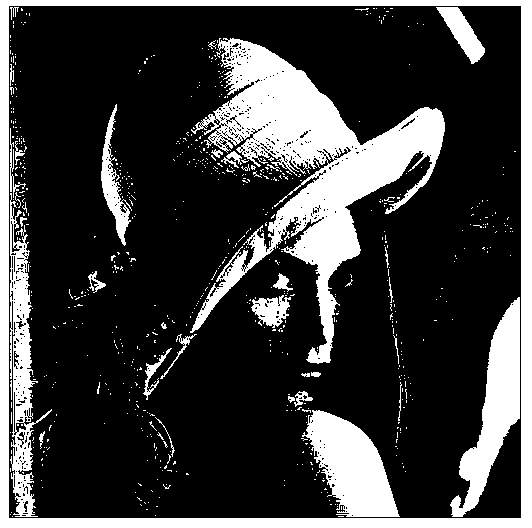
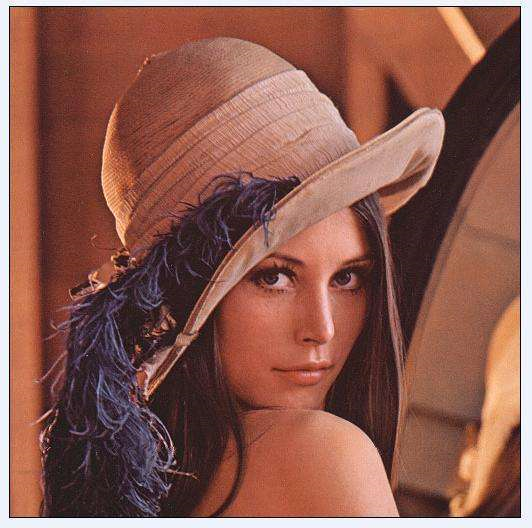
根据OTSU，应该首先将图像转化成灰度图，然后遍历所有的阈值T，求出最大的间类方差，此时对应的T就是前后景分割的阈值。任务一其实只需要计算出相关的系数进行比较即可，没有太多可以介绍的。

不过我对此做了一些小小的优化，在将图片转换成灰度图的时候记录下该图片的灰度的最大值max与最小值min。在对T进行遍历的时候将原来的[0,255]压缩到[min+1,max-1]，减少了遍历次数的同时还避免了x/0的风险。

实验结果如下（原图与结果）：







1. 任务二：把给定图像的数字切割出来

按照作业PDF上提供的思路，这个任务可以分为四步：

1. 对图像进行膨胀(**dilate**)操作：

（作业pdf上给的英文是delate，其实CImg库里面是有dilate这个函数的，但是我用delate没有找到，所以就只能自己写一个了）

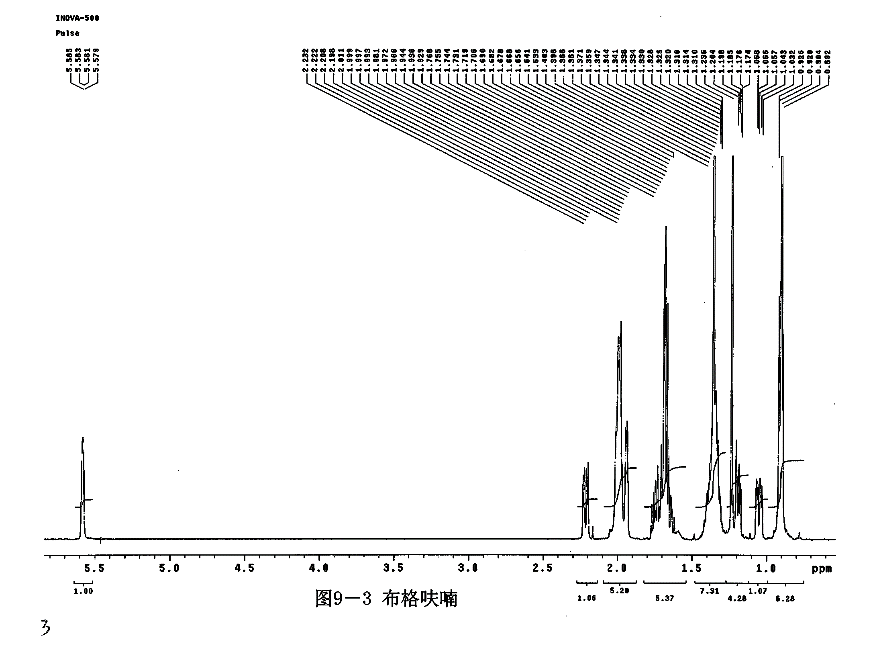
首先将输入的图片转换成二值图像（灰度值的阈值T = 128）

然后遍历整张图，对于每一个像素值为0的块进行四邻域的膨胀，与此同时，还增加了一个判断以避免将两个“不应该相连”的区域连在一起（如果不进行这个判断，那么膨胀后“0”会编程一个实心椭圆…）

程序中对应的函数为：

CImg<uchar> Delate(CImg<uchar> img);

膨胀的结果如下：



再使用Two-Pass的方法求出连通块：其实我实现的方法与上面介绍的有一些区别，大致的伪代码如下：

label\_count = 0

for i in rows:

    for j in cols:

        if img[i][j] == 0: #black

            Run run(-1,i,j,0)   # lable, rows, start\_col, end\_col

            for k in range(j,cols):

                if img[i][k] == 255: #white

                    run.end\_col = k - 1

                    if lable == -1:

                        lable = label\_count

                        label\_count += 1

                    run\_vector.push(run)

                    j = k

                    break

                if img[i-1][j] == 0: # the up neighbor is black

                    find v.label of up label in run\_vector

                    if run.lable == -1: #initial value

                        run.label = v.label

                    elif run.lable != -1 and run.label != v.label: #2 same blocks

                      eq\_set.insert(min(v.label, run.label),max(v.lable, run.label))

for v in run\_vector:    # Replace equivalent label

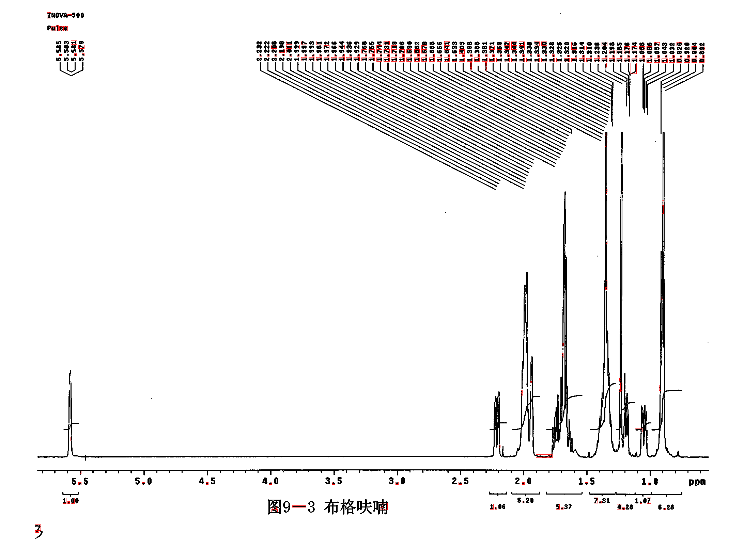
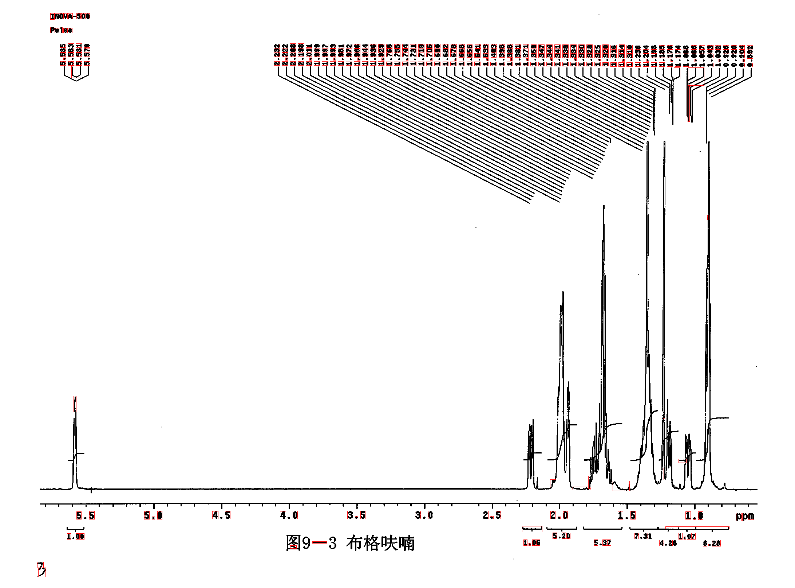
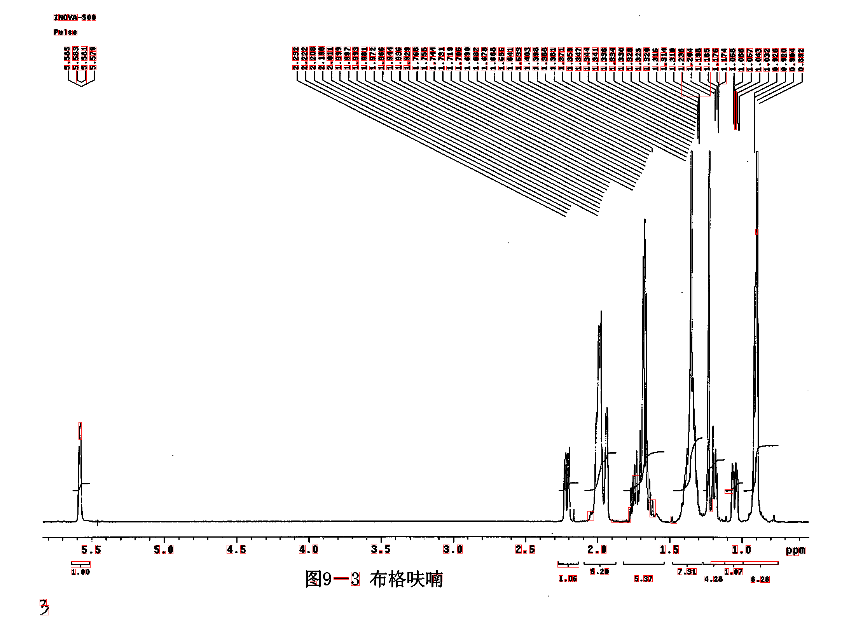
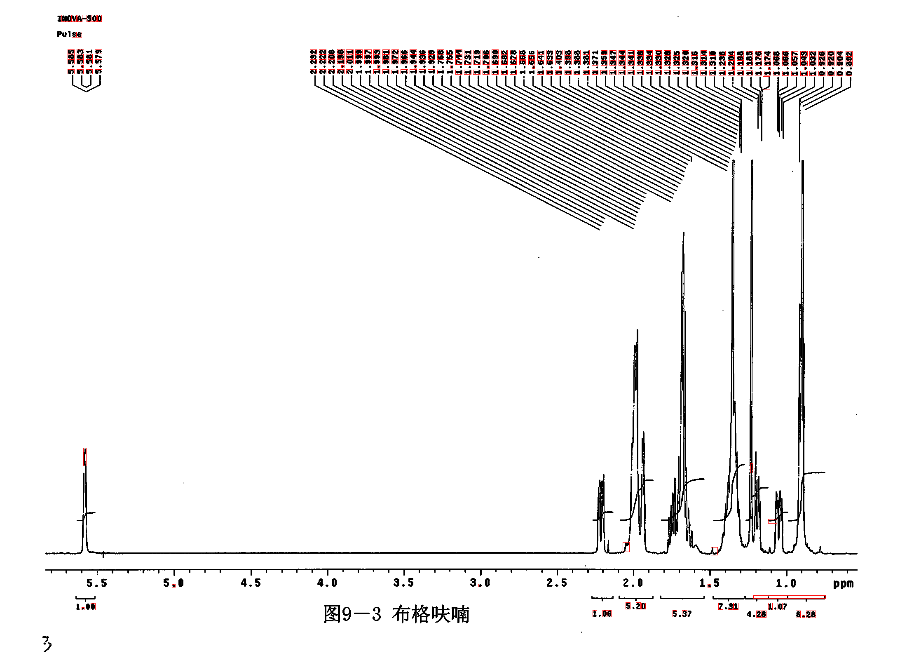
    for s in eq\_set:

        if v.label == s.second:

            v.label == s.first

然后再对上述搜索得到的连通块进行上下阈值的排查，得到长度L满足low < L < high的连通块，求得它们的边界，然后画上红线即可。

实验结果如下：

1. （0，100）
2. （50，250）
3. （50，300）
4. （80，150）
5. 任务三：将给定图像的标尺OCR

按照作业PDF给的思路，首先是需要计算出标尺所在的位置，通过观察可以看到标尺在整张图片中是拥有最多像素值为0的那一行，所以可以对图像进行行遍历，找到标尺所在的那一行line，然后将（line-5,line+45）的行存储在一张新的图像中，程序中对应的函数为：

CImg<uchar> find\_coordinates();

得到的结果如下：



然后使用任务二中的程序对数字进行截取，程序中对应的函数为：

void find\_numbers(CImg<uchar> img, CImg<uchar> &output);

得到的结果如下：



最后再将上述截取的数字与“标准库”中的数字以一比对，找出相似对最高的那一个，认为就是该数字，程序中对应的函数为：

void recognize\_number();

得到的结果为：



可以看到，能够识别出标尺下面的数字。