数字图象处理

综合作业二

2019010485　自 91　刘祖炎*

2021 年 12 月 12 日

1　肺部分割原理及代码分析

肺部分割的基本思路过程为：首先从 nii 文件中读取对应三维数据，将其保存为 $[H, W, D]$ 大小的三维数组。对上述三维数组中的每一个二维图片进行处理，提取出每一个二维图片中的肺部部分，并将其保存在新的 $[H, W, D]$ 大小的三维数组中，即可得完成分割后的三维肺部模型。具体代码如下所示。

1.1　文件读取

利用 *load_nii* 函数进行文件读取操作。读取的数据为一结构体，访问其 img 属性即可得到 $[H, W, D]$ 大小的三维数组。其代码如下所示。定义 *seg_lung* 和 *acc_lung* 变量分别保存人工分割 (Ground Truth) 和代码分割的肺部三维数组。

```
ori_lung = load_nii(['./data/coronacases_org_00', pic_name(i), '.nii']);
gt_lung = load_nii(['./data/coronacases_lung_00', pic_name(i), '.nii']);
img = ori_lung.img;
gt = gt_lung.img;
[H, W, D] = size(img);
seg_lung = zeros(H, W, D);
acc_lung = zeros(H, W, D);
```

1.2　二维图像预处理

逐层进行二维图像处理。由于 nii 格式存储的图片数据范围与一般的图片不同，在 Matlab 中无法正常显示。因此，需要对原图和人工分割后的二维图像首先应用 *im2double* 和 *Normalize* 操作，将其数据范围 *scale* 至 $[0, 1]$ 范围内。对人工分割图像，只需要利用 *imbinarize* 函数，并设定分割阈值为 0.1 即可将其分为正确的二值图像。对原图，利用 *graythresh* 函数获取自适应分割阈值，同样可分割出有效区域的大致轮廓。具体代码如下所示。

```
img_process = img(:, :, depth);
gt_process = gt(:, :, depth);
```

*liuzuyan19@mails.tsinghua.edu.cn

```
img_process = im2double(img_process);
img_process = Normalize(img_process);
level = graythresh(img_process);
img_process = imbinarize(img_process, level);

%Ground Truth图像二值化
gt_process = im2double(gt_process);
gt_process = Normalize(gt_process);
gt_process = imbinarize(gt_process, 0.1);
acc_lung(:, :, depth) = gt_process;
```

该步骤运行结果如图1、2所示。可以看到，已经提取出了 CT 图像中有效的人体区域，其中，肺部有效部分为黑色，其余无效部分为白色。

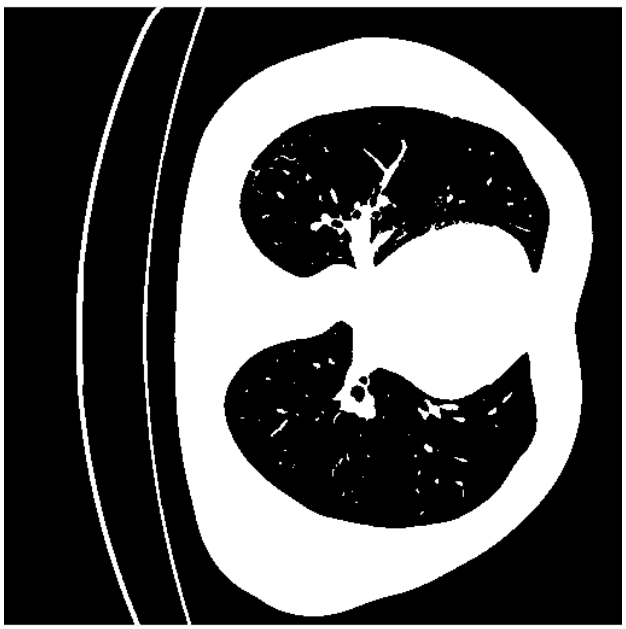

图 1: 待处理图像预处理结果

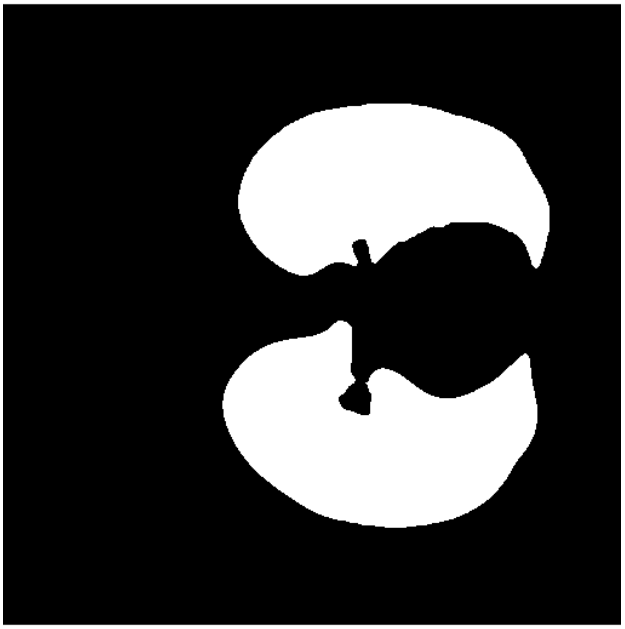

图 2: 真实值预处理结果

1.3 提取有效部分

利用上述分割结果，可利用形态学运算大致提取出有效部分。

观察到有效区域在大片白色人体区域内，因此，可首先找到上述大片人体区域，并利用 $imfill$ 函数填充内部的肺部区域。显然可知，填充的部分即为我们所需的肺部区域。最后利用 $bwareaopen$ 函数消除小块噪声的干扰。具体代码如下所示。

笔者定义了 $findMax$ 函数用于寻找二值图中的最大连通域。具体算法为利用 $bwlabel$ 函数寻找图像中所有的连通域，并赋予其对应的 $label$ 值。遍历每个 $label$ 值的面积，统计面积最大值，并返回对应的 $Mask$ 数组即可。

```
maxRegion = findMax(img_process);
body = imfill(maxRegion, 'hole');
lung = body .* (~maxRegion);
lung = bwareaopen(lung, 32, 8);
```

```
5
6  function result = findMax(img)
7      [L, num] = bwlabel(img, 4);
8      maxArea = 0;
9      [H, W] = size(img);
10     result = zeros(H, W);
11     for regionIndex = 1:num
12         area = sum(sum(L == regionIndex));
13         if area > maxArea
14             maxArea = area;
15             result = (L == regionIndex);
16         end
17     end
18 end
```

该步骤运行结果如图3、4、5所示。可以看到，正确通过形态学运算提取出了内部的黑色部分。

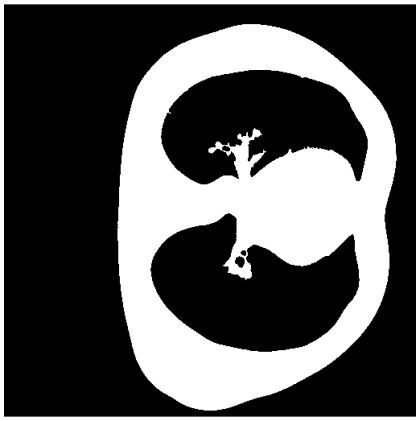

图 3: 最大连通域提取

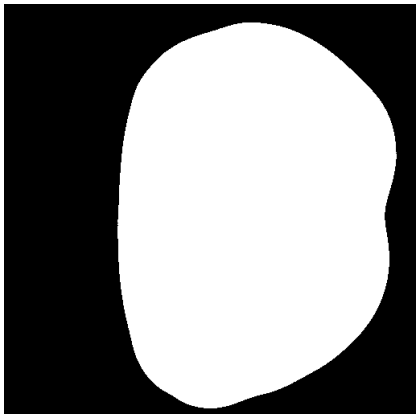

图 4: 图像填充

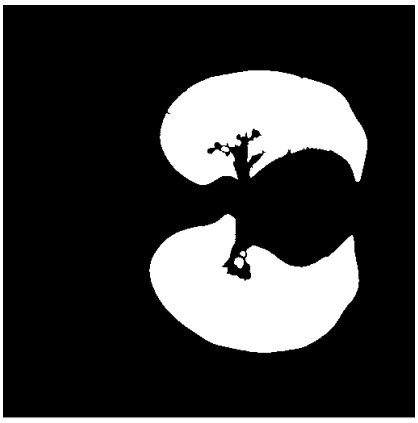

图 5: 有效区域提取

1.4 保留最大区域

由于肺部面积相对较大，因此，对上述二值图像，求取其中面积最大的两个区域，即为待求的肺部区域。具体算法与上一步骤类似。利用 $findMax$ 函数分别找到面积最大、面积次大的两个区域 A、B，则最终待求肺部区域可表示为 $A \cup B$。具体代码如下所示。

```
1 lungPart1 = findMax(lung);
2 lung = lung - lungPart1;
3 lungPart2 = findMax(lung);
4 lung = lungPart1 + lungPart2;
```

该步骤运行结果如图6所示。可以看到，仅保留两块大面积区域，其余小面积干扰区域均被去除。

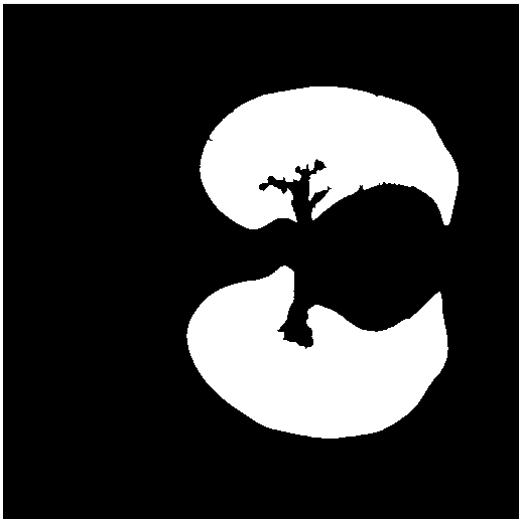

图 6: 最大肺部区域提取结果

最终，将分割结果赋值给 *seg_lung* 变量的对应层数即可。

1.5 三维连通域去噪

完成二维图像处理与合成后，需要对三维模型进行进一步处理。利用 *bwconncomp* 函数获取三维连通域 (参数 26 表示为三维图像)。遍历每一个连通域，判断其体积大小，若其体积大小小于最大值的一半，则将其直接舍弃。具体代码如下所示。

```
connectRegion = bwconncomp(seg_lung, 26);
areaPixels = cellfun(@numel,connectRegion.PixelIdxList);
maxArea = max(areaPixels);
for regionIndex = 1:connectRegion.NumObjects
    if areaPixels(regionIndex) < maxArea / 2
        seg_lung(connectRegion.PixelIdxList{regionIndex}) = 0;
    end
end
```

该步骤运行前后的三维图像如图7、8所示。可以看到，去除了小范围的三维区域，仅保留两块完整的肺部区域。

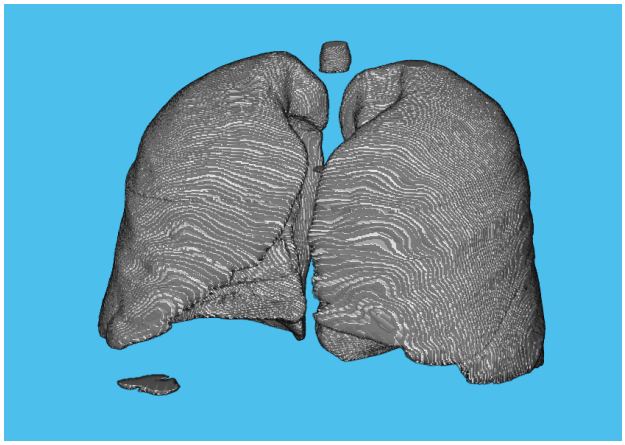

图 7: 三维去噪处理前

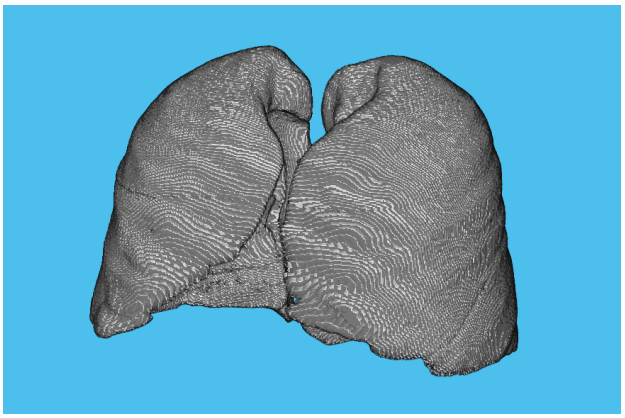

图 8: 三维去噪处理后

1.6 结果显示

显示结果及计算 *dice* 值较为简单，利用 *volshow* 函数显示三维结果，利用 *dice* 函数计算 *dice* 值即可。具体代码如下所示。

```
volshow(seg_lung);
calDice = dice(acc_lung, seg_lung);
disp(calDice);
```

最终结果可参看第三部分。

2 气管分割原理及代码分析

气管分割的基本思路与肺部分割相似，且前半段步骤相同。其不同点在于，气管分割不能通过统计最大两片区域得到分割后结果，需要通过另一类的方法将肺部大区域去除，仅保留气管区域。

文件读取、二维图像预处理、提取有效部分三步代码与肺部分割完全相同，并需要将保留最大区域步骤更改为如下去除肺部区域步骤。

2.1 去除肺部区域

由于气管部分与肺部部分相连区域较为复杂，可能有如图9、10、11所示几种情况，不能通过简单的连通域分割的方式去除。笔者采用多尺寸迭代的方法实现肺部区域与其他无效区域的去除。分别定义半径为 $1 \sim 9$ 的圆形元素，依次对其进行形态学腐蚀运算，尝试消除肺部与气管区域的连通域。对处理后的图像，去除其面积大于 500 的部分。利用该方法，若肺部与气管部分的连接成功消除，则肺部区域由于面积过大将会被去除，气管部分由于面积较小将会被保留。若肺部与气管的部分未被消除，则本次尝试未产生实际效果，在后续循环利用更大的圆形元素进行处理即可。最后，将半径从 $1 \sim 9$ 的处理结果进行并集运算，即可得到完成去除肺部后的图像。

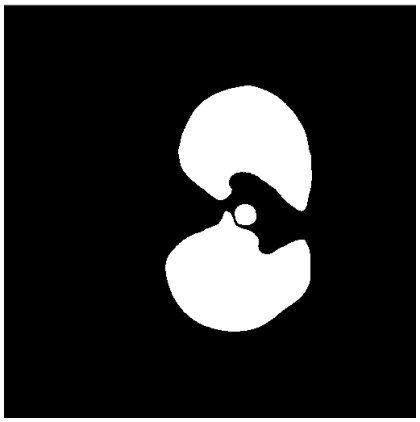

图 9: 气管与肺部分离

图 10: 气管为多个小部分

图 11: 气管与肺部相连

具体代码如下所示。

```
img_erode = zeros(H, W);
for disksize = 1:9
    img_erode = img_erode + multi_erode(trach, disksize);
```

```
end
trach = (img_erode ~= 0);
trach = bwareaopen(trach, 16, 8);

function result = multi_erode(img, disksize)
    se = strel('disk', disksize);
    img_ori = img;
    img = imerode(img, se);
    [L, num] = bwlabel(img, 4);
    for idx = 1:num
        if sum(sum(L == idx)) > 500
            img = (~(L == idx)) .* img;
        end
    end
    img = imdilate(img, se);
    result = img .* img_ori;
end
```

最后，利用 *bwareaopen* 函数去除小块噪声的干扰。

由于人工分割与代码分割的基准不同，需要对分割后的图像进行旋转与平移。代码如下所示。

```
trach = trach';
seg_trach(1:H-2, 1:W-1, depth - 1) = trach(3:H, 2:W);
```

该步骤运行前后的处理结果如图12、13、14所示。

图 12: 气管提取前

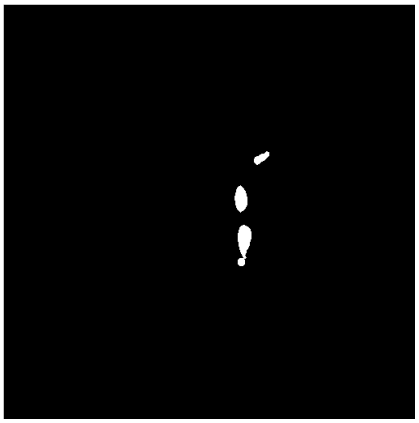

图 13: 气管提取后

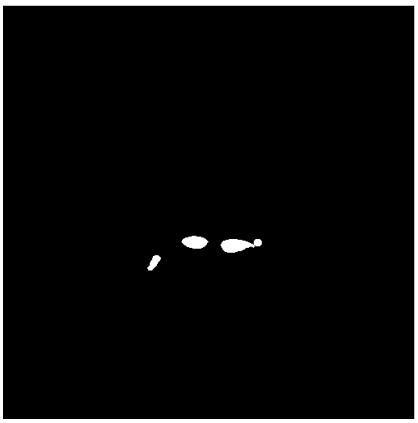

图 14: 图像旋转后

2.2 三维连通域处理

相比肺部分割代码，气管分割中的三维连通域处理有一定的不同。首先，需要利用 *imerode* 函数去除三维连通域中细小的连接部分，并利用与肺部分割类似的方法保留体积最大的连通域。最后，再利用 *imdilate* 函数将去除的部分恢复。具体代码如下所示。

```
se = strel('cube',3);
seg_trach = imerode(seg_trach, se);

```

```
connectRegion = bwconncomp(seg_trach, 26);
areaPixels = cellfun(@numel,connectRegion.PixelIdxList);
maxArea = max(areaPixels);
for regionIndex = 1:connectRegion.NumObjects
    if areaPixels(regionIndex) < maxArea
        seg_trach(connectRegion.PixelIdxList{regionIndex}) = 0;
    end
end

seg_trach = imdilate(seg_trach, se);
```

该步骤运行前后的三维图像如图15、16所示。可以看到，该步骤去除了大量的噪声区域，仅保留一块完整的气管区域。

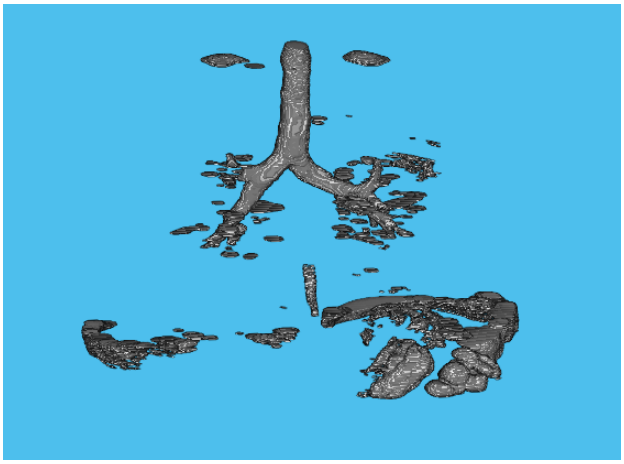

图 15: 三维去噪处理前

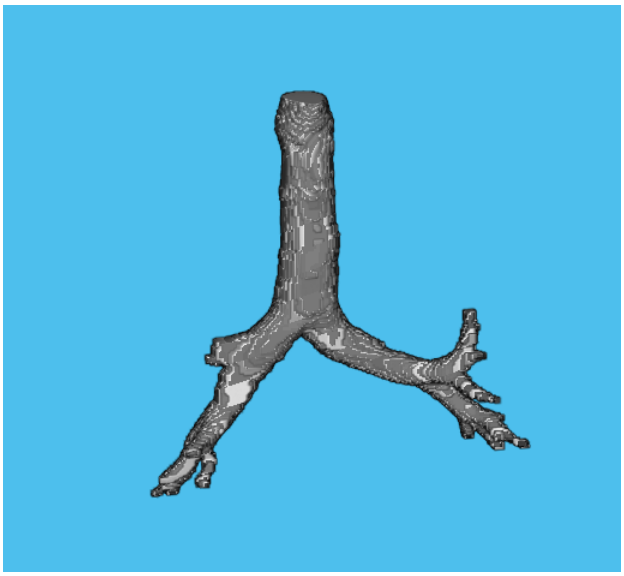

图 16: 三维去噪处理后

3 运行结果

3.1 肺部分割

肺部分割的三维图像结果与对应的真实结果分别如图17、18、19、20、21、22所示。

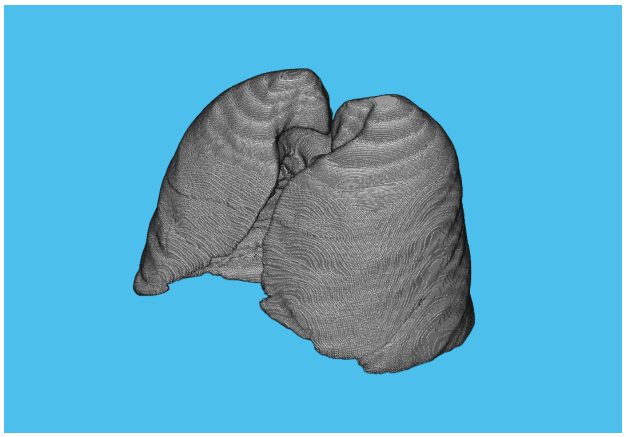

图 17: 程序生成分割图

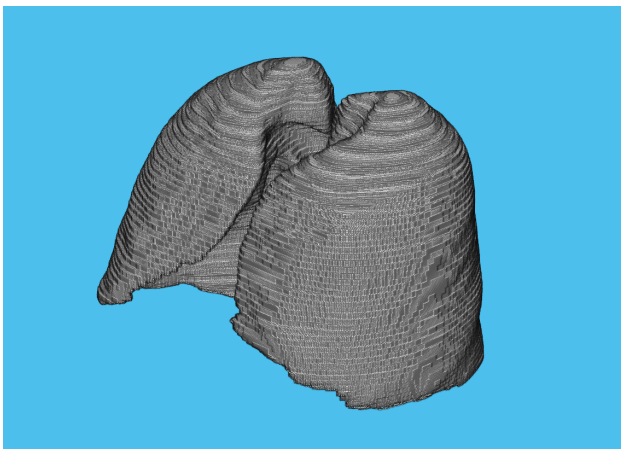

图 18: 正确分割

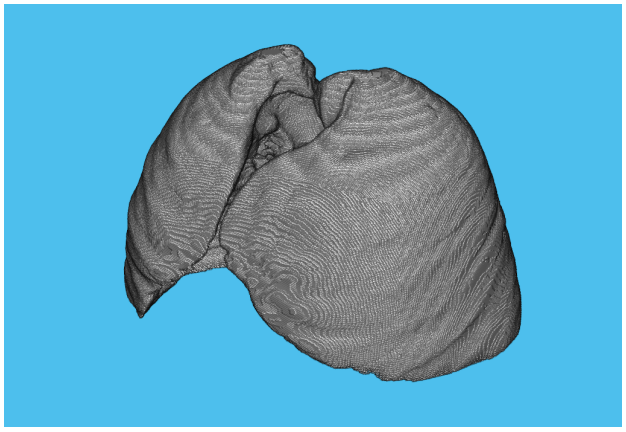

图 19: 程序生成分割图

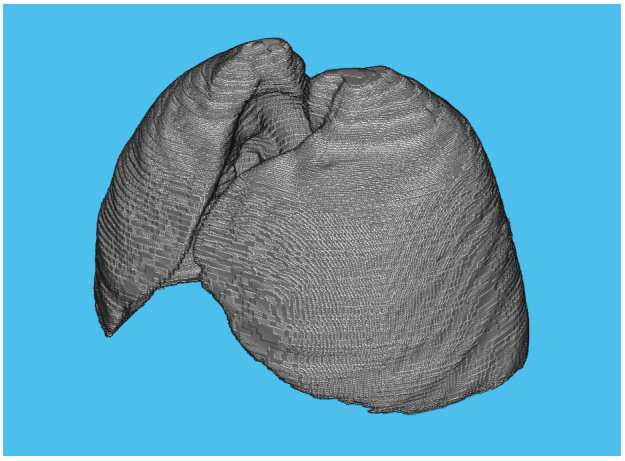

图 20: 正确分割

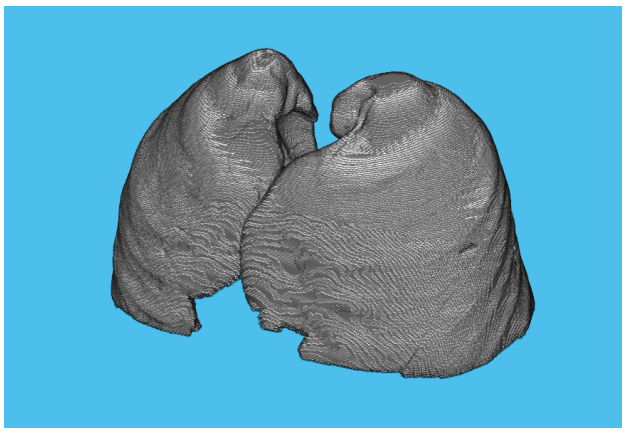

图 21: 程序生成分割图

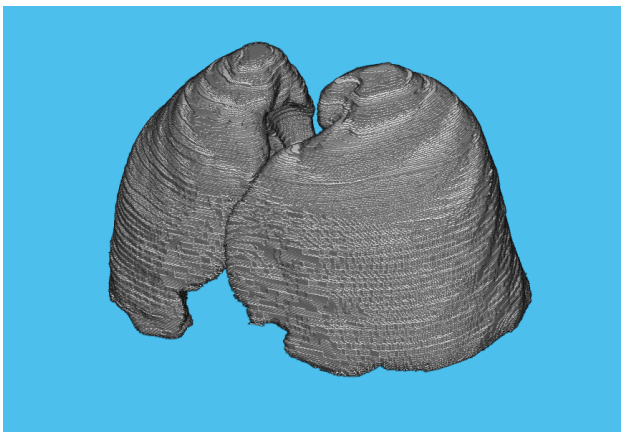

图 22: 正确分割

计算 *dice* 值分别为：

表 1: 肺部分割 dice 值

数据名	dice 值	dice 值 (无三维去噪)
004	**0.9845**	0.9835
005	**0.9859**	0.9837
007	**0.9682**	0.9617

从可视化结果与数据结果两个角度而言，均实现了较好的效果，程序分割结果与人工标注真实值极其接近。

3.2 气管分割

气管分割的三维图像结果与对应的真实结果分别如图23、24、25、26、27、28所示。

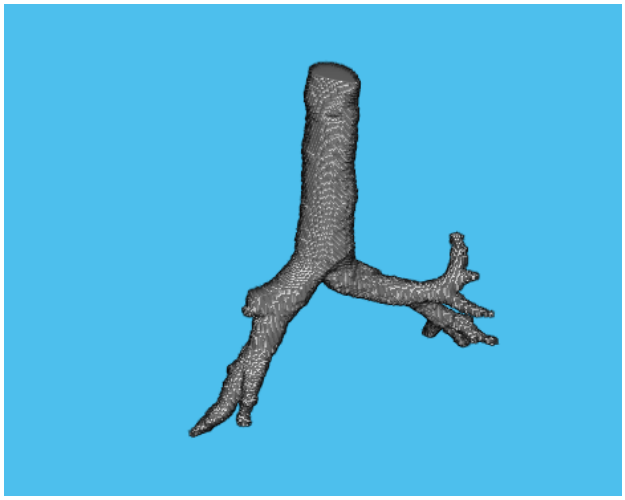

图 23: 程序生成分割图

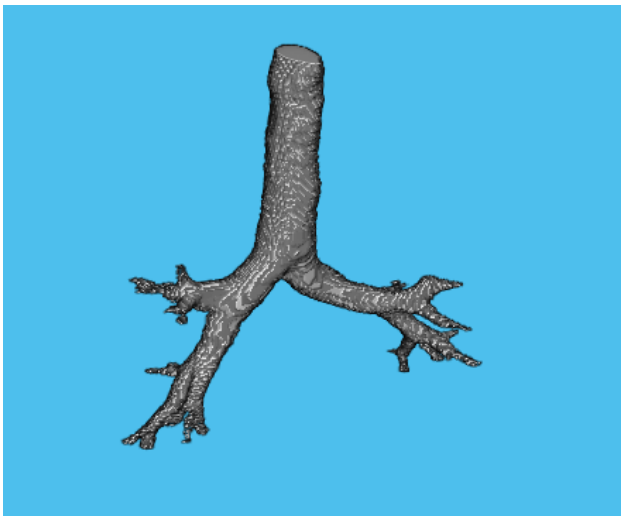

图 24: 正确分割

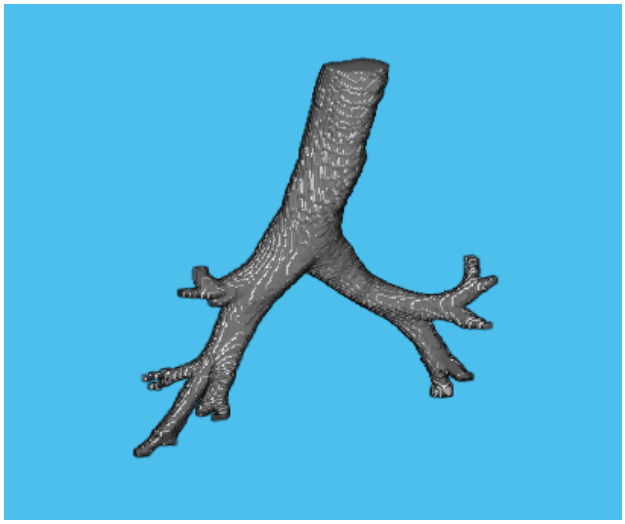

图 25: 程序生成分割图

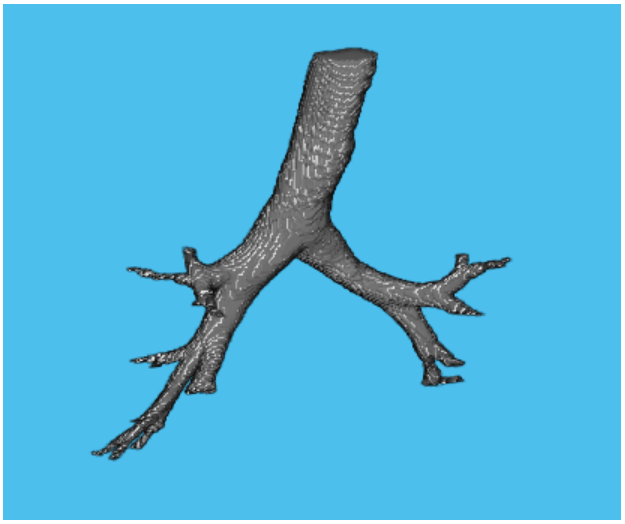

图 26: 正确分割

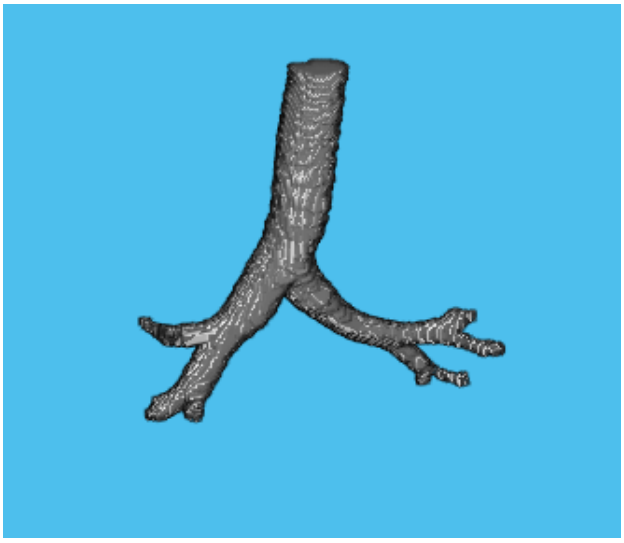

图 27: 程序生成分割图

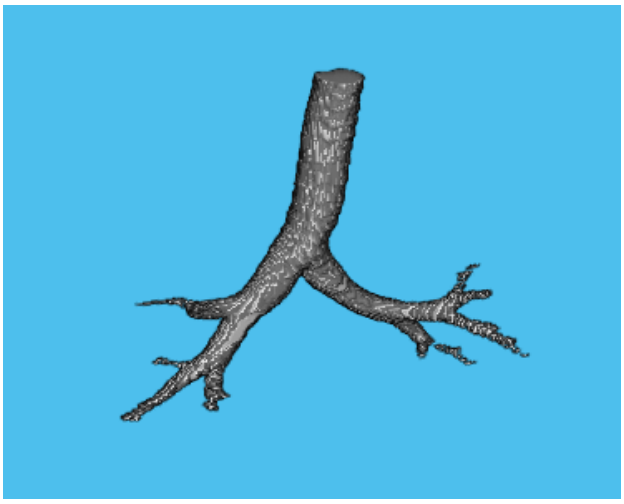

图 28: 正确分割

计算 *dice* 值分别为：

表 2: 气管分割 dice 值

数据名	dice 值	dice 值 (无三维腐蚀)	dice 值 (无三维去噪)
004	**0.9134**	0.8778	0.4314
005	**0.9293**	0.7725	0.3526
007	**0.9149**	0.7915	0.1970

从可视化结果与数据结果两个角度而言，均实现了较好的效果，程序分割结果与人工标注真实值极其接近。

4 遇到的困难及解决方案

由于此前在二维形态学处理方面已经积累了较多经验，故本次实验在二维图像处理方面没有遇到太多困难。由于此次实验涉及到了三维图像的处理，在这一方面笔者了解到了更多知识与相应的处理方法。概括而言，本次实验遇到的困难有以下几点。

- 图像预处理

  由于 nii 格式存储的图像与此前的图像的数据范围完全不同，直接利用 *imshow* 函数无法正确得到结果。笔者通过尝试发现这里同样可以利用 *im2double* 与 *Noramlize* 函数进行配合处理，此后即与此前的图像相同。

- 三维连通域去噪

  在肺部与气管分割中，三维去噪是提高准确率的重要方法。这是由于二维图像只能保存一层的信息，部分错误分割的区域很难通过二维图像进行判断。在三维图像中，增加了层与层之间的关系，从直观

的角度而言，会出现孤立的小块。此时，即可以通过与二维图像相似的处理方法对三维的连通域进行去噪处理。经过实验可知，该方法实现了较好的效果。

- 气管分割提取方法

  本次大作业中，气管分割的难度明显高于肺部分割，其准确率也明显低于肺部分割。这是由于气管的体积较小、有分叉、且很可能与肺部有连接，因此不能通过简单的连通域面积等判断方式。笔者尝试了定义不同尺寸的元素，并分别进行形态学腐蚀的方法，尝试将气管与其他部分分割开，并去除大面积的肺部部分，经过可视化结果判断，该方法取得了较好的效果。

- 消融实验的重要性

  由于本次实验涉及到较多算法优化的内容，因此，消融实验在验证某模块或某参数的有效性上具有重要作用。

5 收获与心得体会

- 普适算法的重要性

  此前几次数图作业，笔者花费大量时间对图片进行调参。然而，此次作业中，老师规定不能对每个图片设定特定的参数。这引导我们不断优化算法，并找到一个对各种数据都适用的解。在调节某个参数或加入某个模块后，可能使得某一张图结果更好，但也可能使得另一张图结果更差。这启示我们应当专注于算法本身，专注于寻找到真正有用的优化方法。

- 三维图像处理的基本方法

  这是笔者首次对三维图像进行处理。尽管其图像较为复杂，但基本的处理方法与流程与二维图像基本相似，三维图像既可以转化为二维图像逐层处理，也可以对其进行整体处理，可能会取得不同的效果。

- 可视化结果的重要性

  本次作业中，可视化结果是引导我们改进算法的重要途径。通过对二维图像、三维图像进行可视化，并与正确值相比较，我们可以较为清楚地知道具体的改进方向，改进的结果也可以通过可视化图像进行进一步验证。这启示我们在以后的作业、科研过程中，应当尽可能进行可视化验证，一方面可以验证代码的正确性，另一方面也可以通过观察引导改进的方向。

6 可能的改进方向

- 优化气管在细分支部分的准确率。
- 尝试利用形态学处理以外的方法提升准确率。