实验报告

518021910346 郭黛筠

前期部分由队友刘勉之同学完成**肺实质分割**，后期协助完成**调参**与**图片预处理**等工作

我负责的部分为**使用纹理分析方法对提取肺实质的CT图像进行病灶区域进行分割**，下述部分未特殊说明的即为个人完成部分

预处理

在进行标准分割病灶获取灰度直方图和后续计算交并比的任务中，都需要将标注图与原图进行相关匹配

首先提取出标注图中相应颜色的部分作为mask，以黄色的网状病灶为例，将标注图转换为hsv模式，依据下表提取相应颜色。再将mask与原标注图结合获得含纹理的mask

	黑	灰	白	红		橙	黄	绿	青	蓝	紫
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34；	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

图1.hsv颜色范围表

```
# 提取黄色
hsv = cv2.cvtColor(standard, cv2.COLOR_BGR2HSV)
low_hsv = np.array([26, 43, 46])
high_hsv = np.array([34, 255, 255])

mask = cv2.inRange(hsv, lowerb=low_hsv, upperb=high_hsv)

standard[mask == 0] = (0, 0, 0)
gray = cv2.cvtColor(standard, cv2.COLOR_BGR2GRAY)
cv2.imwrite(os.path.join(pretreat_path, filename), gray)
```

此时结果如下，可以发现有多余放缩水印，且受黄色标记影响，与原图相比，灰度并不一致

图2.直接截取黄色标记结果

进一步去除左下角水印

```
for i in range(row_size - 150, row_size):
    for j in range(200):
        mask[i][j] = 0
```

观察到标注图是原图的一部分，试图使用简单模板匹配

```
def match(template, target):
    result = cv2.matchTemplate(target, template, cv2.TM_SQDIFF_NORMED)
    # 归一化处理
    cv2.normalize(result, result, 0, 1, cv2.NORM_MINMAX, -1)
    # 寻找矩阵中的最大值和最小值的匹配结果及其位置,
    # 在cv2.TM_SQDIFF_NORMED方法下, 值越小匹配度越高
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
    strmin_val = str(min_val)
    # min_loc：矩形定点, strmin_val：匹配值
    return min_loc, strmin_val
```

此时发现匹配有极大问题，检查发现标注图尺寸为1488*1096，原图则为512*512，因此使用更复杂的基于FLANN的特征匹配，算法思路见注释

```
def match(template, target):
    MIN_MATCH_COUNT = 10  # 设置最低特征点匹配数量为10
    sift = cv2.SIFT_create()  # 创建sift检测器
    kp1, des1 = sift.detectAndCompute(template, None)  # 获得关键点和特征向量
    kp2, des2 = sift.detectAndCompute(target, None)
    # 设置Flann的参数
    FLANN_INDEX_KDTREE = 0
    indexParams = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    searchParams = dict(checks=50)  # 递归遍历次数
    flann = cv2.FlannBasedMatcher(indexParams, searchParams)
    matches = flann.knnMatch(des1, des2, k=2)  # 使用KNN算法匹配，找最近邻和次近邻两个
点

    good = []
    # 比值检测法:舍弃最近邻与次近邻两者距离比大于0.7的匹配
    for m, n in matches:
```

```
        if m.distance < 0.7 * n.distance:
            good.append(m)
    if len(good) > MIN_MATCH_COUNT:
        # 获取关键点的坐标 转化为一列
        src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1,
2)
        dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1,
2)
        # 计算变换矩阵和MASK
        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
        # matchesMask = mask.ravel().tolist()
        h, w = template.shape
        # 使用得到的变换矩阵对原图像的四个角进行变换，获得在目标图像上对应的坐标
        pts = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1,
0]]).reshape(-1, 1, 2)
        dst = cv2.perspectiveTransform(pts, M)  # 变化后的顶点位置
        print(np.int32(dst))
        cv2.polylines(target, [np.int32(dst)], True, (255, 0, 0), 2,
cv2.LINE_AA)
        return np.int32(dst)
    else:
        print("Not enough matches are found - %d/%d" % (len(good),
MIN_MATCH_COUNT))
```

匹配结果如下，能够正确框选（右图白框）

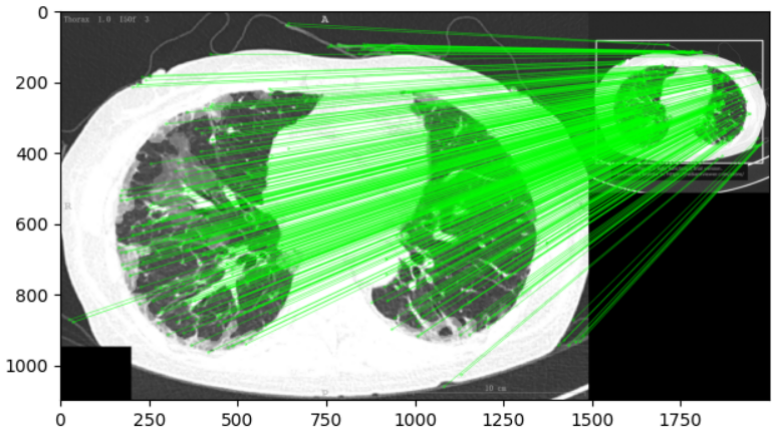

图3.FLANN图像匹配结果

因此可以使用匹配得来的矩形位置切割缩放原图及标注图，使其相合。输出发现坐标均为`[[[ 20 82]] [[ 20 428]] [[490 428]] [[490  82]]]`，即对应原图一个470*346的区域

```
# 缩小标注图
standard = cv2.resize(standard, [470, 346])

# 新建与origin对齐的mask
row_size, col_size = origin.shape[:2]
newmask = np.zeros(origin.shape, dtype=np.uint8)
for i in range(small_row_size - 1):
    for j in range(small_col_size - 1):
        newmask[i + 82][j + 20] = mask[i][j]

# 匹配原图
origin[newmask == 0] = 0
cv2.imwrite(os.path.join(pretreat_path, originname), origin)
```

结果如下

图4.最终标记提取结果

honey combing同理，匹配坐标为`[[[ -9  60]] [[ -9 451]] [[520 451]] [[520  60]]]`，即529*391，此时出现超出原图的部分，进行相关处理

```
small_col_size = min(small_col_size, col_size)
for i in range(small_row_size - 1):
    for j in range(small_col_size - 1):
        newmask[i + 60][j] = mask[i][j + 9]  # honeycombing
```

至此图像预处理完成，将相关mask存至pretreat文件夹相应目录下

灰度直方图

将预处理后提取出的病灶蒙版生成灰度图，绘制直方图。

以**reticular**为例

```
# 读取网状病灶
imgPath = os.path.join(reticular_path, filename)
img = cv2.imread(imgPath)
# 生成灰度图
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# 生成直方图
plot = list(filter(lambda a: a != 0, gray.flatten()))
plt.hist(plot, bins=20)
plt.show()
```

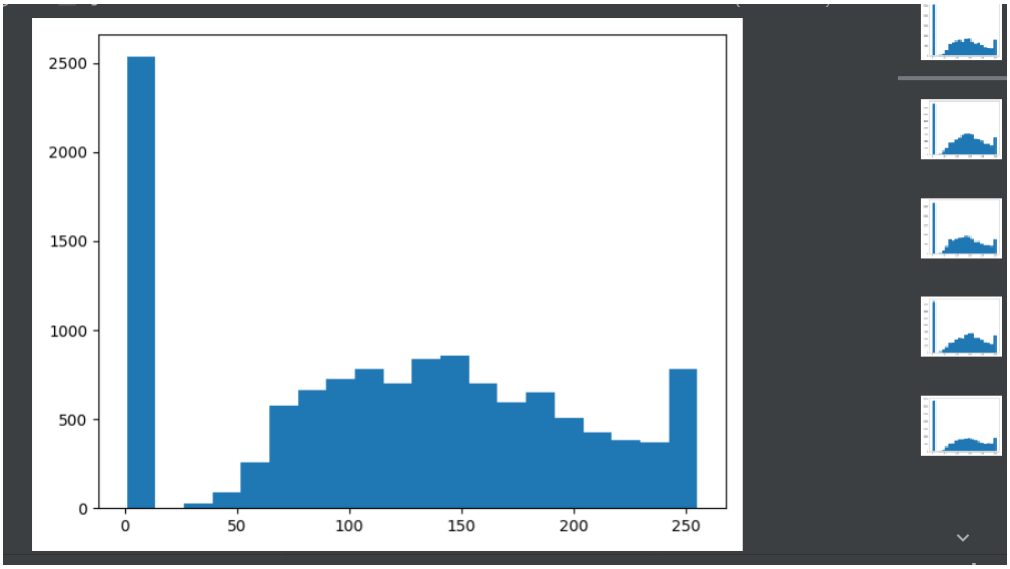

图5.网状病灶标注图的灰度图

对应绘制原图经过肺实质分割后的灰度直方图

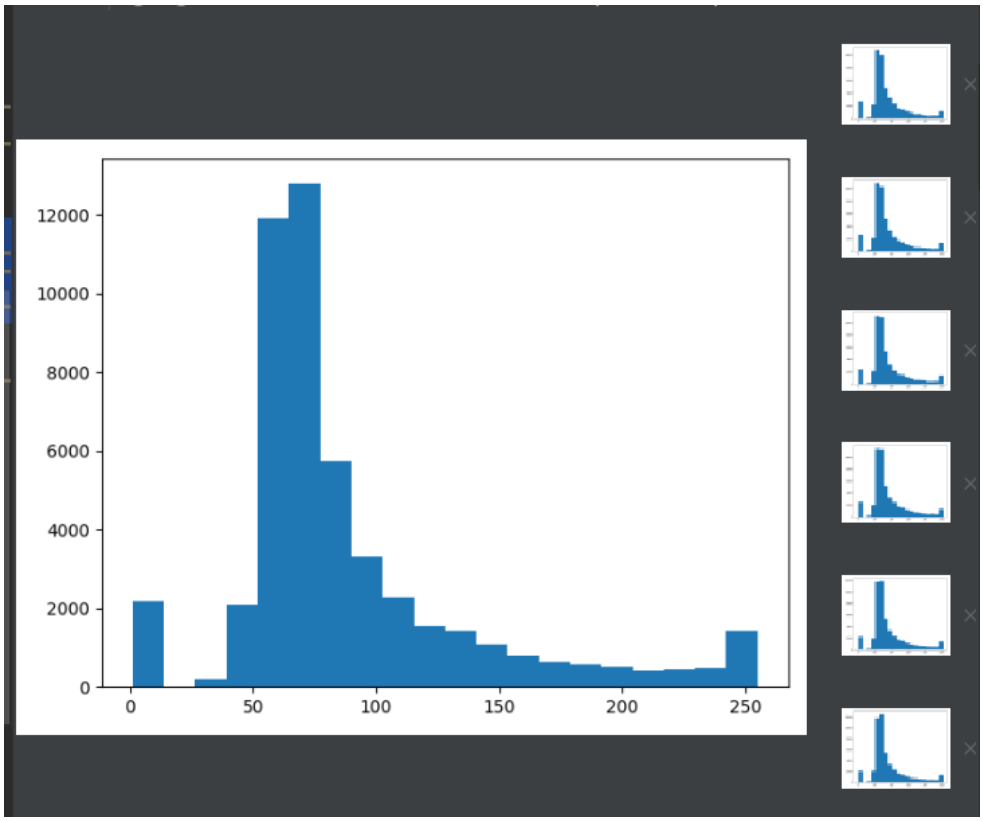

图6.原图经过肺实质分割后的灰度直方图

对比观察可得，灰度值170到200区域主要由reticular病灶组成

因此取值170-200进行尝试，发现会有较多多余判定，当reticular占图片中大多区域时能较好识别，但占少数区域时会有较多假阳性误差，为直观起见，以提取出的标注掩膜为底，如下图，绿色标记原图中灰度值在[170,200]区间范围内

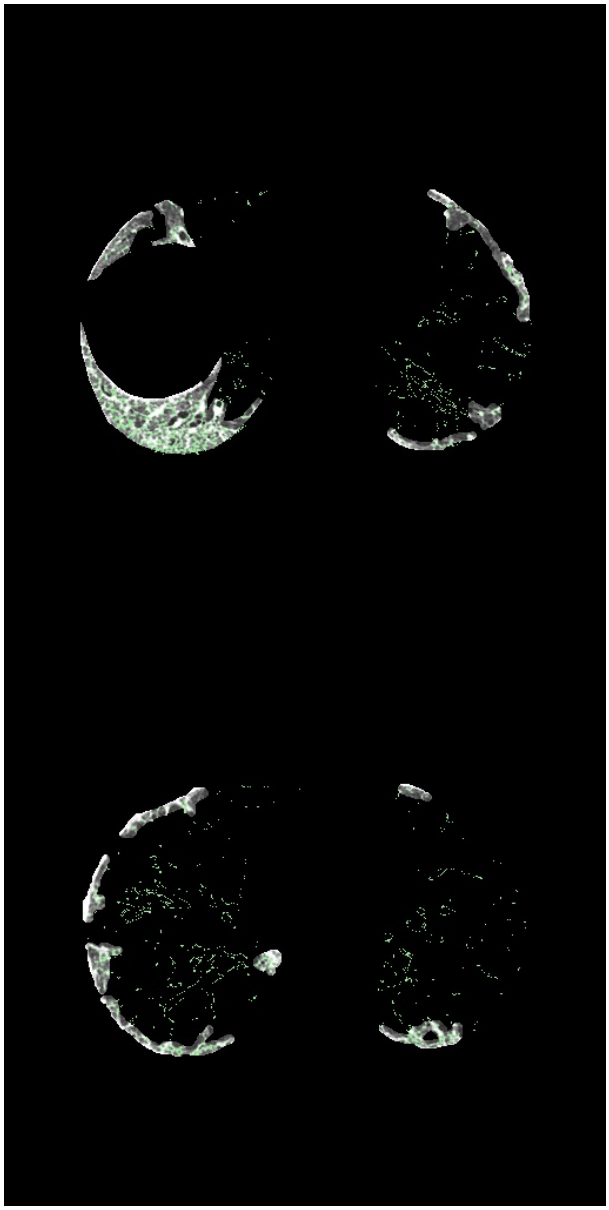

图7.1号图（左）与20号图（右）对reticular的判定结果

honeycombing同理

病灶蒙版生成灰度直方图

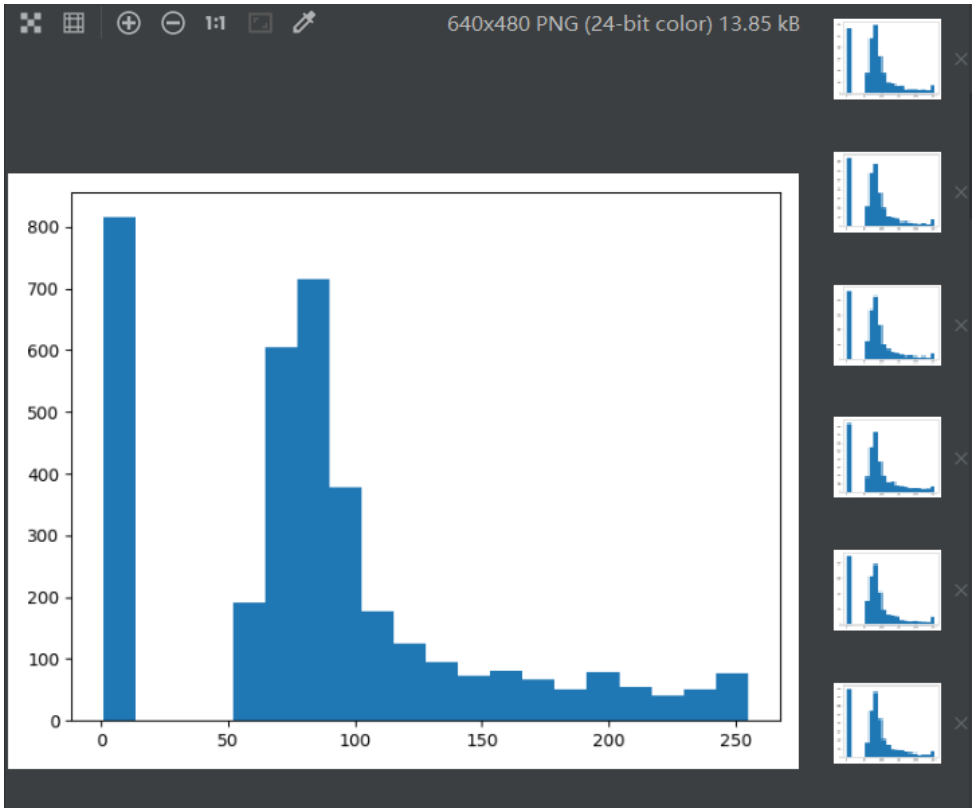

图8.蜂窝病灶标注图的灰度图

选用[60,100]获得如下图像，效果颇糟，有极大误判

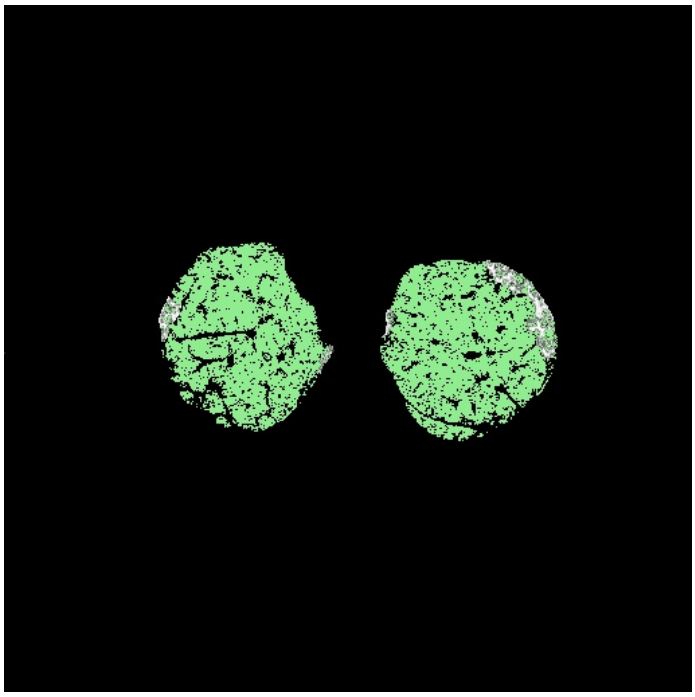

图9.1号图灰度范围[60,100]对honey combing的判定结果

但观察发现灰度值高的地方主要集中在honey combing和reticular处，采用范围[200,255]有更好的效果

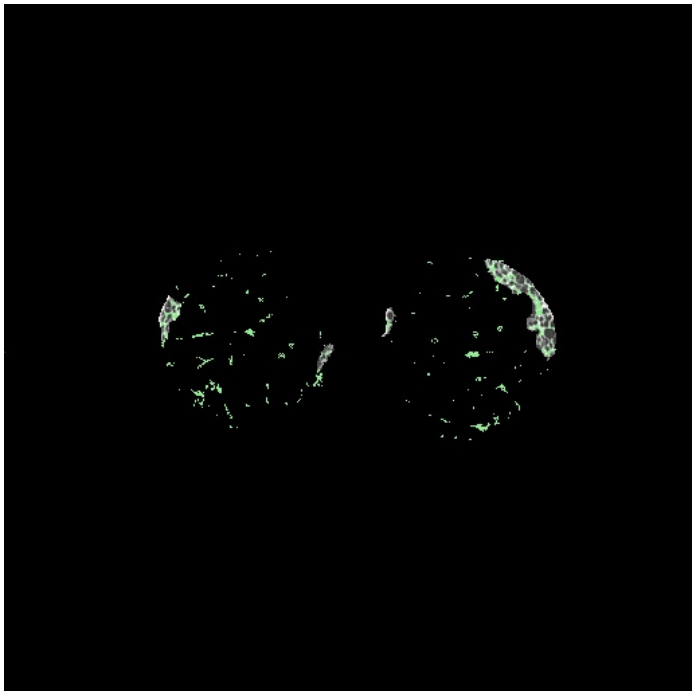

图10.1号图灰度范围[200,255]对honey combing的判定结果

实验分析

通过灰度直方图进行病灶分割能初步区分相差较大的纹理，但无法正确提取各纹理的特征，存在较多噪点，且对所筛选纹理的选择较为零散，可以考虑以窗口为单位对灰度进行分析判断

灰度共生矩阵

灰度共生矩阵被定义为从灰度为i的像素点出发，某个固定距离的上的点灰度值为j的概率，即，所有估计的值可以表示成一个矩阵的形式，以此被称为灰度共生矩阵。通过灰度共生矩阵能够计算出一些统计量作为纹理分类的特征

构建灰度共生矩阵

在图像中任意一点（x,y）及偏离它的一点(x+a,y+b)（其中a，b为整数）构成点对。设该点对的灰度值为(f1,f2)，统计整幅图像中各(f1,f2)出现的次数，然后排列成一个方阵，再用(f1,f2)出现的总次数将它们归一化为出现的概率P(f1,f2)，由此产生的矩阵为灰度共生矩阵。

ab的合理取值受纹理大小影响，此处由于honey combing和reticular纹理均较为细致，因此ab取值应该较小

为避免过于繁琐的计算，先减小灰度级数，其中max_gray_level是原图中最大灰度级

```
# 减小灰度级数
if max_gray_level > gray_level:
    for j in range(height):
        for i in range(width):
            img[j][i] = img[j][i] * gray_level / max_gray_level
```

再进行相关计算

```
# 计算共生矩阵
for j in range(height - dy):
    for i in range(width - dx):
        x = img[j][i]
        y = img[j + dy][i + dx]
        ans[x][y] += 1
```

并将之归一化即转为频率，除此之外归一化也避免了后续计算溢出等

```
sum = (height - dy) * (width - dx)
for i in range(gray_level):
    for j in range(gray_level):
        ans[i][j] = ans[i][j] / sum
```

便得到了灰度共生矩阵`ans`

计算特征值

- **对比度**

  度量矩阵的值是如何分布的和图像中局部变化的多少，反应了图像的清晰度和纹理的沟纹深浅。纹理的沟纹越深，反差越大，效果越清晰；反之，对比值小，则沟纹浅，效果模糊。

$$Con = \sum_{i}\sum_{j}(i-j)^2 P(i,j)$$

- **能量**

  能量变换反映了图像灰度分布均匀程度和纹理粗细度。若灰度共生矩阵的元素值相近，则能量较小，表示纹理细致；若其中一些值大，而其它值小，则能量值较大。能量值大表明一种较均一和规则变化的纹理模式。

$$Asm = \sum_{i}\sum_{j} P(i,j)^2$$

- **熵**

  图像包含信息量的随机性度量。当共生矩阵中所有值均相等或者像素值表现出最大的随机性时，熵最大；因此熵值表明了图像灰度分布的复杂程度，熵值越大，图像越复杂。

$$Ent = -\sum_{i}\sum_{j} P(i,j)\log P(i,j)$$

- **逆方差**

  逆方差反映了图像纹理局部变化的大小，若图像纹理的不同区域间较均匀，变化缓慢，逆方差会较大，反之较小。

$$H = \sum_{i}^{N-1}\sum_{j}^{N-1}\frac{P(i,j|d,\theta)}{1+(i-j)^2}$$

代码如下

```
for i in range(gray_level):
    for j in range(gray_level):
        contrast += (i - j) * (i - j) * m[i][j]
        energy += int(m[i][j] * m[i][j])
        homogeneity += m[i][j] / (1 + (i - j) * (i - j))
        if m[i][j] > 0.0:
            entropy += m[i][j] * math.log(m[i][j])
```

绘制对应图像

在掩膜图上滑动小窗口计算共生矩阵和特征值，为提高计算效率，仅计算以灰度值不为0的像素为中心的小窗口，小窗口设置为7*7的大小，方向设为(2,2)

各窗口的特征值存储在中心位置，将该矩阵值作为灰度进行相关放缩后绘制

观察发现此时四种特征值的效果均不是特别好但勉强可区分

以对比度及能量为例，病灶处及其周围无明显区别，且中部有较多噪声，运行速度也很慢

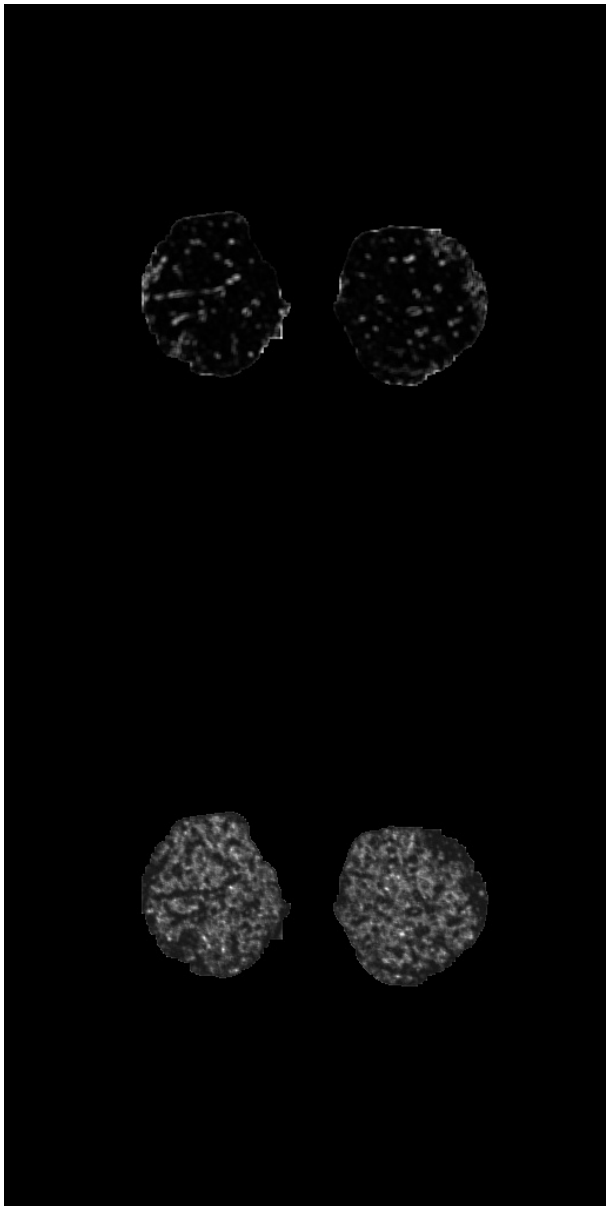

图11.蜂窝1号图对比度（左）与能量（右）绘制结果

但同时可以观察到中间部分与周边病灶有较明显的分界感，因此提出先刨除大连通域的设想，交由另一位组员实现

对比初步刨除后的图与标注图可发现，此时剩下的多出来可能被误判的与蜂窝病灶在灰度上有较明显的差别

图12.蜂窝1号图初步删除大连通域（左）与对应标记图（右）

绘制标注图的灰度共生矩阵对比度项所生成的直方图，并以$(0,1500) \rightarrow (0,255)$的映射绘制成图

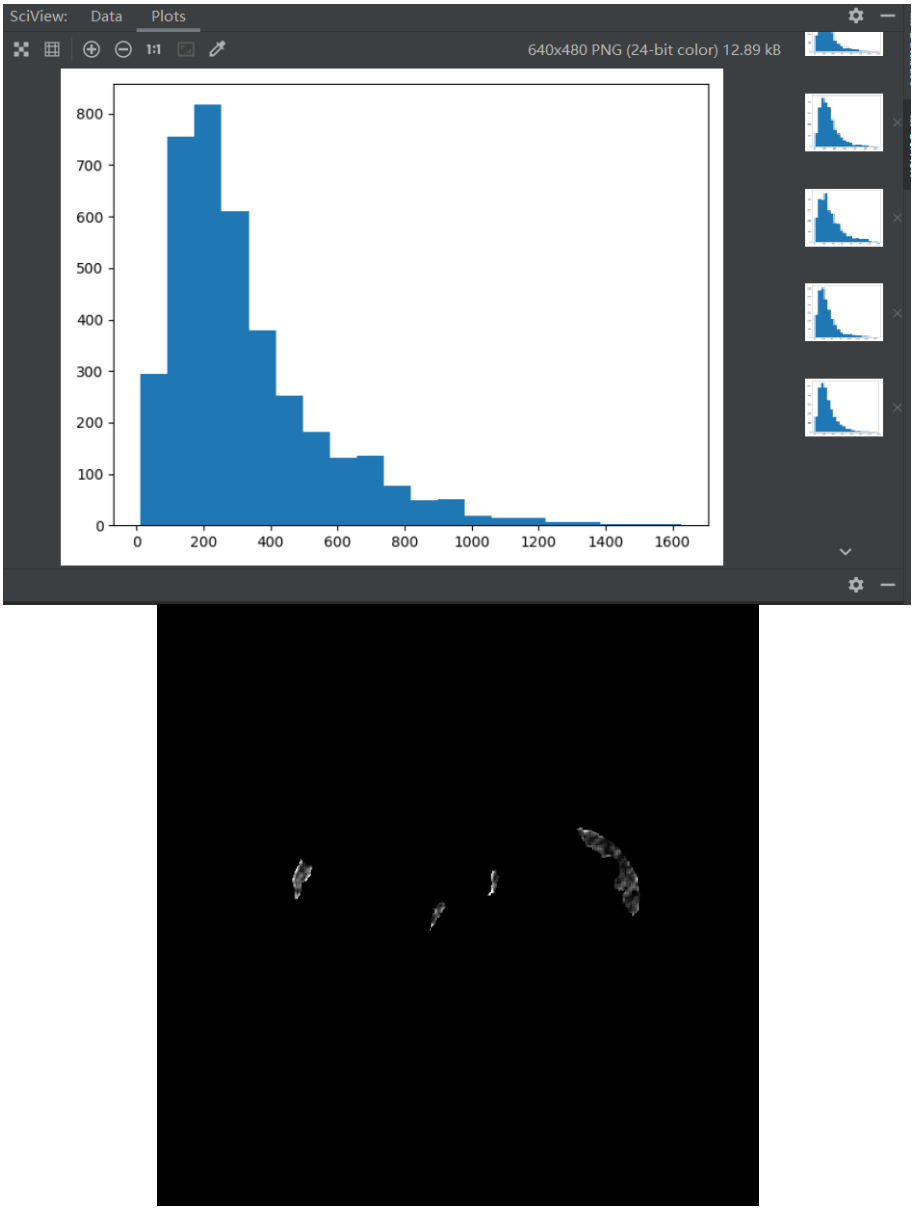

图13.蜂窝1号标注图对比度直方图（左）与对应映射成图（右）

观察可以发现对比度大多处于（100，400）之间

而经过初步删除大连通域的原始图绘制的直方图如下

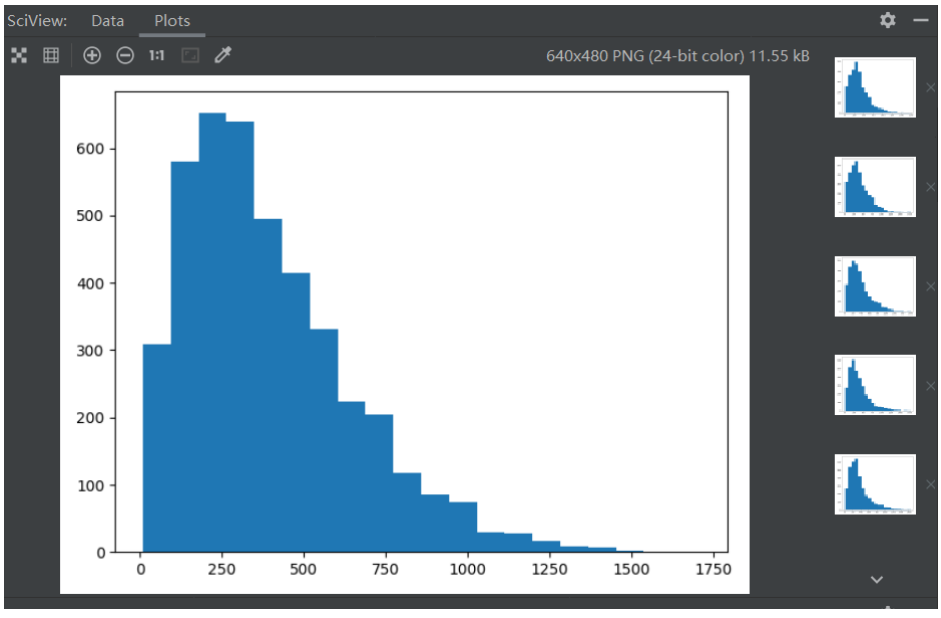

图14.蜂窝预处理后的对比度直方图

对比可以发现与标注图相比在200左右值数明显偏少，700以上虚高

参数调整及优化

为避免空洞被挖除，优化GLCM的实现，不再以1为步长滑动，而是以窗口大小滑动

对各特征值进行参数调整和尝试，该部分调参后续由队友接手

网状部分由本人调参

在过程中发现白色长条病灶在特征值上易于网状病灶混淆，无法有效删除。因此结合灰度直方图进行尝试，对每个窗口进行灰度平均值的计算并对应进行筛选，发现效果不佳。

```
def computeGray(w):
    height, width = w.shape[:2]
    sum = 0
    for i in range(height):
        for j in range(width):
            sum += w[i][j]
    return sum
```

后又进一步尝试对窗口内灰度值较高的像素进行统计，即统计灰度分布，效果依旧不佳，故不采用

```
for i in range(height):
    for j in range(width):
        sum += w[i][j]
        if w[i][j] > threshold:
            num += 1
acNum = int(height * width * k)
acSum = int(threshold * height * width * j)
```

最后综合考虑，所得参数为使用熵计算，window_size为（5，5），方向（0，1）

选取（-11，0）范围，preprocess调参 size阈值300 ，闭操作5开操作4

在此参数下标注图如下

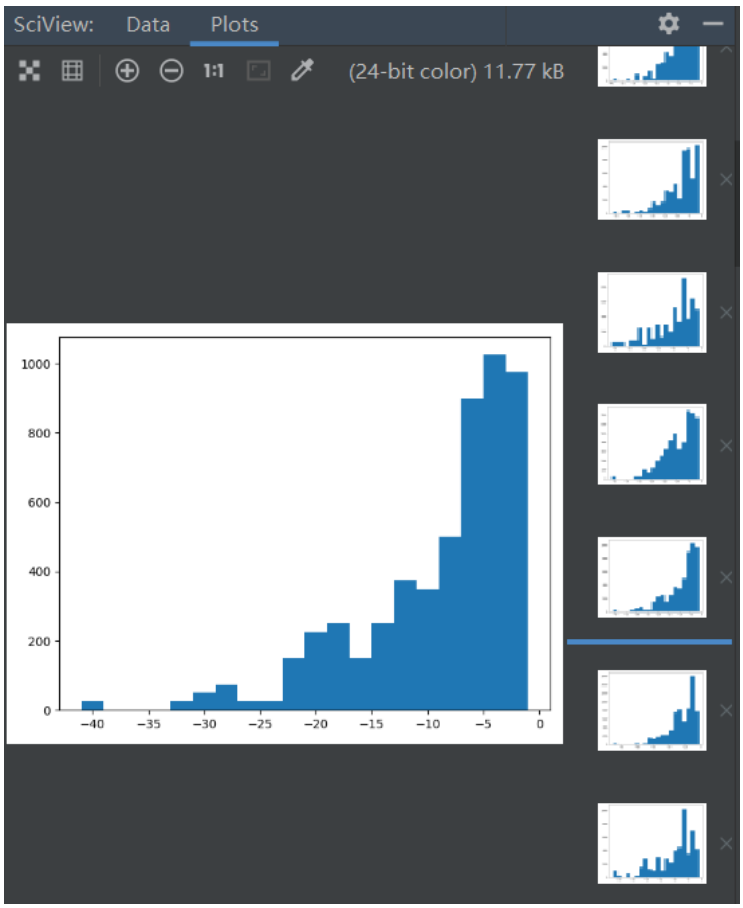

图15.网状病灶标注图的熵的直方图

最终结果如下

```
0100001.jpg   0.5168886447892352
0200001.jpg   0.5129879448909299
0300001.jpg   0.5577094758321642
0400001.jpg   0.5099079189686925
0500001.jpg   0.4846518892292906
0600001.jpg   0.5112094395280236
0700001.jpg   0.4299842043815672
0800001.jpg   0.4493457328614929
0900001.jpg   0.43561691412694276
1000001.jpg   0.43557111040407254
1100001.jpg   0.3796082060774333
1200001.jpg   0.3394739096906863
1300001.jpg   0.3670537634408602
1400001.jpg   0.3888254486133768
1500001.jpg   0.4170941657247196
1600001.jpg   0.3806097330051127
1700001.jpg   0.3132795304475422
1800001.jpg   0.36411688441208734
1900001.jpg   0.343937575030012
2000001.jpg   0.34812851937727723
 avg: 0.424300050541576
```

```
0100001.jpg   0.4648910411622276
0200001.jpg   0.4446529080675422
0300001.jpg   0.5510818953711312
0400001.jpg   0.5631119343741731
0500001.jpg   0.6204418418951291
0600001.jpg   0.47289024923480544
0700001.jpg   0.5062473071951745
0800001.jpg   0.549989650175947
0900001.jpg   0.5952599669300018
1000001.jpg   0.5982205163991626
1100001.jpg   0.5891691656376786
1200001.jpg   0.529807342784442
1300001.jpg   0.5566717210007581
1400001.jpg   0.5794493431707756
1500001.jpg   0.6045417680454177
1600001.jpg   0.5674227540632256
1700001.jpg   0.6332378223495702
1800001.jpg   0.5687745014487813
1900001.jpg   0.6132723112128147
2000001.jpg   0.5912685337726524
 avg: 0.5600201287145705
```

图16.网状病灶（左）与蜂窝病灶（右）分割的交并比结果

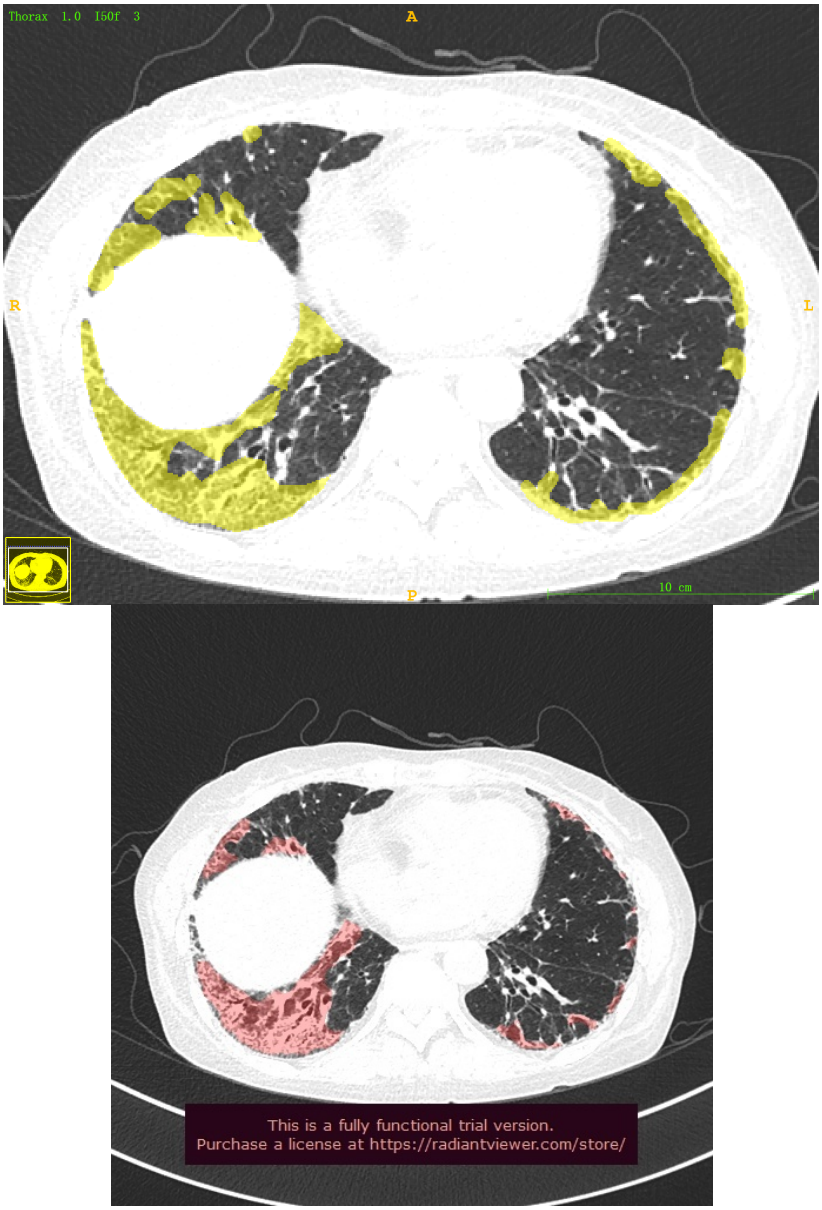

图17.网状5号图标注图（左）与程序识别图（右）

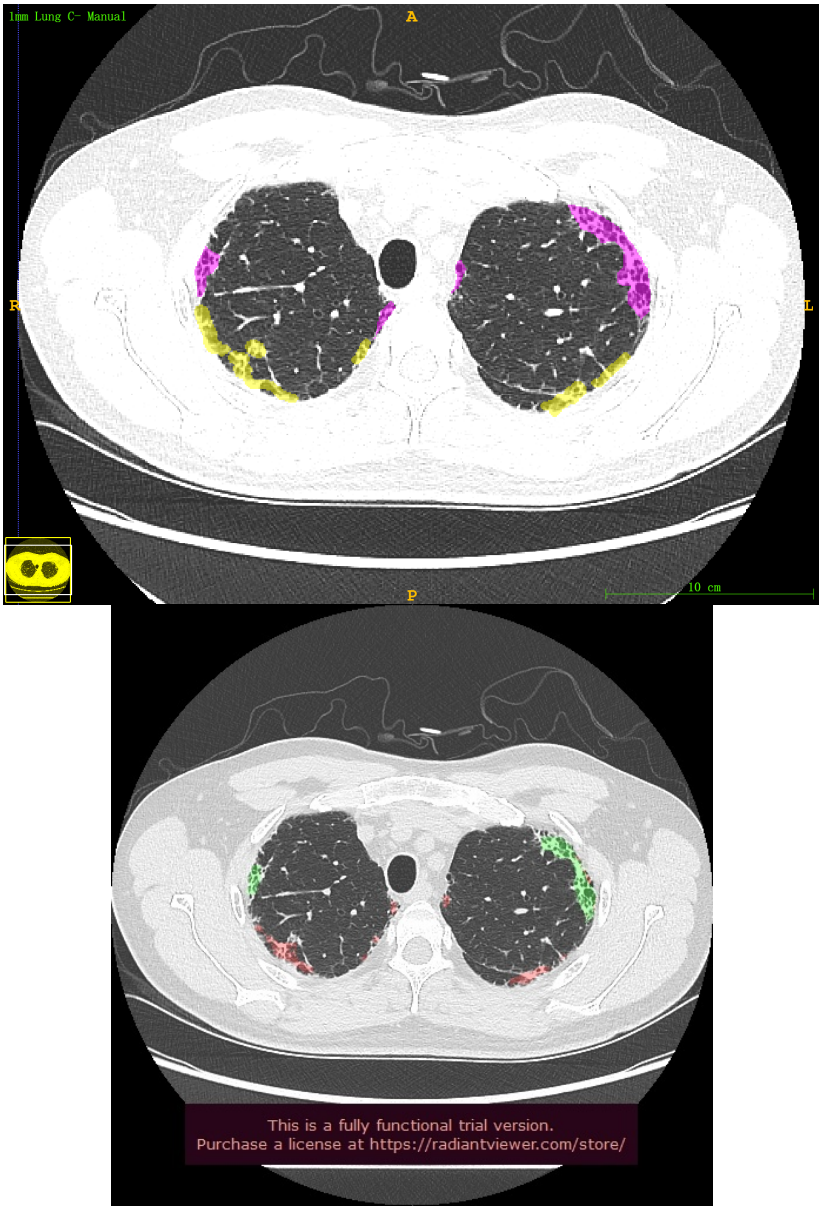

图18.蜂窝1号图标注图（左）与程序识别图（右）

最终结果

网状病灶：

平均交并比为：**0.424300050541576**

其中**0300001.jpg** 文件夹的交并比最大，为： **0.5577094758321642**

蜂巢病灶：

平均交并比为：**0.5600201287145705**

其中**1700001.jpg** 文件夹的交并比最大，为： **0.6332378223495702**

实验分析

大连通域的删除对网状及蜂窝两种病灶的提取有着非常明显的提升效果，大幅减少噪点干扰的同时保留蜂窝及网状的特点

蜂窝病灶在去除大连通域后效果较好，存在极少数与网状混淆的情况，可以考虑进一步对小连通域按照处于边缘以及近似圆形进行筛选

在网状病灶的识别过程中，观察发现部分白色长条病灶易与网状病灶混淆，大致原因为灰度均值等区别不够明显，且特征值会混在一起，可以考虑采用多个特征值及方向等同时筛选，或使用相似性计算与标注图比较来进行处理，由于时间关系不予实现

同时网状病灶通常分布在脉络不清的整块边界内，容易被忽视，不易计算共生矩阵。但同时基于网状病灶在边界处的特点，可以考虑对病灶提取结果进行进一步处理，删除非边缘处的较大连通域

同时，由于标注图中原本的标注具有一定的位移偏差，程序提取出来的标准病灶存在一定偏误，在实验过程中带来了一定误差

补充说明

代码运行时，仅需运行`1.py`及`part2.py`两个文件，分别代表第一部分及第二部分。肺实质分割结果在`honeycombing_parenchyma`及`reticular_parenchyma`文件夹中，最终标定结果在`honeycombing_result`及`reticular_result`文件夹的`final`文件夹中

补充了基于FLANN图像匹配的算法思路，补充了网状部分结合灰度进行筛选的失败过程，对灰度直方图与灰度共生矩阵的实验分析进行了补充，并补充了最终结果图的展示

参考文献

python+opencv 获取图片中指定颜色的部分_飞天土豆的博客-CSDN博客

opencv+python实现图像匹配----模板匹配、特征点匹配*GaoSimin-CSDN博客*python 模板匹配

OpenCV—python 角点特征检测之三（FLANN匹配）*wsp_1138886114的博客-CSDN博客*flann python

[图像纹理——灰度共生矩阵*sunny的专栏-CSDN博客*灰度共生矩阵]
(https://blog.csdn.net/guanyuqiu/article/details/53117507)