# 肝脏段切割程序文档

## 图形用户界面, 文本, 应用程序, 聊天或短信 描述已自动生成程序组成

\*\*src内部包含了所有主要文件

主函数main，调用所有切割函数；

Cut为开头的多个切割函数，分别执行每一刀的操作；

Liver\_cut.cpp 包含切割时需要的所有函数

Skeletonize.cpp 模块化后的血管骨架化

\*\*data内部包含了需要用的血管STL

\*\*new\_stl内部包含了输出后的所有分段STL

\*\*blew\_up内部包含了炸裂后的所有分段STL

\*\*两层CMakeLists，可运行文件的索引在内层

\*\*其他文件均为测试用文件

## 主函数

读取全部所需图示

描述已自动生成stl，依次序调用切割函数

**切割函数的次序不可随意更改**

文本

描述已自动生成具体原因见liver\_cut模块说明

## 每一刀的切割函数，以cut\_tail为例

切割函数会导入上一步切割后肝脏的剩余部分继续切割图示, 示意图

描述已自动生成

调用liver\_cut提供的所有关键函数，具体函数说明见liver\_cut

## Liver\_cut.cpp

Liver\_cut模块包含了所有关键函数：

### find\_points（）

该函数功能为：接收两个vtkPolyData，寻找切割关键点

主要调用库：CGAL（Delaunay三角化、Vonoroi图）、Eigen、VTK

**下一页为原理讲解，如果不需要看可以跳过**

### Delaunay三维剖分在本程序中的意义

在寻找两个血管STL的中点的点云时，需要分别遍历两个STL，然后对两个血管最相邻的那一部分点对计算中点。但我们不得不面临一个问题：我们如何得知当前遍历点对是最相邻点对？。如果出现下图2的情况，那么得到的点云将包含大量噪声：

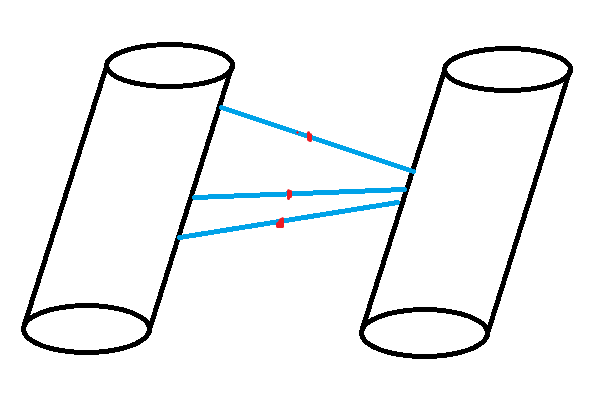


图1 我们需要的点对

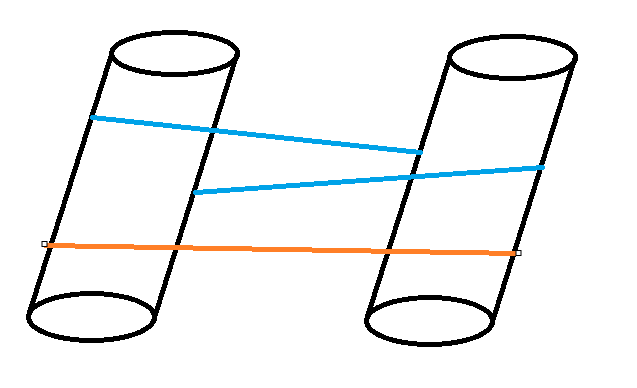


图2 我们不需要的点对

（蓝色为错误点对，橙色为不相邻的赘余点对）

通过三维剖分建立的四面体网格，我们可以解决这个问题，因为Delaunay三维剖分的结果只会包含相邻点组成的最优三角体。同时，三角体的数量远小于点对的寻找，我们可以通过给两个STL的点分别加上不同的标签，组成新的带标签的点结构体，随后合并两个点集，进行三维剖分，提取端点标签存在不同的四面体，这些四面体就是我们接下来要处理得到Voronoi对偶线段的四面体。同时，通过这种方式，可以避免大量的遍历线段，从而成倍的提高算法运行速度。

### Voronoi图

维诺图是由一组由连接两邻点直线的垂直平分线组成的连续多边形组成。具备以下特点：

     （1）维诺图把平面划分成n个多边形域，每个多边形内 只有 一个生成元；

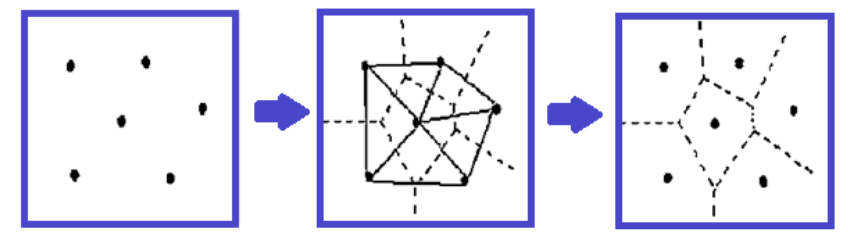
    （2）每个多边形内的点到该生成元距离短于到其它生成元距离；

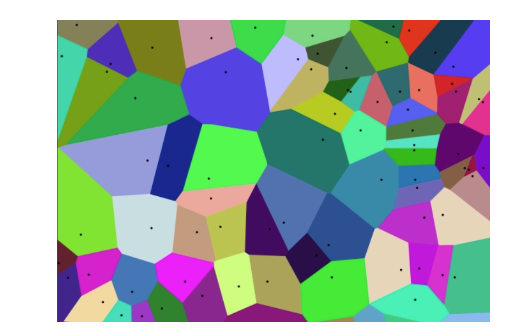
    （3）多边形边界上的点到生成此边界的两个生成元的距离相等；

    （4）邻接图形的Voronoi多边形界线以原邻接界线作为子集。

    （5）Voronoi图至多有2 \* n - 5个顶点和3 \* n - 6条边。

    （6）多边形内的生成元是形成三边的三点构成的三角形的外界圆圆心，而且所有的这些外界圆内部不含任何除这三点之外的顶点（空心圆特性）。





简单来说，维诺图的每个区域都是由相邻点的中垂线所组成。

**此外，维诺图是Delaunay三角剖分的对偶：**

二维中的Voronoi图和Delaunay三角化对偶性

1. Voronoi图的多边形单元：
   * 面在二维Voronoi图中就是所谓的Voronoi单元，它是一个多边形区域。每个Voronoi单元代表离某个种子点最近的区域（通常是一个不规则的多边形）。所以，在二维中，Voronoi的“面”其实是指这些多边形。
2. Delaunay三角形：
   * Delaunay三角化由一组连接种子点的三角形组成。这些三角形形成的网格是Voronoi图的对偶图。
   * 在Delaunay三角化中，三角形的每条边连接了两个相邻的种子点，而在Voronoi图中，这两个种子点对应的Voronoi单元由一条Voronoi边分隔。

二者为什么对偶？

Voronoi图和Delaunay三角化之间的对偶性反映了两者的几何关系：

* Voronoi顶点（多个Voronoi单元的交点）：对应于Delaunay三角形的外接圆心。这个圆心就是离三个Delaunay三角形的顶点等距离的点。
* Voronoi边：表示两个相邻种子点之间的等距离点，这些Voronoi边是连接两个Voronoi单元的边。对应于Delaunay三角化中的边，它连接了两个生成点。

因此，我们也可以把维诺图拓展到三维，来产生三维Delaunay四面体的对偶线段，这些线段的端点都是我们需要的中点，Delaunay三维剖分和Vonoroi图都借助CGAL库实现

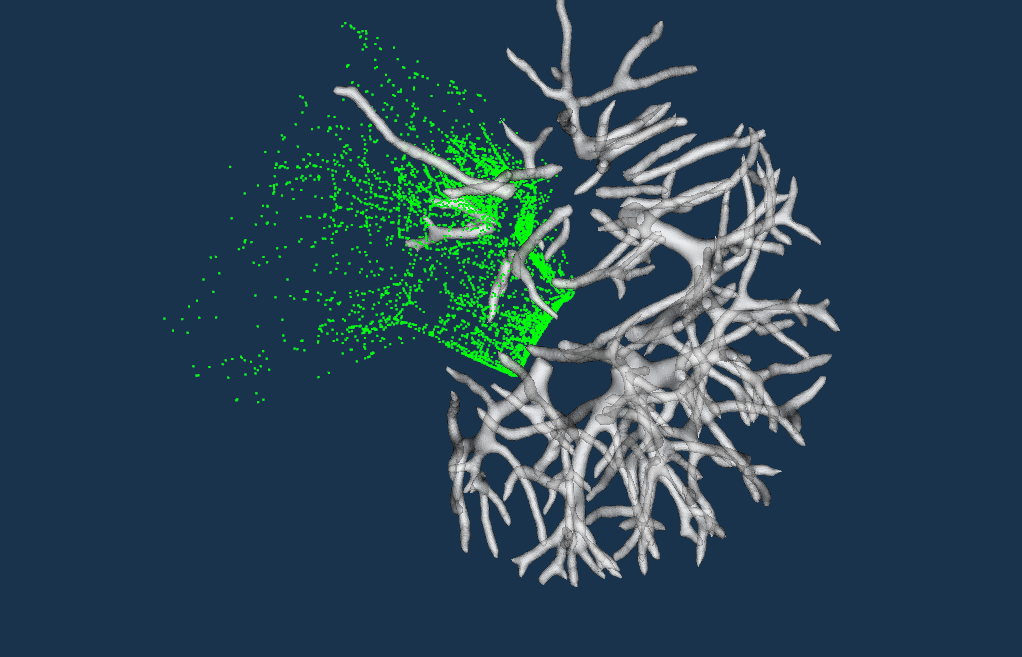
### Plane\_generator()

文本

描述已自动生成接收点集，使用vtkSurfaceReconstruction重建表面，输出隐含函数（注意看一眼类型）

### 表面重建遇到的困难：

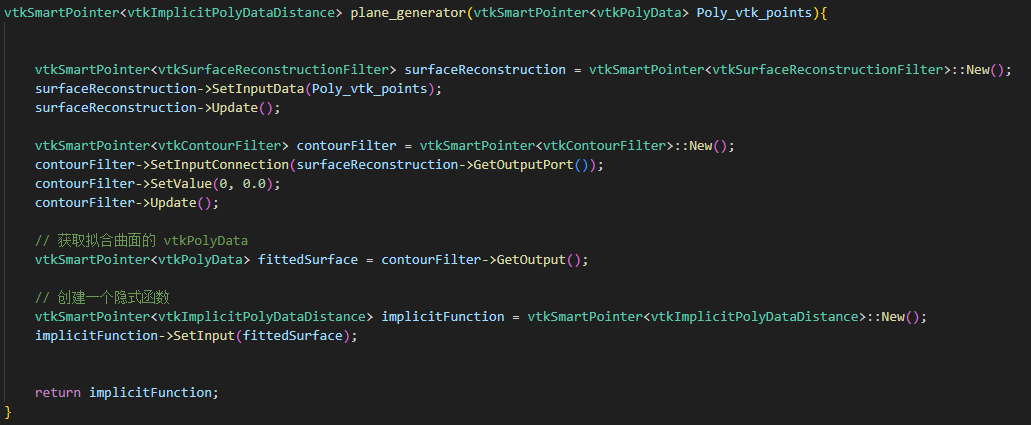
在试图拟合点云时，起初我使用的是多项式拟合，但发现多项式拟合并不能很好的捕捉点云的几何特性。例如，为降低切割时重复读取的STL数量，我希望第一刀先切割尾状叶段，但从点云特性来开，这是一个非常复杂的漏斗形，而二次多项式拟合只能非常粗略的描述这个表面，高次多项式则出现了收敛问题。其他方法还试过rbf重建和pcl库提供的泊松重建，但均因环境问题没能成功，尤其是pcl库，使用vcpkg安装编译后仍然调取不了某些库。



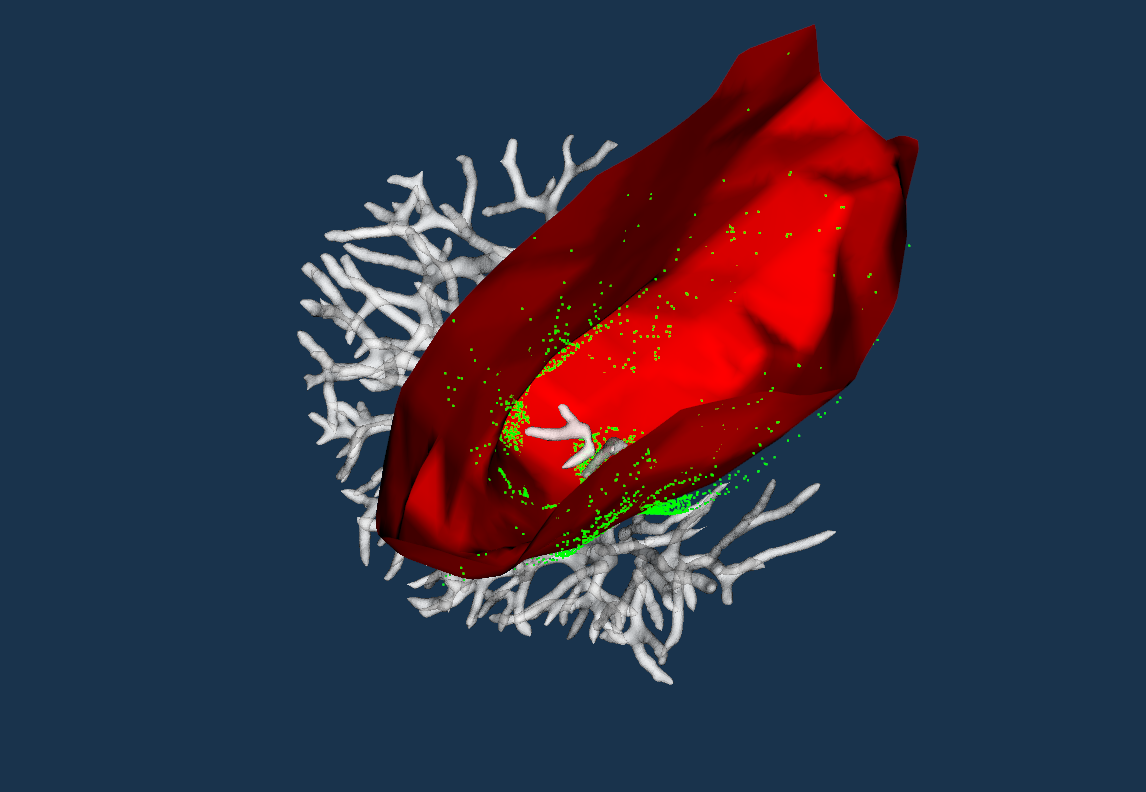
尾状段切割点云

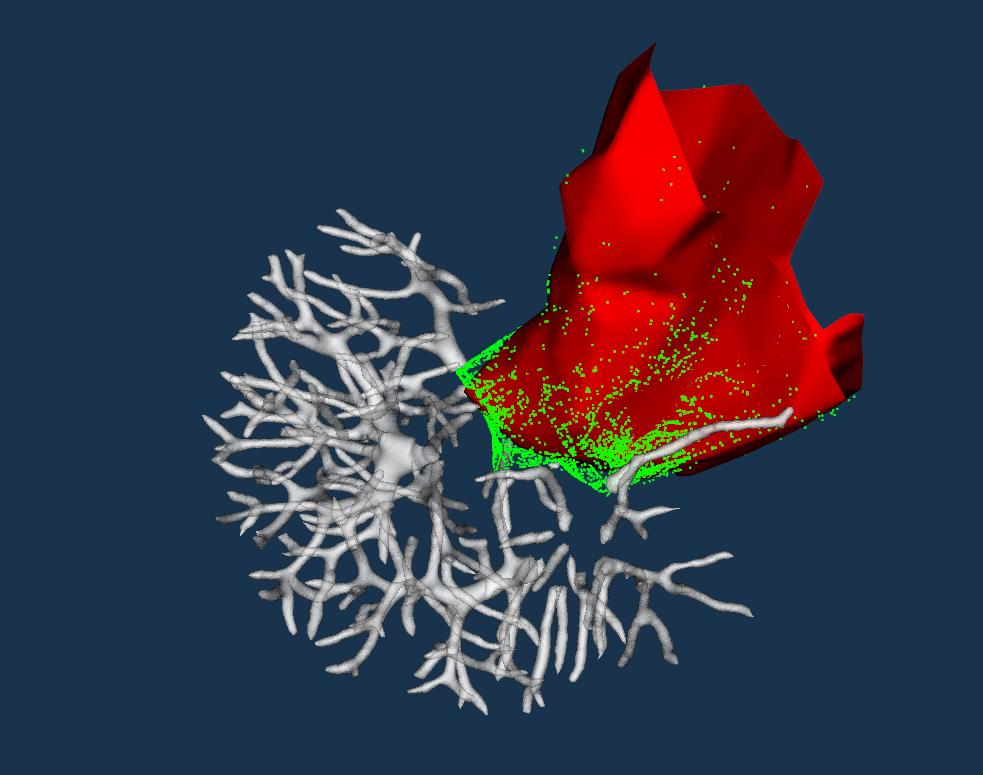
### 最终方案

最后我选择的方法是vtkSurfaceReconstruction库提供的表面重建函数，可以较为准确的重建切割表面，并且由于是vtk自带的库，不需要做多余的数据转换：

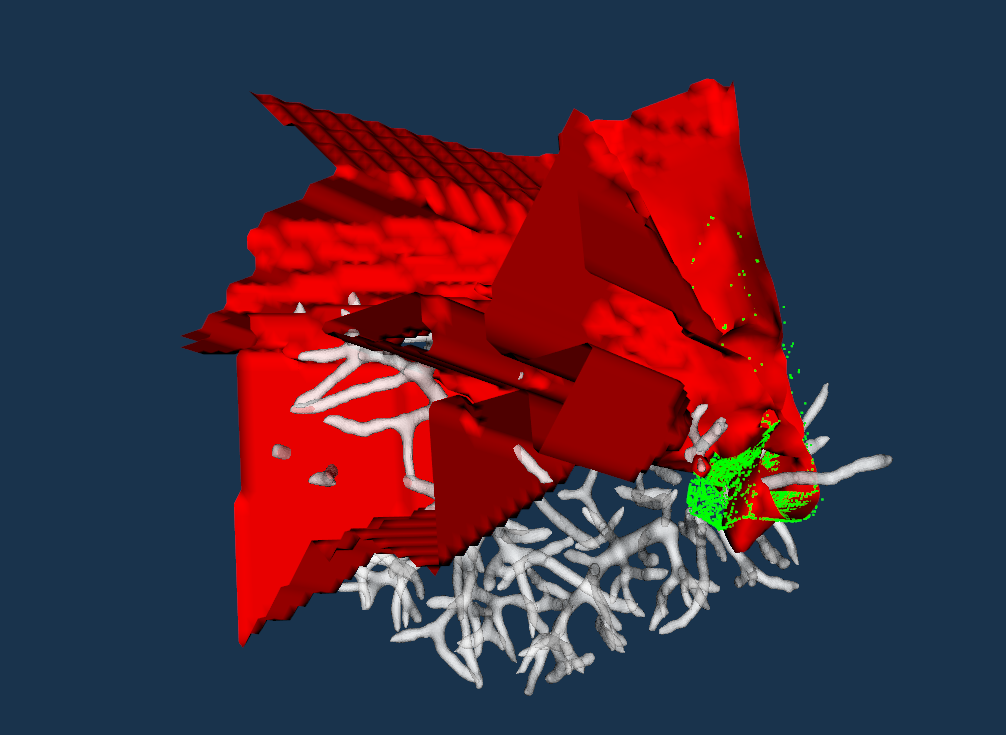


输出形式为隐式函数中的一种vtkImplicitPolyDataDistance，其基本思想为记录空间中一特定点到该表面上每个采样点的距离，最后可视化输出如下：





但仍然有一些复杂拓扑会导致该表面重建算法出现自环和不收敛的问题，例如单独切割左外叶上段的切割面：



**因此我在切割顺序上进行了安排**，每次切割后提取切下的部分和剩余的肝脏，以此忽略掉已切割的肝段，**避免出现单独切割时的复杂拓扑,尤其是右前叶下段的处理**。

### Cut\_run2()

图表, 箱线图

描述已自动生成接收隐含函数，使用ctkClipPolydata进行切割，输出切割后的两部分，剩余肝脏继续用于切割

目前封闭效果并不好，可能是因为原始数据有内表面造成的

图表, 雷达图

描述已自动生成

## 骨架化

骨架化调用了cgal库的平均曲率流提取，输出在txt中，仅输出了骨架点以及其相对应的曲面点

图片包含 游戏机, 烟花

描述已自动生成

树上的叶子

中度可信度描述已自动生成

骨架化效果图