计算机视觉实验报告

学号：
姓名：

实验一 编程环境与图像读写

一、实验题目：

掌握实验环境的搭建，能实现对图像的读写。

二、实验意义：

实验环境是做实验的基础，本实验让学生们通过自己上网搜索资料学习搭建计算机视觉课程所需要的环境搭建。

三、实验目的

1、了解计算机视觉课程的实验环境。本课程实验的实验环境选择的是VS2013+OpenCV 3，版本自己选择

2、掌握 OpenCV 在 VS 中的环境配置方法；

3、学会使用 OpenCV 在 VS 中读入图像并显示图像的方法；

4、能够对读取的彩色图像进行灰度化处理，并且能够获取灰度图像中每一个像素点的值。

四、实验内容

1、安装 VS 2013 和 OpenCV 3 软件；

2、配置 OpenCV：

(1)配置环境变量；

(2)创建 VS 一个控制项目工程，对工程目录进行配置。

3、读取给定图像、灰度化并显示，将灰度图像中的每一个像素的值读出存放在一个文本文件中。

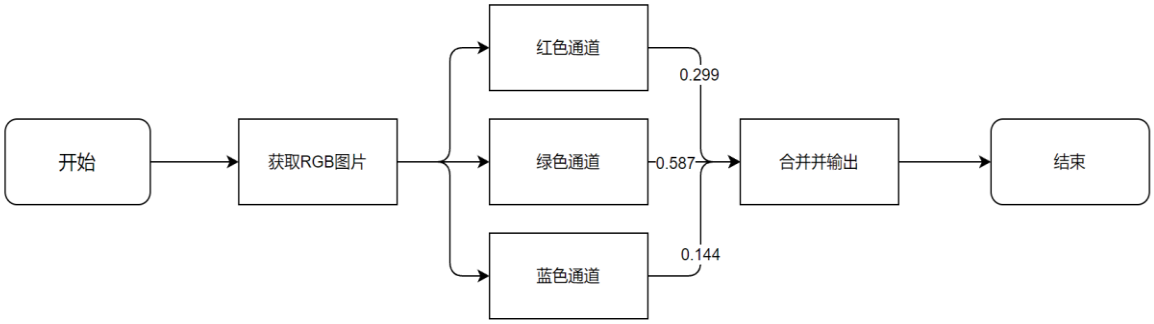
五、实验步骤

1、 实验平台采用 pycharm 和 Anaconda3。pycharm 是一个非常流程IDE（集成开发环境），主要用于python开发；Anaconda3 是一个主流计算科学库。在配置好 Anaconda3 的环境变量，在此基础上我们将 Anaconda3 的作为 pycharm 的编译环境。最后执行 pip3 install opencv-python 将 opencv 库导入即可。

2、 整体流程如下程序流程图。首先是获取三通道的 RGB 图片；接着我们需要将RGB 图片进行通道分离，分别为 R 通道、G 通道、B 通道，再根据灰度计算公式

```
Gray = R*0.299 + G*0.587 + B*0.114
```

计算灰度图；最后我们将三个通道进行合并，即完成整个灰度转换的过程。



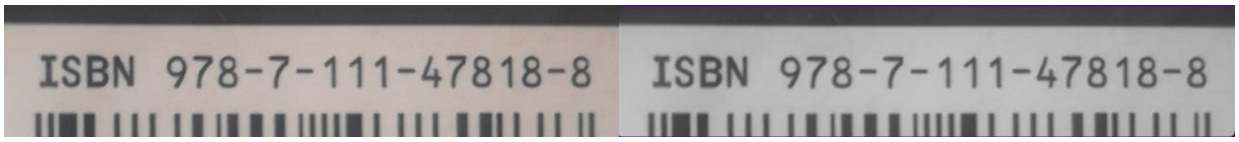
3、 最后将灰度图存储在名为gray_image.txt的内。

六、实验核心代码

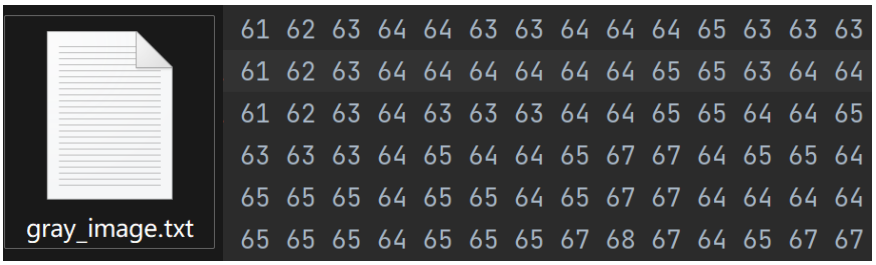
```
def rgb2gray(src):
    height = src.shape[0]
    width = src.shape[1]
    red_channel, green_channel, blue_channel = cv2.split(src)
    dst = np.zeros(red_channel.shape, red_channel.dtype)
    for h in range(height):
        for w in range(width):
            dst[h][w] = (red_channel[h][w] * 299 +
                         green_channel[h][w] * 587 +
                         blue_channel[h][w] * 144 +
                         500) / 1000
    np.savetxt('gray_image.txt', dst, "%d", " ")
    return dst
```

七、实验结果分析与讨论

1、原图与灰度图对比，左为原图，右为灰度图：



2、用txt文件存储灰度图，左为txt文件，右为文件中的一段内容



讨论：唯一值得讨论的一点就是灰度计算时，将系数乘上1000后整数化，这样可以避免浮点运算带来的额外计算量和可能存在的小数像素点。

实验二　图像分割

一、实验题目：

用阈值分割技术实现对ISBN号图片中字符的分割。

二、实验意义：

在计算机视觉中一个非常重要的基本问题是图像分割。分割后目标和物体区分出来便于后期处理，因此在视觉与图像处理领域中具有重要作用。

三、实验目的

1、了解图像分割的目的及意义，加深对图像的感性认识；

2、掌握简单的图像分割方法。

四、实验内容

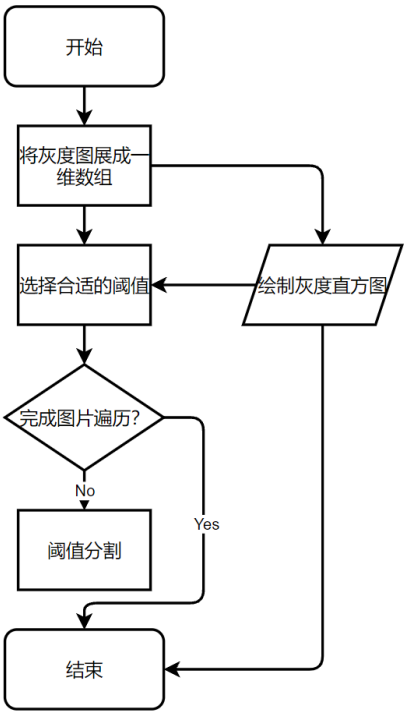
1、在实验一的基础上完成如下步骤；

2、画出ISBN号图像的灰度直方图；

3、根据该直方图，确定图像分割的阈值，将图像转变成只有字符和背景的二值图像。

五、实验步骤

1、首先获得实验一运算得到的灰度图，这点很重要，也是基础；接着我们再将灰度图的二维矩阵展平成一个一维向量,代码中src.ravel()就是起这个作用的；利用这个一维向量绘制一个数值为（0，255）的直方图，像素的按照对应数值依分类叠加，完成绘制灰度直方图绘制。



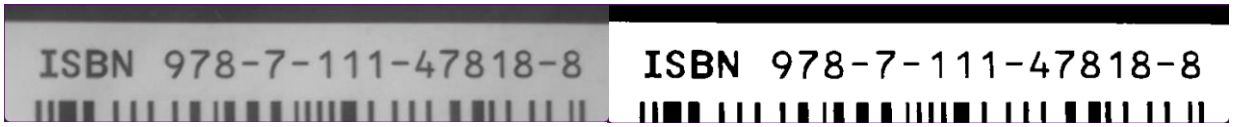
2、如上流程图，紧接着步骤1。我们根据绘制的直方图寻找最合适的阈值。再遍历灰度矩阵。依次根据阈值大小判断改编像素点。若是小于阈值，像素值更新为0，若是大于阈值，更新为255。最后返回分割后的二值图。

六、实验核心代码

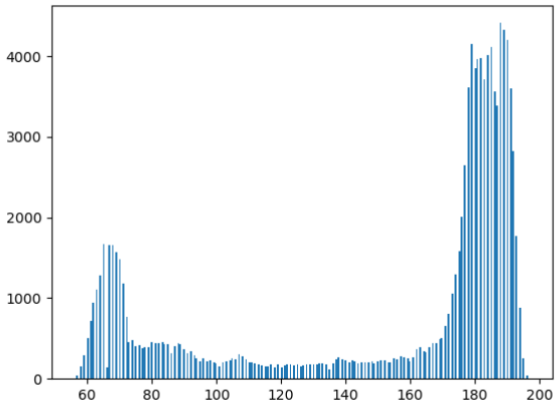
```
def cacle_hist(src):
    height = src.shape[0]
    width = src.shape[1]
    flat_array = src.ravel()
    plt.hist(flat_array, 256)
    plt.show()
    dst = np.zeros(src.shape, dtype=src.dtype)
    dst.fill(255)
    for h in range(height):
        for w in range(width):
            if src[h][w] <= 125:
                dst[h][w] = 0
            else:
                pass
    return dst
```

七、实验结果分析与讨论

1、原灰度图与阈值分割后的二值图对比。下左为灰度图，下右为二值图:



2、输出灰度直方图，如下图：



讨论：通过灰度直方图可以看出。数字以及条码的像素值主要集中在30-80之间，背景的像素集中再160-200之间。所以再80-160之间的数作为阈值都有一定可行性，不断修正后得到合理的阈值为125。

实验三　边缘检测

一、实验题目：

任意选用一种边缘检测算子实现对字符边缘的检测。

二、实验意义：

在计算机视觉中另一个非常重要的基本问题是边缘检测。边缘表现了目标边界， 因此在视觉与图像处理领域中具有重要作用。

三、实验目的

1、了解图像边缘检测的目的及意义，加深对边缘检测的感性认识；

2、了解边缘检测方法，利用某种边缘检测方法实现对图中字符边缘的检测。

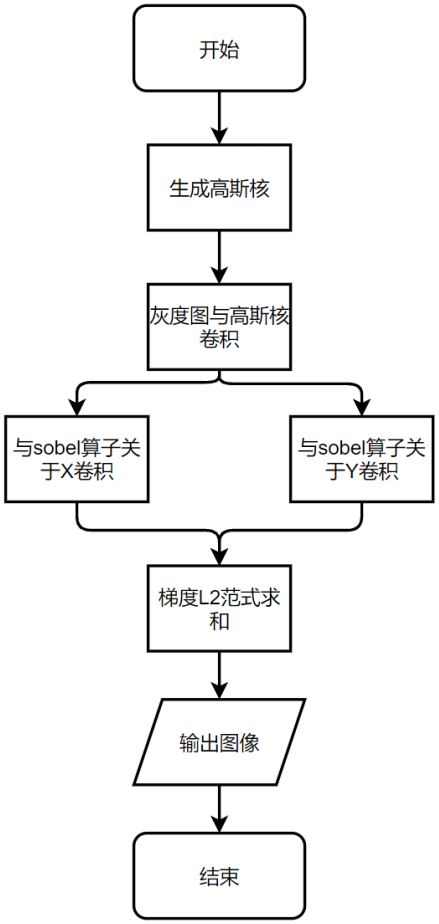
四、实验内容

1、首先对图片进行滤波处理；

2、对滤波后的图像选用一种边缘检测 算子进行边缘检测。

五、实验步骤

1、首先利用高斯分布公式，选取适合的size，也就是高斯核的大小，将严密的数学公式代码化，得到尺寸合适的高斯核。本实验采用大小为3的高斯核。其中该实验的整体算法流程图如下。



2、如上图，接续步骤1。我们将灰度图与高斯核进行卷积运算。与 3*3 的高斯核卷积一般会使原图尺寸减1，因此我们可以利用padding或者保持原图边缘不变的方法进行尺寸扩充。卷积操作在计算机视觉上一般指原图与给定的kernel进行对应位置相乘后数加。

3、接着我们利用sobel算子进行边缘提取。考虑到sobel算子有两个kernel，每个kernel分别都要与图像进行卷积运算，运算的结果分别代表图像在X轴方向上的梯度和图像在Y轴方向上的梯度。我们需要将两个方向上梯度进行L2范式求和并计算出最终梯度。通俗点就是我们两个方向上梯度平方求和再开根号的运算。紧接着我们设定一个梯度阈值，本实验设定为50。最后将边缘图像输出。至此，实验三流程结束。

六、实验核心代码

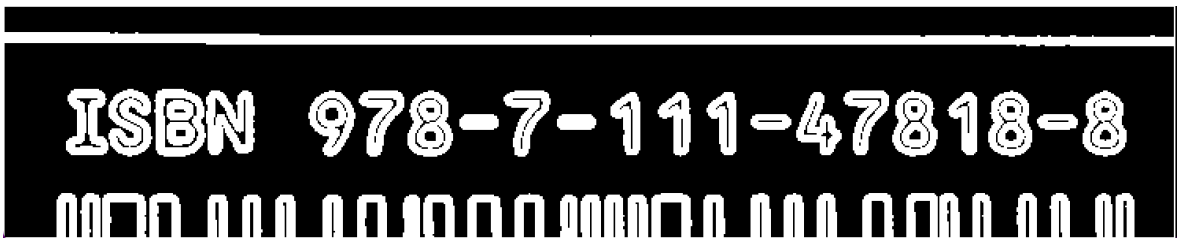
```
#生成高斯核
def gausskernel(size):
    sigma = 1.0
    gausskernel = np.zeros((size, size), np.float32)
    for i in range(size):
        for j in range(size):
            norm = math.pow(i - 1, 2) + pow(j - 1, 2)
            gausskernel[i, j] = math.exp(-norm / (2 * math.pow(sigma, 2)))
    sum = np.sum(gausskernel)
    kernel = gausskernel / sum
    return kernel
#高斯滤波
def gausssion_filter(src):
    height = src.shape[0]
    width = src.shape[1]
    dst = np.zeros((height, width), np.uint8)
    kernel = gausskernel(3)
    for h in range(1, height - 1):
        for w in range(1, width - 1):
            sum = 0
            for k in range(-1, 2):
                for l in range(-1, 2):
                    sum += src[h + k, w + l] * kernel[k + 1, l + 1]
            dst[h, w] = sum
    return dst
#sobel算子边缘检测
def sobel_edge(src):
    image = gausssion_filter(src)
```

```
    height = image.shape[0]
    width = image.shape[1]
    dst = np.zeros((height, width, 1), np.uint8)
    for i in range(0, height - 2):
        for j in range(0, width - 2):
            gy = image[i, j] * 1 + image[i, j + 1] * 2 + image[i, j + 2] * 1 - image[i + 2, j] * 1 - image[i + 2, j + 1] * 2 - image[i + 2, j + 2] * 1
            gx = image[i, j] * 1 + image[i + 1, j] * 2 + image[i + 2, j] * 1 - image[i, j + 2] * 1 - image[i + 1, j + 2] * 2 - image[i + 2, j + 2] * 1
            grad = math.sqrt(gx * gx + gy * gy)
            if grad > 50:
                dst[i, j] = 255
            else:
                dst[i, j] = 0
    return dst
```

七、实验结果分析与讨论

1、 原灰度图与进行高斯滤波后的灰度图对比。左为原灰度图，右为滤波后的灰度图。

2、输出边缘计算后的图片。



讨论：通过高斯滤波算法和边缘检测算法，我们发现这两种本质上是拿一个小矩阵遍历的与图像矩阵进行对应位置相乘求和，再将计算后的新值作为输出图像的像素值。这个步骤统称为卷积运算。事实证明，卷积运算在图像处理中有着至关重要的作用。一个好的算子，可以大大优化自己的算法。这给我们带来了很多思考。

实验四 图像特征提取

一、实验题目：

提取图像中每一个字符的特征。

二、实验意义：

计算机视觉的应用非常广泛，本实验就是让同学们对计算机视觉技术的应用有一个简单的了解。

三、实验目的

1、了解目标特征提取的意义，加深对特征提取的感性认识。

2、掌握基于几何特征的特征提取方法。

四、实验内容

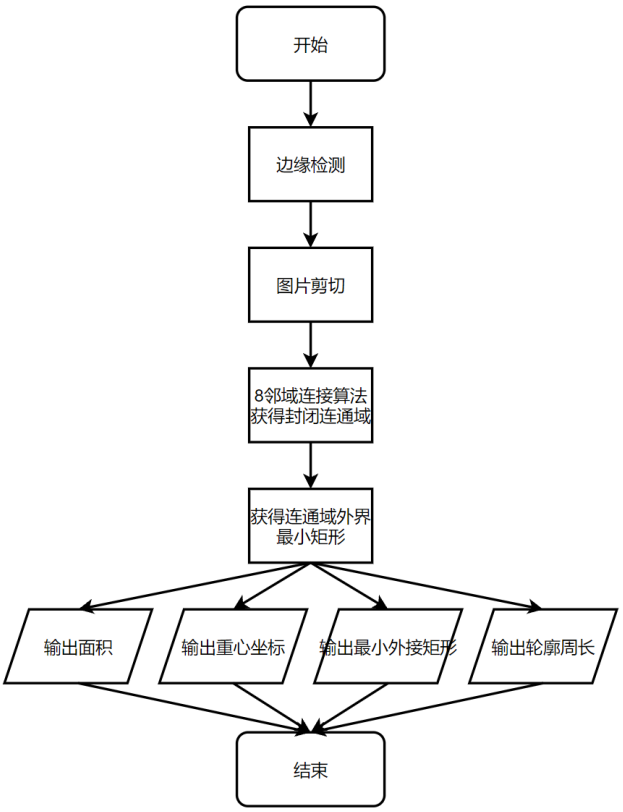
在实验二的基础上提取每一个字符的以下几个特征。

1、先找到二值图像中第一个值为 1 的点，通过其 8 邻域找到字符区域；

2、计算每一个区域面积、重心、最小平行轴外接矩形、轮廓长度等特征。

五、实验步骤

1、在实验二的基础上可以得到了图像的二值图，但是观察后发现二值图数字匹配存在两个干扰因素。一个是上方图像黑色边界，一个下方的条形区域。所以首先做的事情应该是图像分割。具体流程如下图。



2、如上图，首先进行图像剪切获得只包含数字块的区域。在此基础上，本文利用opencv提供的封装好的8邻域算法 cv2.connectedComponentsWithStats()获得最佳的封闭连通域，包括一些字符和数字。接着我们根据这些连通域的大小获得相应的属性，其中最重要也是最为麻烦的是最小外接矩形。我们根据封装好的算法获得相应的坐标。其中封闭域的面积，本文采用opencv提供的cv2.findContours()函数寻找连通域，配合cv2.arcLength()便可以计算连通域面积。最后我们将相关信息与对应连通域统一输出。至此，实验四结束。

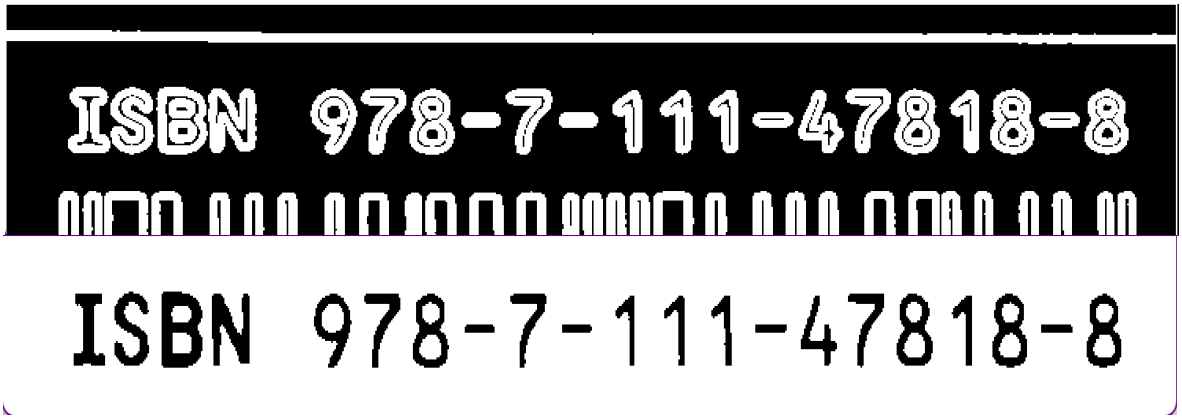
六、实验核心代码

```
def find_aera(src):
    height = src.shape[0]
    width = src.shape[1]
    start = -1
    end = -1
    for h in range(height):
        for w in range(width):
            if src[h][w] == 0:
                break
            elif w == width - 1:
                start = h + 5
            else:
                pass
        if start != -1:
            break
    for h in range(height - 1, 0, -1):
        for w in range(width):
            if src[h][w] == 0:
                break
            elif w == width - 1:
                end = h - 5
            else:
                pass
        if end != -1:
            break
    ROI = src[start:end, :]
    contours, hierarchy = cv2.findContours(ROI,
mode=cv2.RETR_TREE,method=cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(ROI, contours,-1,color=255,thickness=1)
    for i in contours:
      length = cv2.arcLength(i, True)
      print(length)
    ROI_inv = cv2.bitwise_not(ROI)
```

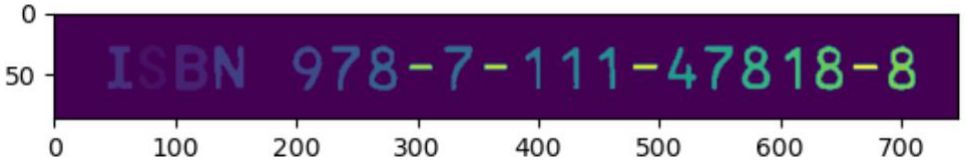
```
    retval, labels, stats, centroids = cv2.connectedComponentsWithStats(ROI_inv,
labels=None, stats=None, centroids=None, connectivity=8, ltype=None)
    print(stats[:,0])
    stats_ = np.argsort(stats[:,0])
    print(stats_)
    for i in stats_:
        print('The rectangular box coordinates are (', stats[i][0], stats[i][1], ')\t(',
            stats[i][0] + stats[i][2], stats[i][1] + stats[i][3], ')\t\t',
            'The area are ', stats[i][4], '\t\t', 'The centroid', centroids[i], '\t\t', 'The lenth are', 2
* (stats[i][2] * stats[i][3]))
```

七、实验结果分析与讨论

1、原二值图与切割处理后的图片对边。上为原二值图，下为切割后的二值图并反转像素值。



2、数字符号轮廓的连通域图，不同连通域由不同的颜色标出，下图是一个32位的彩色图。



3、最终输出的结果，包含对应连通域的最大外接矩形坐标、连通域面积、重心坐标、轮廓长度。其中第一个为图片本身的相关属性

```
The rectangular box coordinates are ( 0 0 ) ( 747 87 )      The area are  58992       The centroid [375.66197111  42.86001492]       The lenth are 129978
The rectangular box coordinates are ( 45 28 )   ( 65 63 )       The area are  371       The centroid [54.06469003 45.57142857]       The lenth are 1400
The rectangular box coordinates are ( 71 27 )   ( 93 63 )       The area are  401       The centroid [81.19451372 45.04738155]       The lenth are 1584
The rectangular box coordinates are ( 101 27 )  ( 127 64 )      The area are  669       The centroid [112.34977578  45.20328849]      The lenth are 1924
The rectangular box coordinates are ( 133 28 )  ( 157 63 )      The area are  505       The centroid [144.3960396   44.44158416]       The lenth are 1680
The rectangular box coordinates are ( 198 28 )  ( 222 64 )      The area are  352       The centroid [210.08522727  43.53693182]      The lenth are 1728
The rectangular box coordinates are ( 230 28 )  ( 253 64 )      The area are  281       The centroid [241.29893238  39.70462633]      The lenth are 1656
The rectangular box coordinates are ( 261 28 )  ( 284 64 )      The area are  408       The centroid [271.52941176  46.43137255]      The lenth are 1656
The rectangular box coordinates are ( 294 43 )  ( 313 47 )      The area are  74        The centroid [303.          44.54054054]       The lenth are 152
The rectangular box coordinates are ( 323 28 )  ( 346 64 )      The area are  260       The centroid [334.47307692  39.36538462]      The lenth are 1656
The rectangular box coordinates are ( 356 43 )  ( 375 47 )      The area are  74        The centroid [364.75675676  44.5       ]       The lenth are 152
The rectangular box coordinates are ( 390 29 )  ( 402 63 )      The area are  153       The centroid [398.20261438  42.01960784]      The lenth are 816
The rectangular box coordinates are ( 421 29 )  ( 434 63 )      The area are  177       The centroid [429.96045198  42.81355932]      The lenth are 884
The rectangular box coordinates are ( 451 29 )  ( 464 63 )      The area are  177       The centroid [459.5480226   43.04519774]      The lenth are 884
The rectangular box coordinates are ( 479 43 )  ( 499 47 )      The area are  79        The centroid [488.37974684  44.48101266]      The lenth are 160
The rectangular box coordinates are ( 508 29 )  ( 531 63 )      The area are  253       The centroid [518.5770751   47.79841897]      The lenth are 1564
The rectangular box coordinates are ( 538 28 )  ( 561 64 )      The area are  262       The centroid [549.69083969  39.5       ]       The lenth are 1656
The rectangular box coordinates are ( 569 28 )  ( 592 63 )      The area are  412       The centroid [579.82524272  46.00485437]      The lenth are 1610
The rectangular box coordinates are ( 604 29 )  ( 617 63 )      The area are  186       The centroid [612.91935484  42.67741935]      The lenth are 884
The rectangular box coordinates are ( 630 28 )  ( 652 64 )      The area are  414       The centroid [640.24879227  46.23188406]      The lenth are 1584
The rectangular box coordinates are ( 661 43 )  ( 681 47 )      The area are  79        The centroid [670.37974684  44.51898734]      The lenth are 160
The rectangular box coordinates are ( 690 28 )  ( 712 64 )      The area are  410       The centroid [700.36097561  46.56829268]      The lenth are 1584
```

讨论：实验四是四个实验中最为困难的一个实验。求最大连通域和相关面

积，重心，最小外接矩形的过程中，本实验并没有给出源码，而是直接调用 opencv 提供的内置接口。这是我四个实验中最为遗憾的一部分，没有自己手写完源码。确实也是自己能力有限，时间有限。除此之外，上述四个实验在算法复杂度，算法合理性上仍有很多优化的空间。望在今后的学习中继续自己在能力上的短板。