快速3D建模原型系统

目录

[快速3D建模原型系统 1](#_Toc26744)

[简介 2](#_Toc16315)

[文档版本更新速览 2](#_Toc5050)

[1 功能 2](#_Toc14169)

[1.1近期 2](#_Toc9943)

[1.1.1近期功能描述 2](#_Toc23997)

[1.2远期 5](#_Toc11159)

[2 开发平台及编译方法 5](#_Toc8786)

[2.1 开发平台 5](#_Toc13690)

[2.2 编译方法 6](#_Toc3436)

[3 开发流程 6](#_Toc11013)

[3.1 CC开发规则 6](#_Toc5338)

[3.1.1 命名 6](#_Toc13996)

[3.1.2编译 7](#_Toc32181)

[3.1.3文件头 8](#_Toc969)

[3.2如何制作一个插件 8](#_Toc32243)

[3.3 算法库CCLib和qCC的参考文档 9](#_Toc20602)

[3.4 熟悉用户手册 10](#_Toc9031)

[4 注意事项 10](#_Toc25449)

[4.1 版权问题 10](#_Toc29271)

[4.2 可持续性 11](#_Toc11458)

[4.2.1 编写类参考文档 11](#_Toc2129)

[4.2.2 编写插件参考文档 12](#_Toc21474)

[5 数据库建设 12](#_Toc6750)

[5.1 仓库建设 12](#_Toc16268)

[5.1.1创建组织增加队员 12](#_Toc18829)

[5.1.2队员push自己的代码到主仓库 13](#_Toc21661)

[5.1.3分支 13](#_Toc14115)

[5.2 资料积累 15](#_Toc18333)

[5.3 算法库积累 15](#_Toc25725)

[5.4 点云和图片数据库 15](#_Toc32672)

# 简介

建设一个方便我们研究和使用的快速3D模型系统，如需要也可以支持商用。每一届的研究生或者本科生（毕设或者srp）都可以使用这个系统来做研究或者为这个系统做贡献。

# 文档版本更新速览

文档2.0：增加了第五章中的资料积累和算法库积累。

文档3.0：增加了编译cc的guide链接；

1.1.1功能实现简介

增加了5.4 点云和图片数据库的建设

目前正在代码实现的功能是1.1.1的(2). –吴优2018/10/9

增加了Github的组织创建并邀请成员

增加了Github的分支管理 —丁晓凤2018/10/9

# 1 功能

## 1.1近期

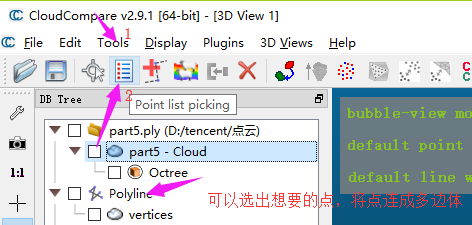
* 基本的点云交互功能
* 自动识别点云对应的3D模型（面表示）
* 渐进式合并点云。
* Ransac算法的改进
* 梁柱提取算法
* 开口门窗定位算法
* 类纱窗的提取算法
* 模型求交算法
* 多站点模型拼接算法？

### 1.1.1近期功能描述

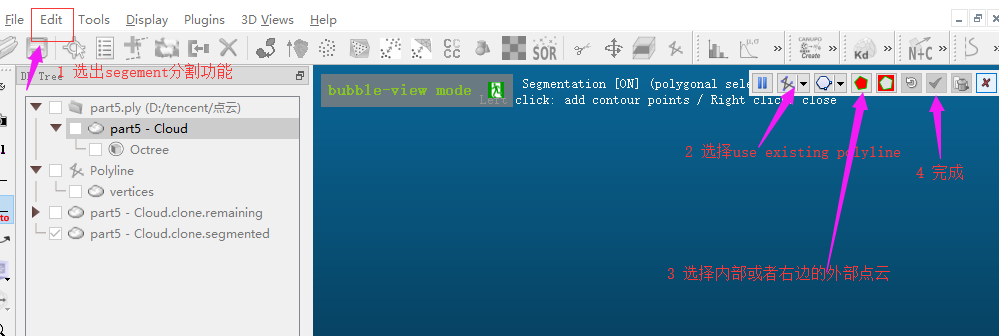
1. 基本的点云交互功能

* **能够显示和旋转点云；**
* **能够用某种工具部分选取点云，比如用一个正方体盒子把兔子点云的头部选取出来**：

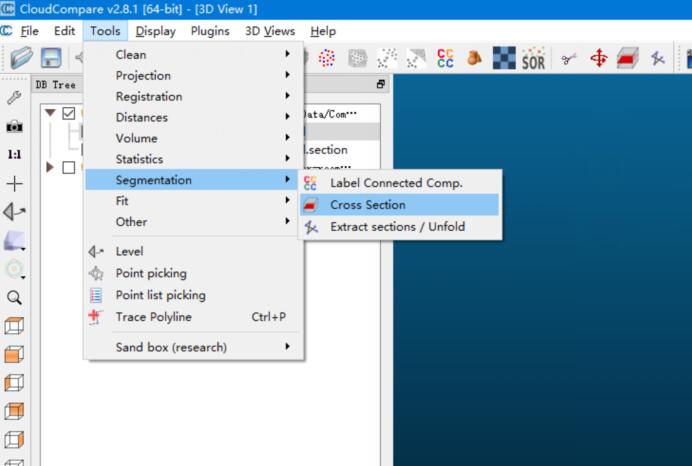
**a)**可以使用Tools中的Point list picking的功能，在点云上选择点，这些点可以生成多边体，也可以生成多边体的包围盒。

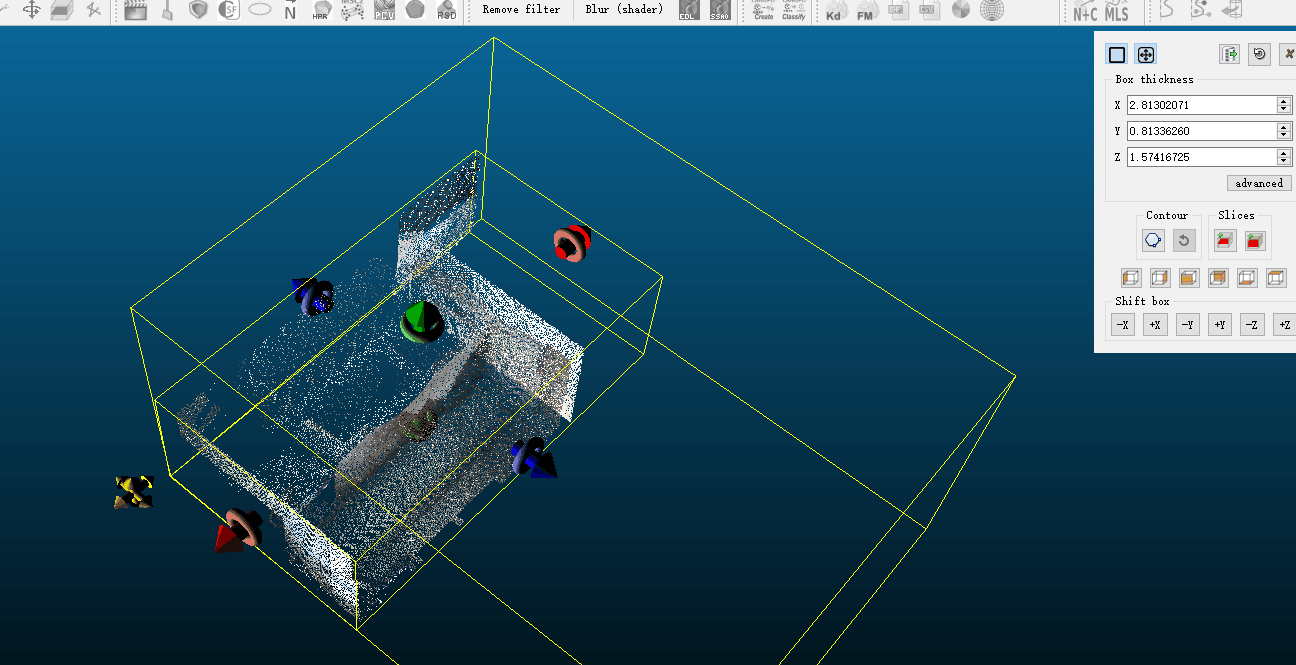


用包围体圈出来点云后，通过以下操作将它分割出来。



**b)**有一个更为简便的操作，移动下图中各个方向的箭头，可以将想要的部分切割出来，且生成一个新的点云。





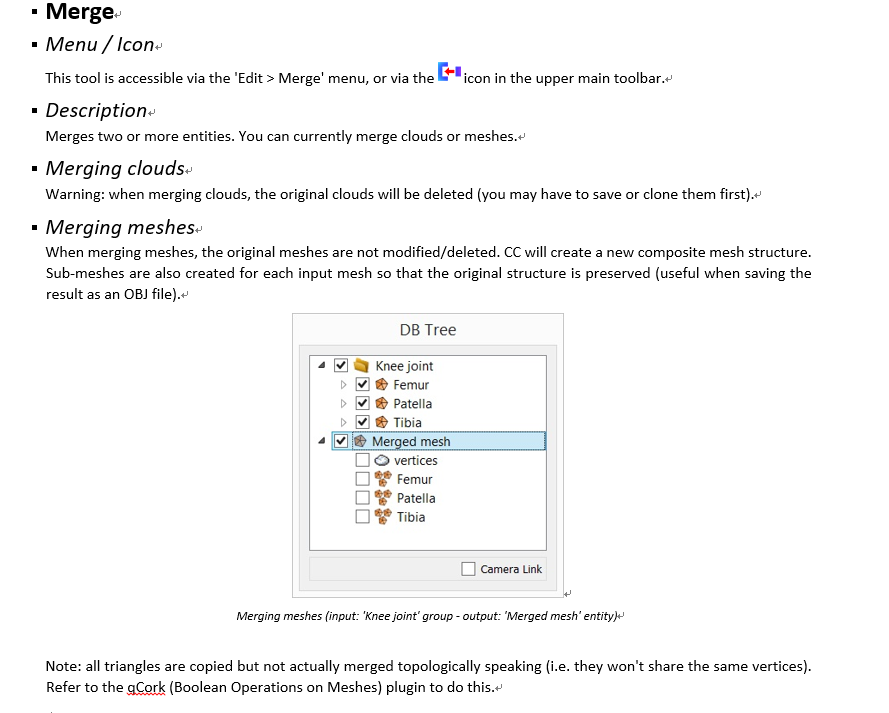
* **选择出来的点云会变成一个新的点云，因此能够对选择的点云进行修改**

（2）自动识别局部点云对应的3D模型（mesh表示）

利用改造后的ransac来识别局部点云，每次识别只给出一个最佳匹配的基元图形。基元图形选取通用的plane，torus，cone，cylinder，Sphere。**ransac算法里对基元图形的定义在拟合建筑上效果没有预期的好，这是一个可以改进的点。**由于ransac拟合出来的基元图形只是部分和原始点云有交集，因此需要对基元图形进行截取。例如我对一个墙面的点云进行拟合，结果出来一个plane的点云，这个plane远比这个墙面大得多，并不是我想要的墙面，所以需要进一步改造该算法。其实再增加一步操作就可以了，这个墙面点云有一个包围盒，我可以利用这个包围盒来截取出部分的plane，这样就完成了一个简单的墙面重建。重建后的点云要不要用mesh表示，目前先不考虑，CC自带的Merge功能可以合并点云，也可以合并网格，但是网格合并会不会有比较大的麻烦，这个还需后面好好考虑。

（3）渐进式合并点云

依次循环，获取第二个3D模型之后，能自动合并3D模型，这样就可以进行渐进式的建模。在edit功能下有个merge，可以实现模型合并。但是实际使用中可能涉及到点云对齐的问题。



## 1.2远期

* 纹理库的准备
* 纹理贴图算法
* 常用建筑物基本单元点云库的建立
* PointNet算法的集成
* Bim模型的生成，与常见建模工具之间进行数据交换。

# 2 开发平台及编译方法

## 2.1 开发平台

使用Windows版本的开源点云交互软件CloudCompare进行二次开发。用到的其他相关软件待补充。

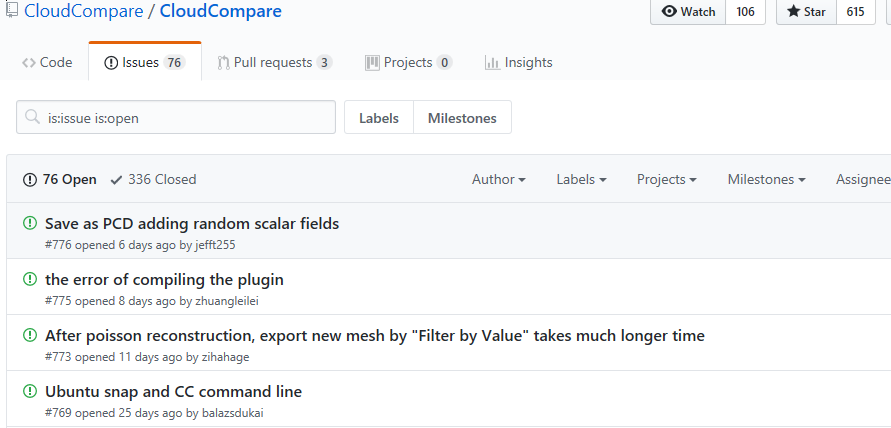
## 2.2 编译方法

<https://github.com/CloudCompare/CloudCompare/blob/master/BUILD.md>

（暂时沿用2017年7月编译的平台，cc2.7+，qt5.6，win10,vs2017）

# 3 开发流程

以插件的形式，在CloudCompare中加入相关的功能，还能拥有自己的界面。平时可以多阅读CloudCompare的github文档，https://github.com/CloudCompare/CloudCompare。其中的CONTRIBUTING.md介绍了CC的一些开发规则和注意事项。Github项目上的Issues分组里，是各位建设者对一些bug的讨论，在开发过程中遇到问题可以在这个板块参与讨论，以便寻求帮助。



以下是对CONTRIBUTING.md的一个简单翻译。

## 3.1 CC开发规则

### 3.1.1 命名

CloudCompare中的名称应该尽可能具有描述性，没有缩写，除非是非常明确或常见的（例如fw而不是forward等）。 大多数变量名称应以小写字母开头。 如果名称由多个单词组成，则每个单词的第一个字母应为大写（当然，第一个单词除外）。  
示例：numberOfPoints，ptsCount

**（1）特殊情况下：**

* 静态变量：应始终以前缀s\_开头（小写 - 如s\_defaultFilename
* 静态方法：应始终以大写字母开头（如InitGLEW）
* 类：应始终以前缀cc开头（小写 - 如ccConsole）
* enumerators:：

所有字母应大写  
应始终以前缀CC\_开头  
单词由下划线分隔（如CC\_OBJECT\_FLAG）

* 宏：以前缀MACRO\_开头，后跟标准方法名称（如MACRO\_SkipUnselected）
* const变量：  
  所有字母应大写  
  单词由下划线分隔（如NORMALS\_QUANTIZE\_LEVEL）
* macro const（#define）：应该避免; 与const的语法相同

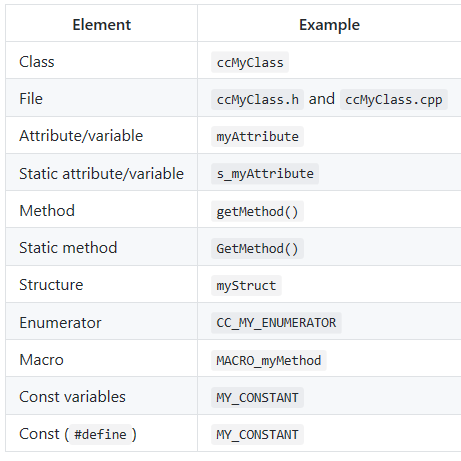
**（2）文件**

* 文件命名遵循与大多数CloudCompare元素相同的规则（小写的第一个字母等）
* 每个类应该单独保存在标头+源文件中。在特殊情况下，单个类使用的非常小的类可以与此类一起保存。标头+源文件名应与主类相同。

示例：ccConsole保存在ccConsole.h和ccConsole.cpp中

* 文件名不应包含任何空格字符，可以用下划线代替。
* 所有与数据相关的类（数据模型，数据库等）都应保存在db目录中。
* 图像（图标）应全部保存在images目录（或其子目录之一）中。
* GUI模板（主要是.ui Qt文件）应保存在ui\_templates目录中

示例小结：

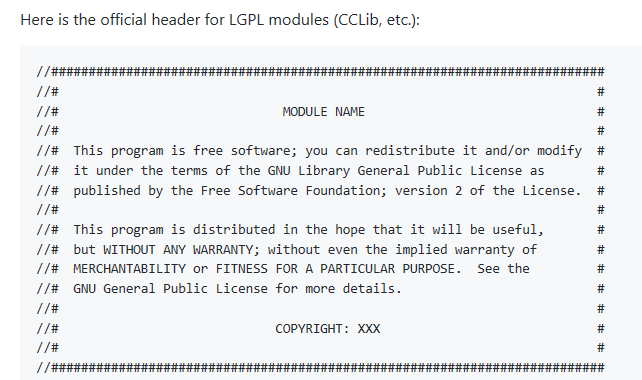


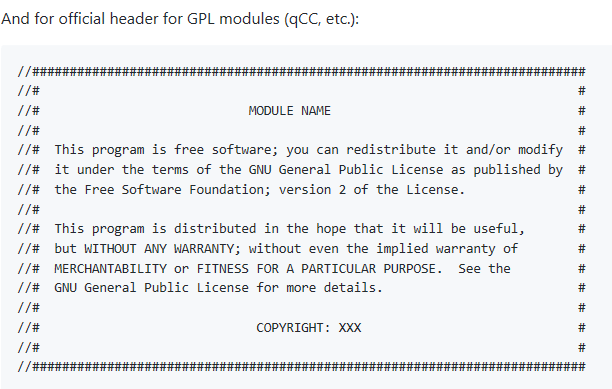
### 3.1.2编译

为了避免与Unix环境不兼容的语法，必须遵守以下规则：¬仅对包含路径使用“/”。  
示例：include“../db/ccPointCloud.h”

### 3.1.3文件头

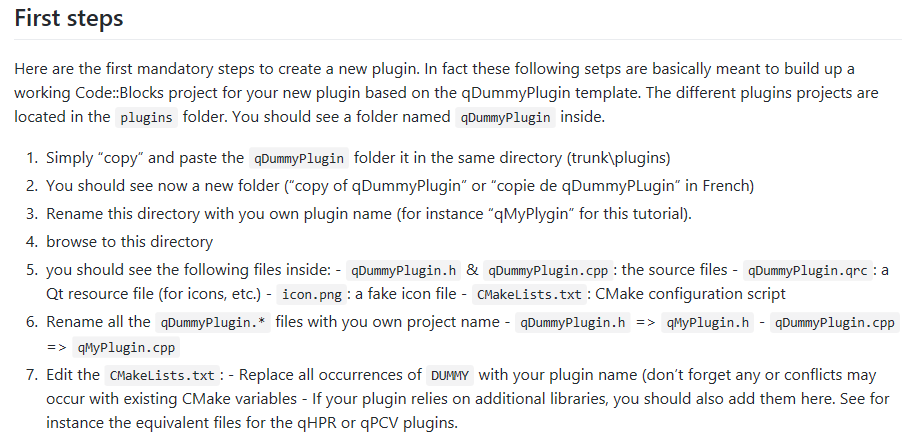
集成到任何CloudCompare模块（CCLib，qCC等）的任何新源文件（.h，.cpp等）都必须显示正式标头。使用时请到CONTRIBUTING.md文件下copy.





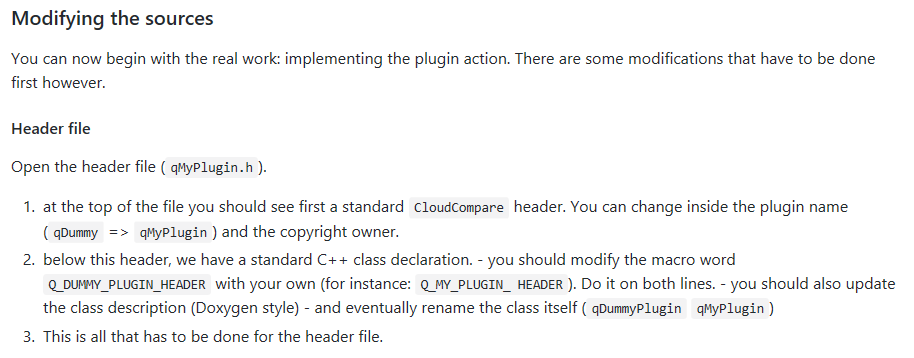
## 3.2如何制作一个插件

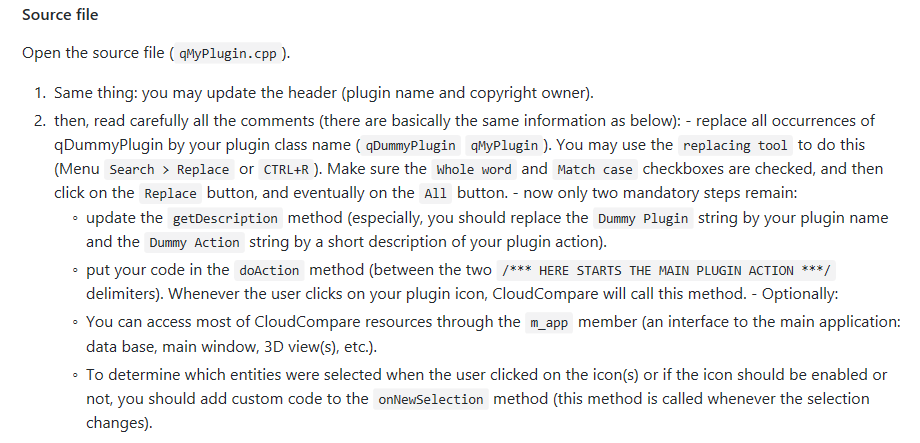
设计新插件是扩展qCC（CloudCompare）功能的简单方法，无需修改其核心并完成所有连接，就可以轻松设计一个新功能，该功能可以应用于当前在CloudCompare中加载的一个或多个实体。 而且，插件可以显示自己的对话框。 提供了一个虚拟插件结构（源代码和相应的Code :: Blocks项目）作为模板。



## 3.3 算法库CCLib和qCC的参考文档

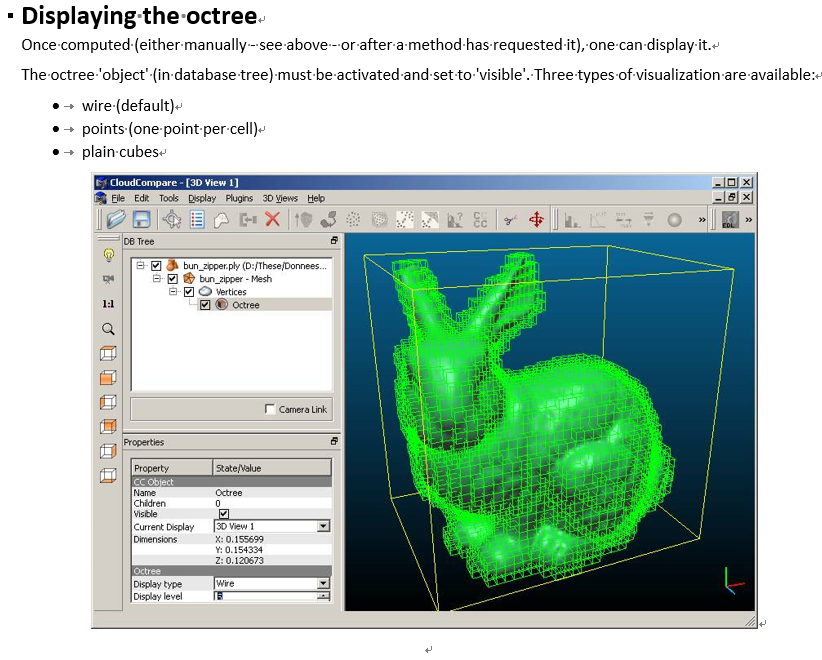
CCLib doxygen文档https://www.cloudcompare.org/doc/CCLib/html/index.html  
qCC doxygen文档 https://www.cloudcompare.org/doc/qCC/html/index.html





## 3.4 熟悉用户手册

CC在文件夹doc/en\_2.6.1 目录下有一个非常详尽的用户手册 CloudCompare User manual 2.6.1.docx，没事多翻翻，熟悉功能。



# 4 注意事项

## 4.1 版权问题

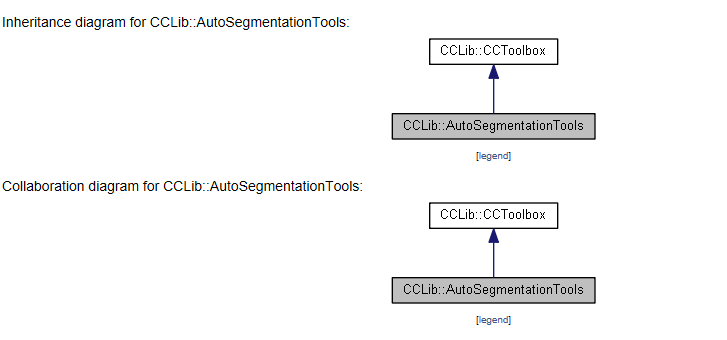
注意使用到的外部算法库是否能够任意使用，需要版权的就不太适合商用。

## 4.2 可持续性

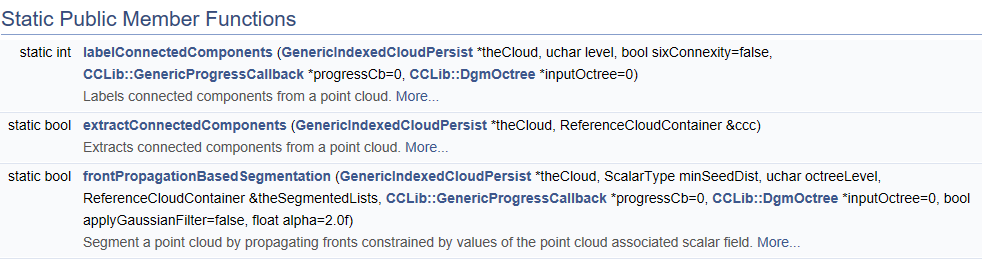
为了方便阅读和使用二次开发的代码，需要写参考文档。

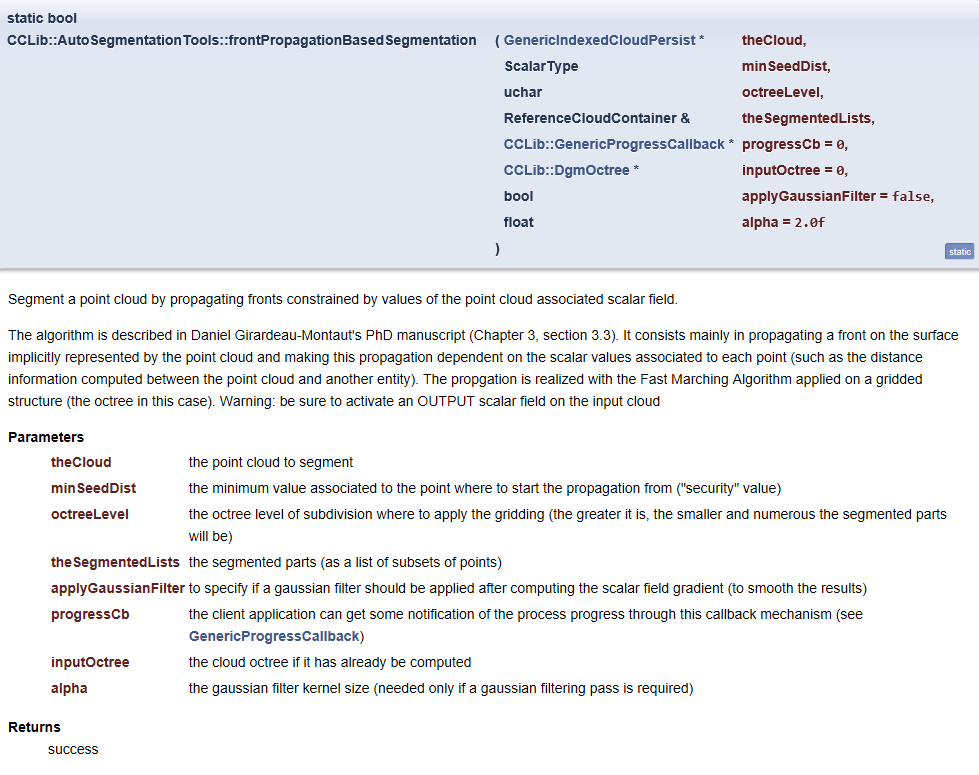
### 4.2.1 编写类参考文档

如果是增加了一个子类，格式参考CCLib的某一个类的reference，其中包括（1）继承示意图：



（2）对类成员的描述：





https://www.cloudcompare.org/doc/CCLib/html/class\_c\_c\_lib\_1\_1\_auto\_segmentation\_tools.html

### 4.2.2 编写插件参考文档

如果是增加了一个插件，要详细写出：

1. 这个插件使用了什么算法，简单介绍这个算法的思想，每一次更新都有什么改进。
2. 如何使用这个插件，注意多一点截图；

# 5 数据库建设

## 5.1 仓库建设

### 5.1.1创建组织增加队员

组织是共享帐户，企业和开源项目可以同时在多个项目中进行协作。所有者和管理员可以使用复杂的安全性和管理功能管理成员对组织数据和项目的访问权限。

1. 在任意页面的右上角，单击您的个人资料照片，然后单击“设置”；

（2）在用户设置侧栏中，单击“ 组织”；

（3）在“组织”部分中，单击“ 新建组织”；

（4）在“组织名称”下，键入组织的名称；

（5）在“结算电子邮件”下，键入应发送组织付费计划收据的电子邮件。

队员需要在Github上进行注册才可以被邀请到组织。

1. 在GitHub的右上角，单击您的个人资料照片，然后单击您的个人资料；
2. 在个人资料页面的左侧，在“组织”下，单击您组织的图标；
3. 在您的组织名称下，单击人；
4. 在“人员”选项卡上，单击“ 邀请成员”；
5. 输入您要邀请的人的用户名，全名或电子邮件地址，然后单击“ 邀请”；
6. 受邀人员将收到一封邀请他们加入该组织的电子邮件，他们需要在成为组织成员之前接受邀请。

### 5.1.2队员push自己的代码到主仓库

暂行方案：把组织成员的权限设置为“管理员”，这样组织成员就拥有了clone，pull，push权限，还能够添加新的协作者到所有储存库。

### 5.1.3分支

分支是存储库的并行版本。它包含在存储库中，但不影响主服务器或master分支，可以让我们在不中断“实时”版本的情况下自由工作。当我们觉得分支足够满足我们的要求时，可以将分支合并回master分支以发布更改。每个存储库都有一个默认分支，并且可以有多个其他分支。我们可以使用拉取请求将分支合并到另一个分支。分支的名称是自定义的。

我们可以使用分支来：开发功能，修复错误，安全地尝试新想法。

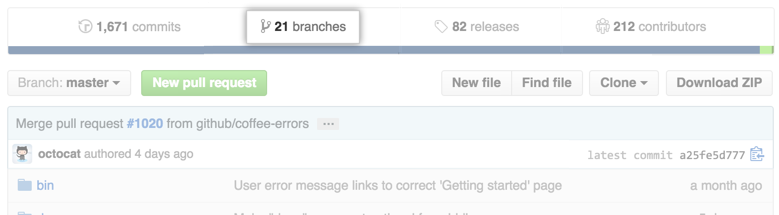
1. 分支的基本管理

A.设置默认分支（一般是master分支，暂时不需要修改默认分支）：默认分支被视为存储库中的基本分支，除非我们指定其他分支，否则将自动生成所有拉取请求和代码提交。

B.查看储存库中的分支：分支是GitHub上协作的核心，查看它们的最佳方式是分支页面。

1.在GitHub上，导航到存储库的主页面。

2.在文件列表上方，单击 NUMBER个分支。



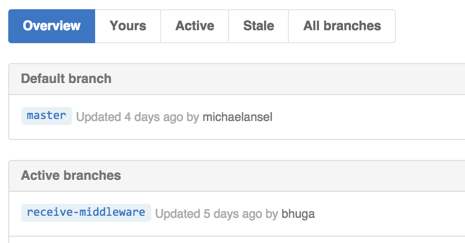
3.使用页面顶部的导航查看特定的分支列表：

****Your branches****：在你要推送权限库，在此致视图显示您第一次推到，与最近的分行所有分支。

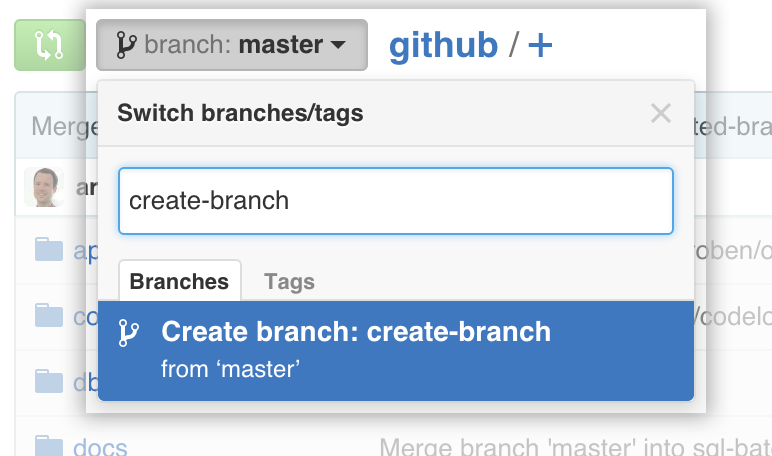
****Active branches：****活动视图显示过去三个月内任何人都已提交的所有分支，由具有最新提交的分支排序。

****Stale branches：****Stale视图显示过去三个月中没有人承诺的所有分支，由具有最早提交的分支排序。使用此列表确定[要删除的分支](https://help.github.com/articles/creating-and-deleting-branches-within-your-repository)。

****All branches：****All视图显示默认分支，后面是分支排序的所有其他分支，最先提交最新的提交。



C.在拉取请求中删除和恢复分支：如果在存储库中具有写访问权限，则可以删除与已关闭或合并的拉取请求关联的分支。但无法删除与打开拉取请求关联的分支。

创建分支：1.在GitHub上，导航到存储库的主页面。

2.单击分支选择器菜单。

3.为新分支键入唯一名称。

4.按Enter键。

删除分支：1.在GitHub上，导航到存储库的主页面。

2.在文件列表上方，单击  NUMBER个分支。

3.滚动到要删除的分支，然后单击删除图标。

1. 分支的作用：

master分支即默认分支（我们的主分支）一般都是稳定的代码，可以直接发布或者被项目之外的人使用。如果我们需要增加新的特性或者修复bug，我们就可以新建不同的分支，在不同的分支上进行开发和测试，测试完毕后可以向主仓库提交pull request请求。分支之间的互不影响这种特性可以增加团队合作的效率。

## 5.2 资料积累

（1）图形学的基础知识

（2）面向对象设计，类及类的继承

（3）C++

（4）

## 5.3 算法库积累

（1）RANSAC 平面拟合算法

（2）PCL ？

（3）CGAL ？

（4）Polyfit ？

（5）PointNet ？

## 5.4 点云和图片数据库

1 古建筑的点云数据

2 网上的数据

3 给房屋不同角度下拍照的数据