中山大學
SUN YAT-SEN UNIVERSITY

《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

一、 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。

二、 实验原理

1. 大津算法：

对于图像 *I(x,y)*，前景(即目标)和背景的分割阈值记作 **T**，属于前景的像素点数占整幅图像的比例记为 ω0，平均灰度为 μ0；背景像素点数占整幅图像的比例为 ω1，平均灰度为 μ1；整幅图像的平均灰度记为μ，类间方差记为g。

假设图像大小为M×N，图像中像素的灰度值小于阈值 **T** 的像素个数为 N0，像素灰度大于阈值T的像素个数为 N1，那么：

ω0=N0/ M×N (1)

ω1=N1/ M×N (2)

N0+N1=M×N (3)

ω0+ω1=1 (4)

μ=ω0*μ0+ω1*μ1 (5)

g=ω0(μ0-μ)^2+ω1(μ1-μ)^2 (6)

g=ω0ω1(μ0-μ1)^2 (7)

采用遍历的方法使得类间方差g最大的阈值T，即为所求。Ostu方法可以形象地理解为：求取直方图有两个峰值的图像中那两个峰值之间的低谷值 **T**。

2. Two-Pass搜索连通块：

1）Two-Pass（两遍扫描法）

两遍扫描法，正如其名，指的就是通过扫描两遍图像，就可以将图像中存在的所有连通区域找出并标记。思路：第一遍扫描时赋予每个像素位置一个label，扫描过程中同一个连通区域内的像素集合中可能会被赋予一个或多个不同label，因此需要将这些属于同一个连通区域但具有不同值的label合并，也就是记录它们之间的相等关系；第二遍扫描就是将具有相等关系的equal_labels所标记的像素归为一个连通区域并赋予一个相同的label（通常这个label是equal_labels中的最小值）。

下面给出Two-Pass算法的简单步骤：

（1）第一次扫描：

访问当前像素B(x,y)，如果B(x,y) == 1：

a、如果B(x,y)的领域中像素值都为0，则赋予B(x,y)一个新的label：

label += 1， B(x,y) = label；

b、如果B(x,y)的领域中有像素值 > 1的像素Neighbors：

1）将Neighbors中的最小值赋予给B(x,y)：

B(x,y) = min{Neighbors}

2）记录Neighbors中各个值（label）之间的相等关系，即这些值（label）同属同一个连通区域；

labelSet[i] = { label_m, .., label_n }，labelSet[i]中的所有label都属于同一个连通区域（注：这里可以有多种实现方式，只要能够记录这些具有相等关系的label之间的关系即可）

（2）第二次扫描：

访问当前像素B(x,y)，如果B(x,y) > 1：

a、找到与label = B(x,y)同属相等关系的一个最小label值，赋予给B(x,y)；

完成扫描后，图像中具有相同label值的像素就组成了同一个连通区域。

三、 实验器材

Ubuntu18.04、g++ 7.4.0

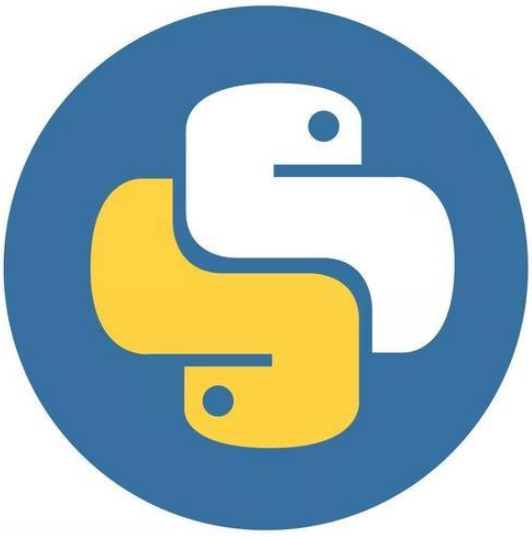
四、 实验过程与结果

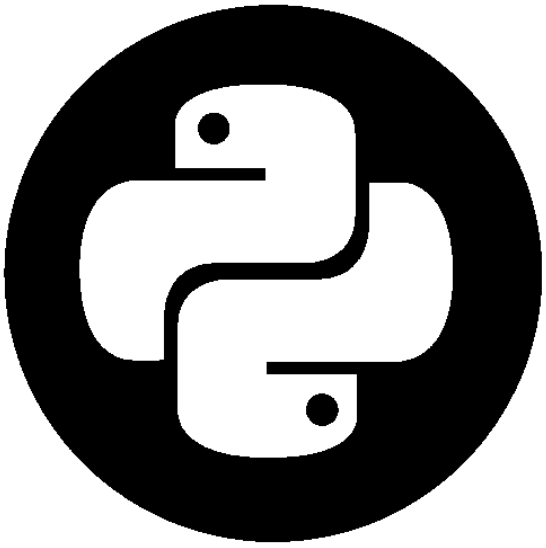
(一) 任务一：将图像的前后景分离

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）：





(二) 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：

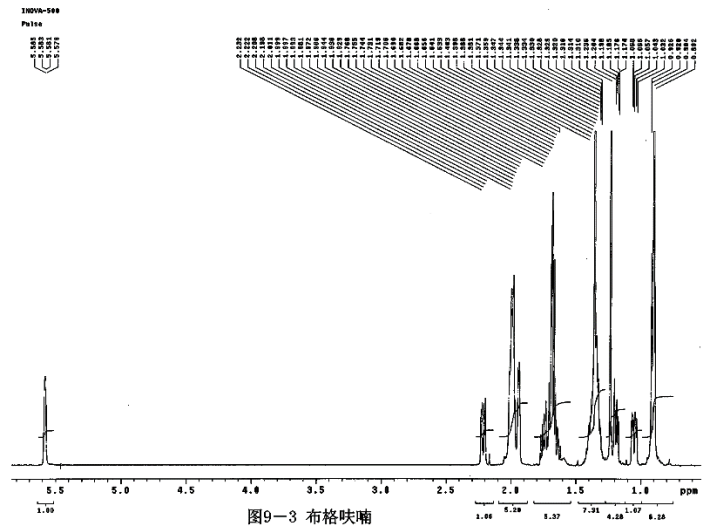


图9—3 布格呋喃

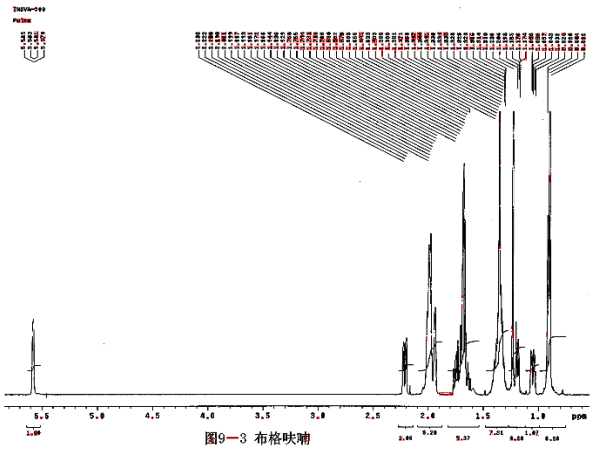
再使用 Two-Pass 的方法求出连通块：其实我实现的方法与上面介绍的有一些区别，大致的伪代码如下：

```
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
```

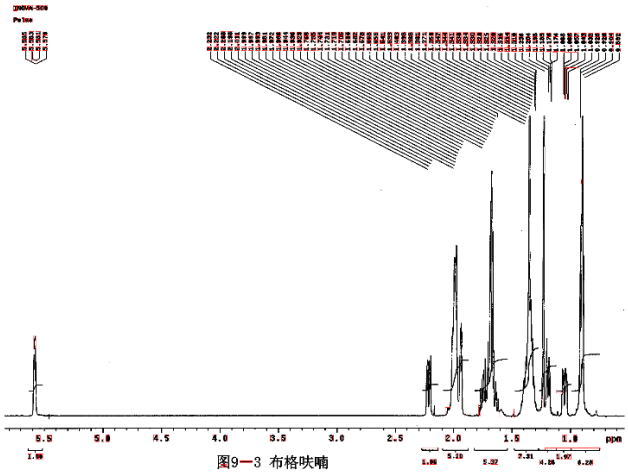
然后再对上述搜索得到的连通块进行上下阈值的排查，得到长度 L 满足 low < L < high 的连通块，求得它们的边界，然后画上红线即可。

实验结果如下：

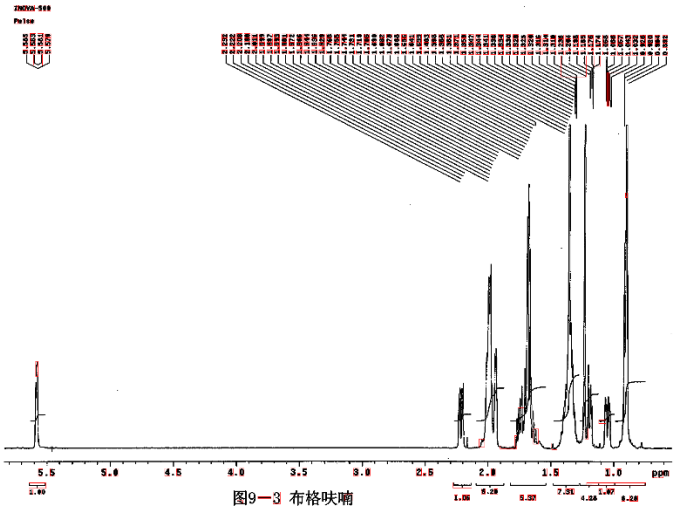
1) （0，100）



2) （50，250）



3) （50，300）



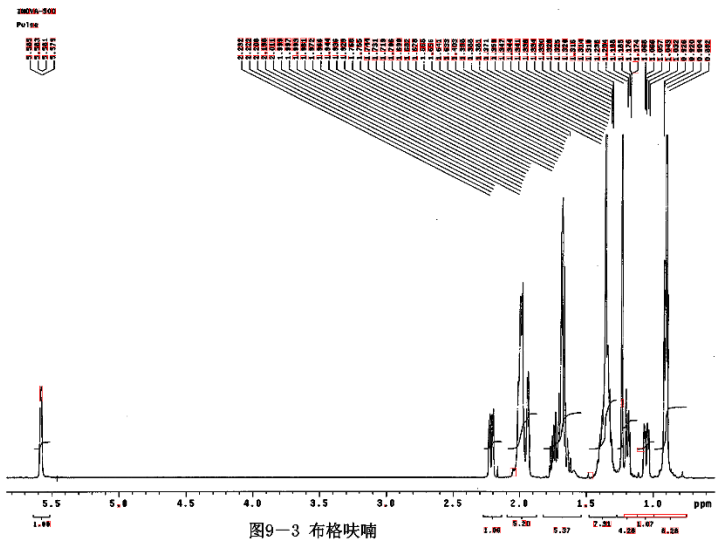


图9—3 布格呋喃

4) （80，150）

(三) 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：

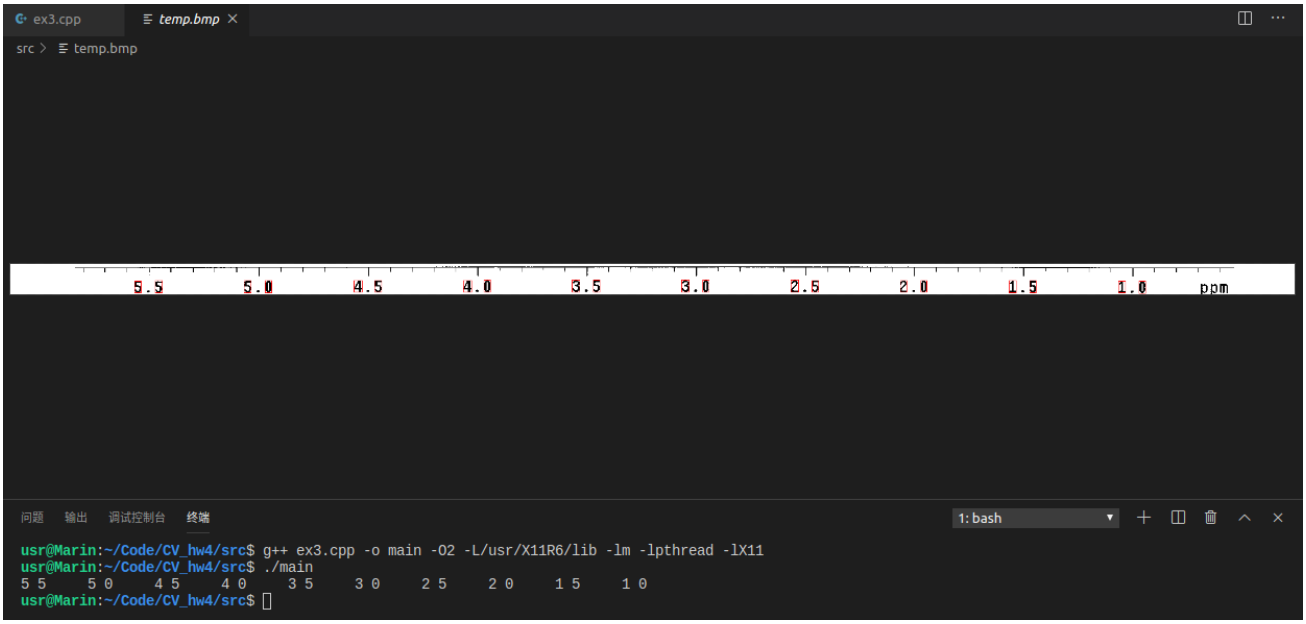
```
void find_numbers(CImg<uchar> img, CImg<uchar> &output);
```

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。