

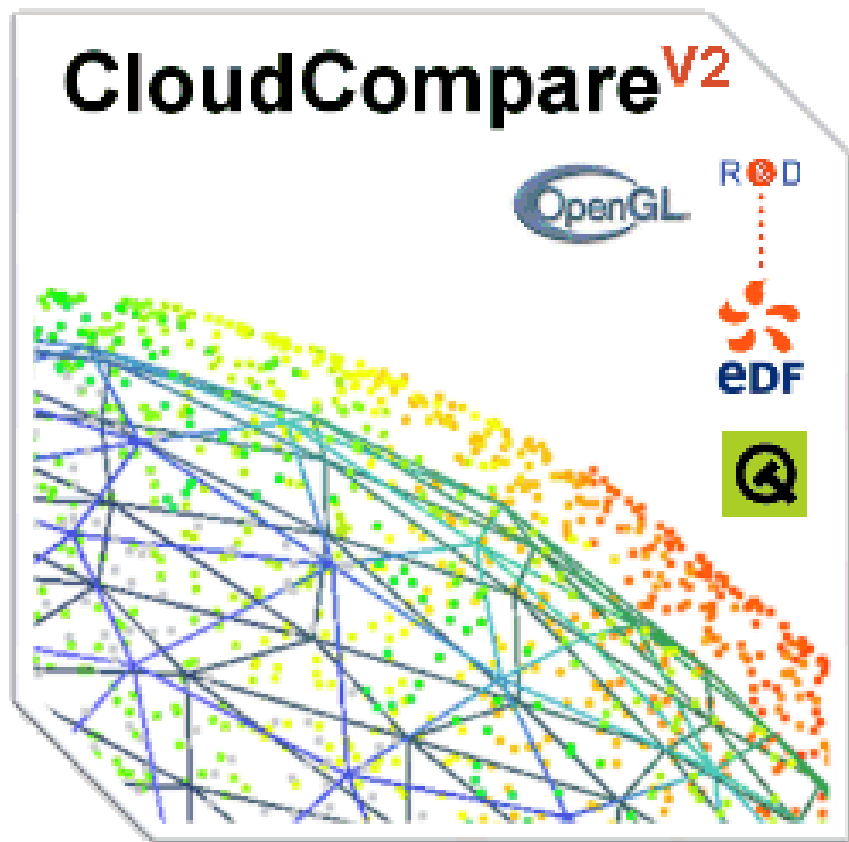


10分钟学会CloudCompare二次开发

分享人：李子宽

河海大学测绘系

日期：2020年6月19日



目录

Contents

1. 目前点云处理软件总结
2. CloudCompare 简单介绍
3. CloudCompare 插件展示
4. 插件实现原理
5. 实践

目前点云处理软件总结

商用软件:

- 1、基于Microstation平台开发的TerraSolid (国内外航测部门主要在用)
- 2、Trimble公司的Realworks (仪器配套)
- 3、Leica公司的Cyclone
- 4、Bentley公司的Pointtools
- 5、ENVI的 ENVI LIDAR (自动提取DEM植被等)
- 6、Geomagic (主要专注于机械零件点云)

国内科研院所或公司的软件:

- 1、武汉天擎的LiDAR Suite
- 2、西安煤航的LiDAR-DP
- 3、中国科学院遥感与数字地球研究所 点云魔方 (布料滤波)
- 4、北京数字绿土的LiDAR 360 (林业资源调查)

目前点云处理软件总结

开源软件：

PCL： C++编写，有python接口， BSD授权方式， 能实现点云相关的获取、滤波、分割、配准、检索、特征提取、识别、追踪、曲面重建、可视化等，但依赖库太多。

CloudCompare： C++编写， GPL授权， 可拔插操作， 可二次开发与源程序互不影响， 富有活力， 经常更新。

Meshlab： C++编写， GPL授权， 主要针对于模型， 重建和过滤， 网格简化等。

Open3d： C++编写， 有python接口， MIT授权， 支持快速开发和处理3D数据。

CloudCompare简单介绍

CloudCompare是一种3D点云（和三角网格）处理软件。设计出发时用于在两个密集的3D点云（例如用激光扫描仪获得的云）之间或在点云和三角形网格之间进行比较。它依赖于专用于此任务的特定八叉树结构。后来扩展为更通用的点云处理软件，包括许多高级算法配准，重采样，颜色/标准/标量字段处理，统计计算，传感器管理，交互式或自动分段，显示增强等。最新一次更新在2020年6月15日出了2.11.0版本，是非常有活力的一款软件！



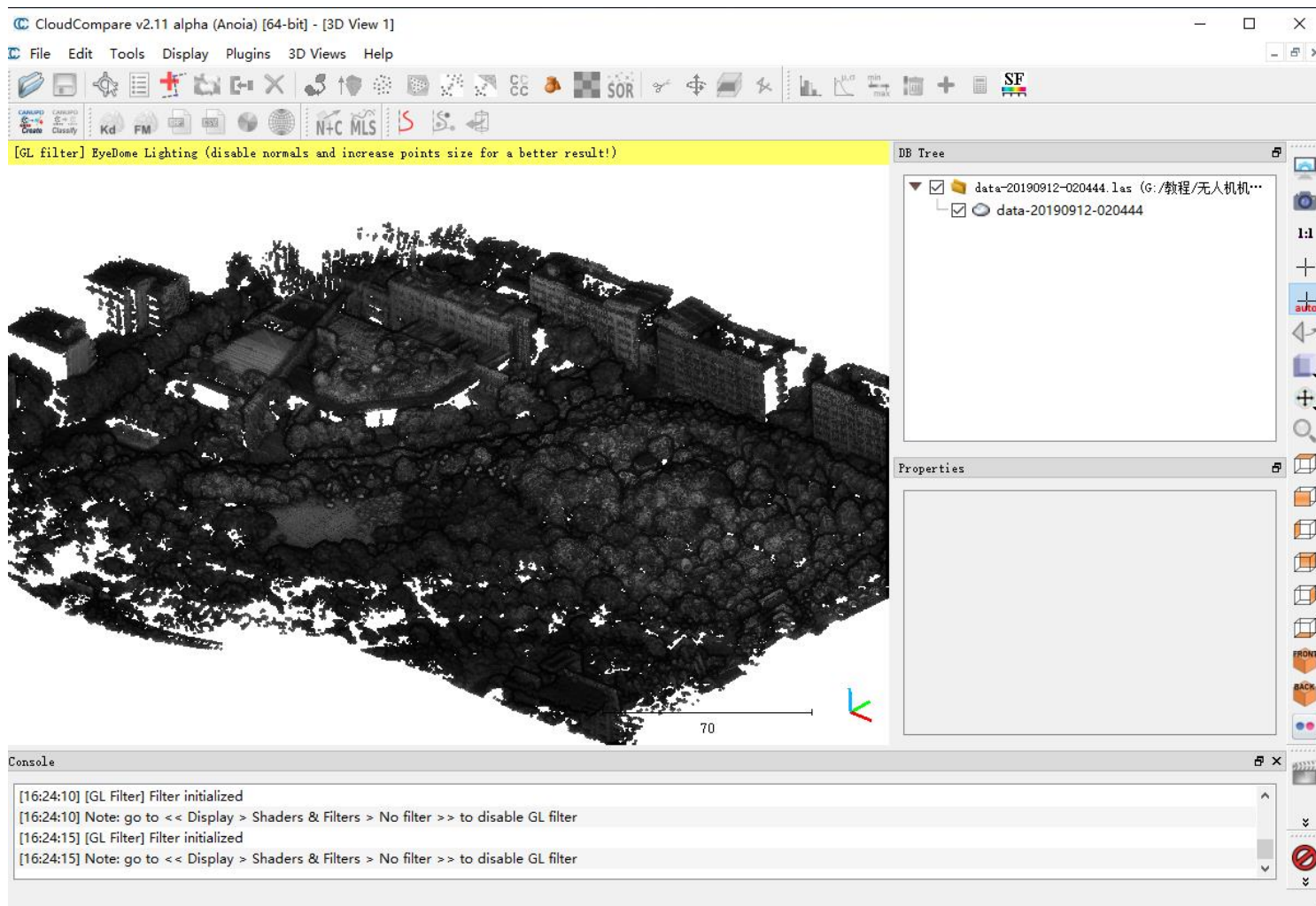
Daniel Girardeau-Montaut

<http://www.cloudcompare.org/>

支持的数据格式

可以加载许多开放式云格式（ASCII，LAS，E57等）以及某些制造商的格式（DP，Riegl，FARO等）。它还可以加载三角形网格（OBJ，PLY，STL，FBX等）和一些折线或多边形格式（SHP，DXF等）。支持一些SfM格式（Bundler，Photoscan PSZ等）。

为什么要学习cloudcompare二次开发



本教程适合人群:

- 1、对算法进行研究, 且有很高的计算效率要求
- 2、有自定义的可视化的需求
- 3、工程项目

优势:

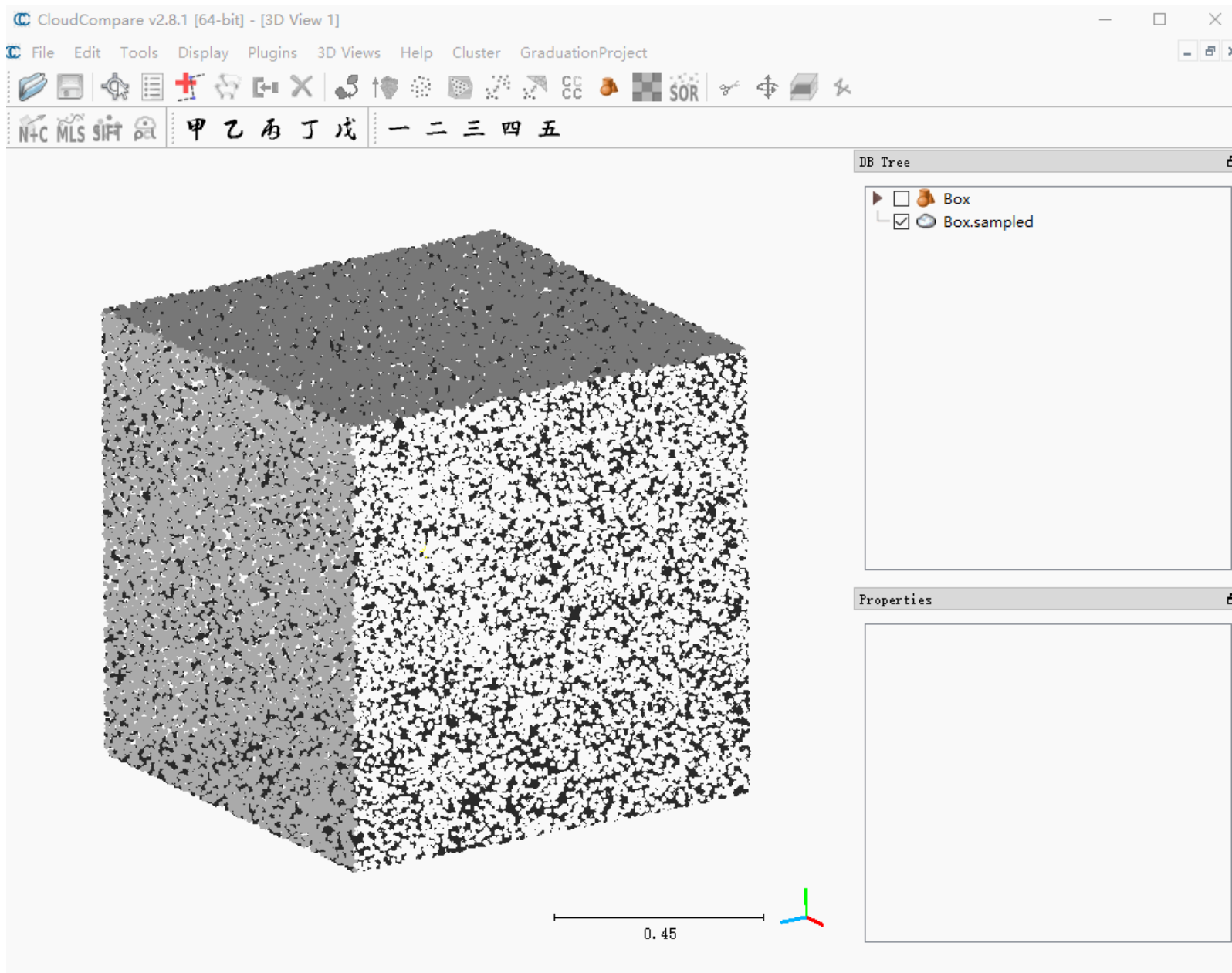
- 1、实时可视化
- 2、插件不影响主程序

缺点:

C++ 学习成本较高, 文档与教程都很少

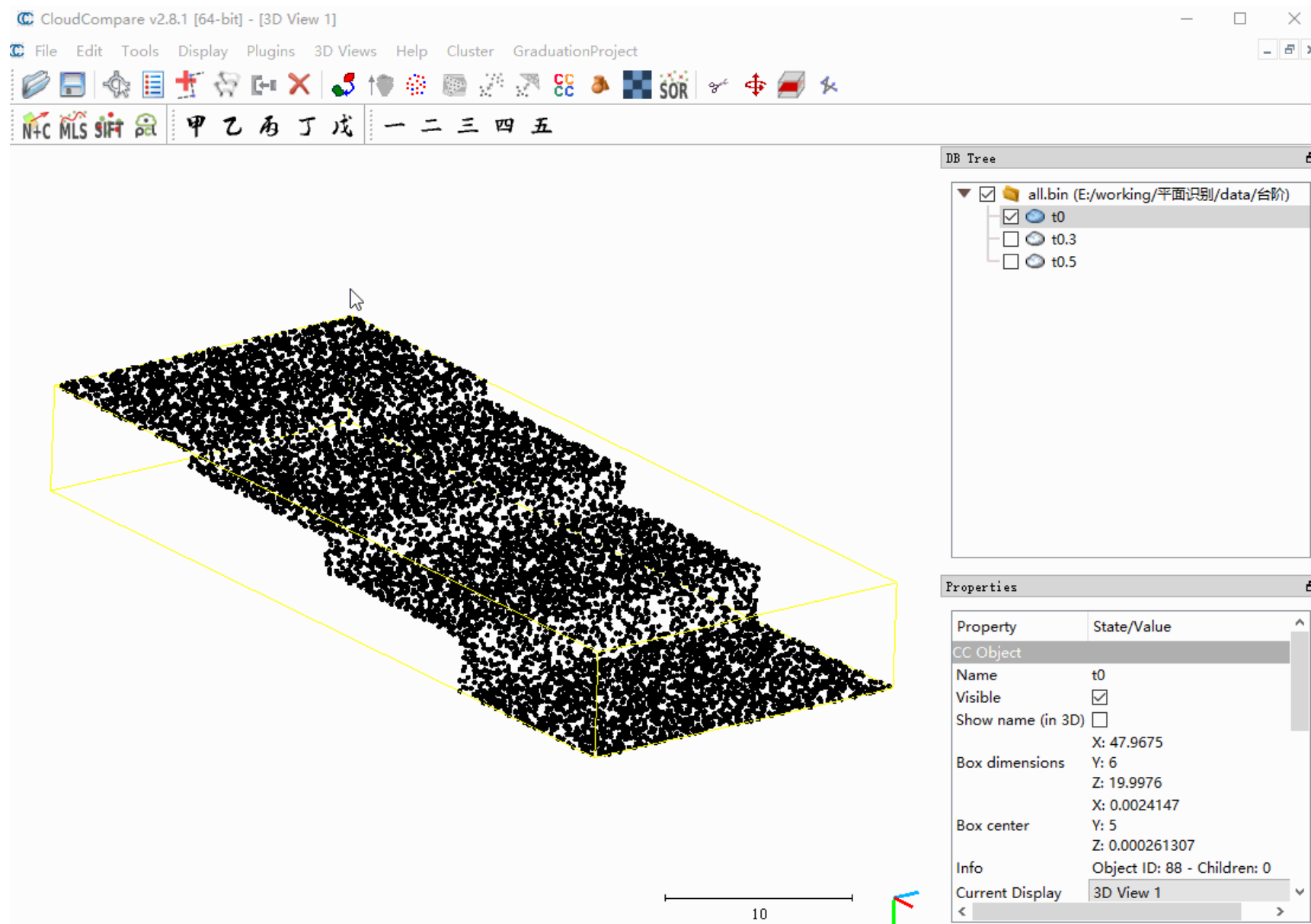
Cloudcompare插件展示

增加噪声插件



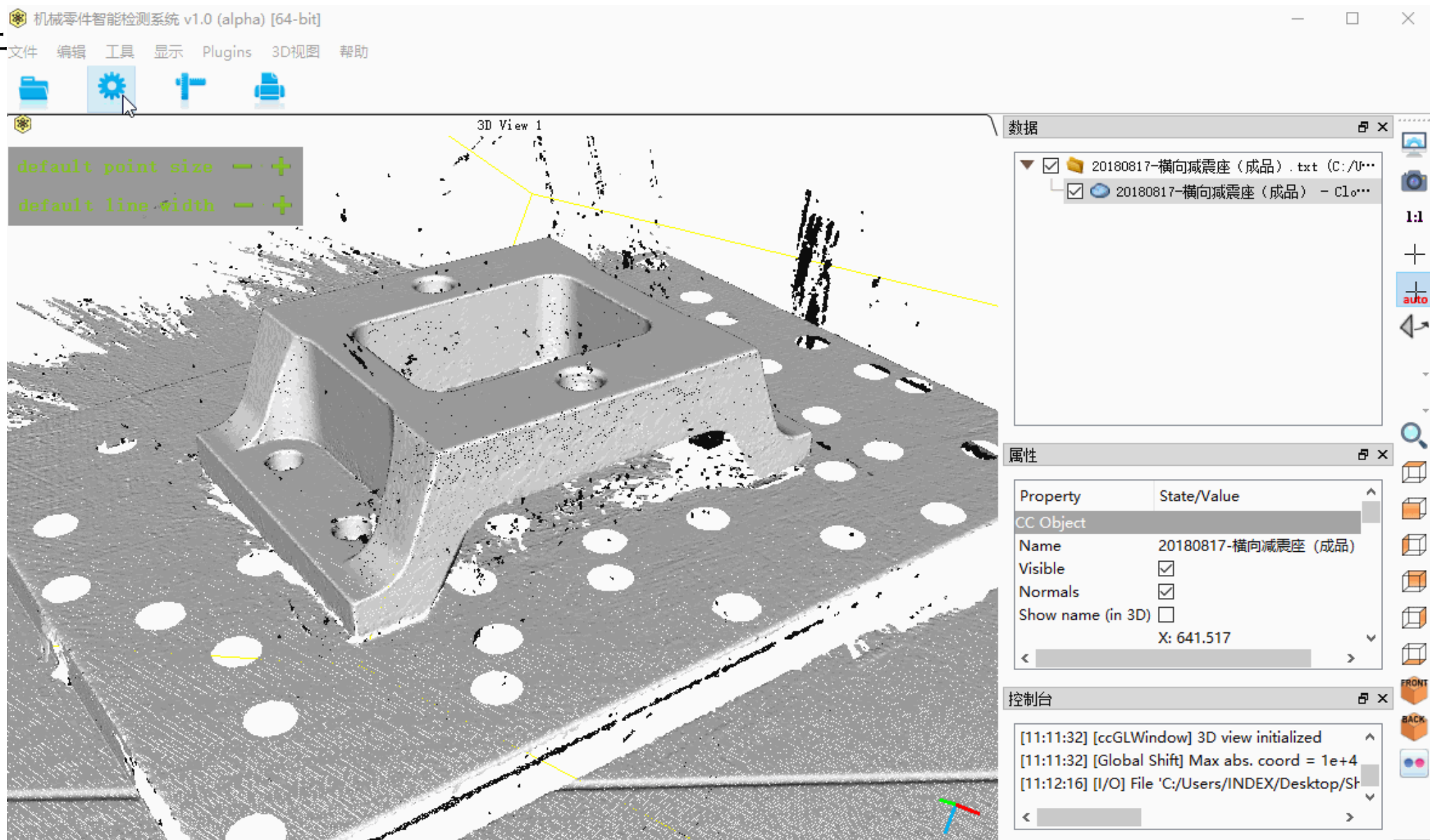
Cloudcompare插件展示

生长算法插件



Cloudcompare插件展示

目标识别插件



学习前的准备

需要的技能

C++ 基础

Qt 基础

Cmake 基础

成功编译 cloudcomapre

拥有探索精神

本教程的环境

Win10

Qt 5.12.8

VS2017

Cloudcomapre 2.11.0

cmake-3.12.3

QT的插件机制

插件是一种遵循一定规范的应用程序接口编写出来的程序。QT的插件的管理机制性能高而且简单，在meshlab和cloudcompare等软件上都有很多的应用。

一般QT写一个插件的步骤：

- 1.声明插件类，该类从QObject和该插件希望实现的接口继承而来。
- 2.用宏Q_INTERFACES()将该接口告诉Qt元对象系统。
- 3.用宏Q_EXPORT_PLUGIN2()导出插件。
Q_EXPORT_PLUGIN2 (PluginName, ClassName)
4. 用适当的.pro文件构建插件

如何编写插件

3rdParty	2020/6/16 3:50	文件夹	
core	2020/6/16 3:50	文件夹	
example	2020/6/16 3:50	文件夹	
ccCommandLineInterface.h	2020/6/16 3:50	C++ Header file	15 KB
ccDefaultPluginInterface.cpp	2020/6/16 3:50	C++ Source file	5 KB
ccDefaultPluginInterface.h	2020/6/16 3:50	C++ Header file	2 KB
ccGLPluginInterface.h	2020/6/16 3:50	C++ Header file	3 KB
ccIOPluginInterface.h	2020/6/16 3:50	C++ Header file	3 KB
ccMainAppInterface.h	2020/6/16 3:50	C++ Header file	8 KB
ccPluginInterface.h	2020/6/16 3:50	C++ Header file	5 KB
ccStdPluginInterface.h	2020/6/16 3:50	C++ Header file	4 KB
CMakeLists.txt	2020/6/16 3:50	文本文档	1 KB
CMakePluginTpl.cmake	2020/6/16 3:50	CMAKE 文件	5 KB

插件编写没有具体的教程但可以模仿原有案例案例

ExampleGLPlugin	2020/6/16 3:50	文件夹	
ExampleIOPlugin	2020/6/16 3:50	文件夹	
ExamplePlugin	2020/6/16 3:50	文件夹	
CMakeLists.txt	2020/6/16 3:50	文本文档	1 KB








基于GL可视化的插件

基于IO读取的插件

普通的函数插件

插件编写具体步骤







第一步 修改cmake文件

-  images → 按钮图标
-  CMakeLists.txt → 
-  ExamplePlugin.cpp → 主程序
-  ExamplePlugin.h → 头文件
-  ExamplePlugin.qrc → 资源文件
-  info.json → 插件描述

```
1  cmake_minimum_required( VERSION 3.0 )
2
3  # CloudCompare example for standard plugins
4
5  # REPLACE ALL 'ExamplePlugin' OCCURENCES BY YOUR PLUGIN NAME
6  # AND ADAPT THE CODE BELOW TO YOUR OWN NEEDS!
7
8  # Add an option to CMake to control whether we build this plugin or not
9  option( PLUGIN_EXAMPLE_STANDARD "Check to install example plugin" OFF )
10
11  if ( PLUGIN_EXAMPLE_STANDARD )
12      # Name the plugin
13      project( ExamplePlugin )
14
15      # load any subdirectories (see qAdditionalIO for an example)
16      # add_subdirectory( LIB1 )
17
18      include( ../../CMakePluginTpl.cmake )
19
20      # set dependencies to necessary libraries (see qPCV for an example)
21      # target_link_libraries( ${PROJECT_NAME} LIB1 )
22      # include_directories( ${LIB1_INCLUDE_DIR} )
23  endif()
24
```

插件编写具体步骤







第二步 修改插件描述文件

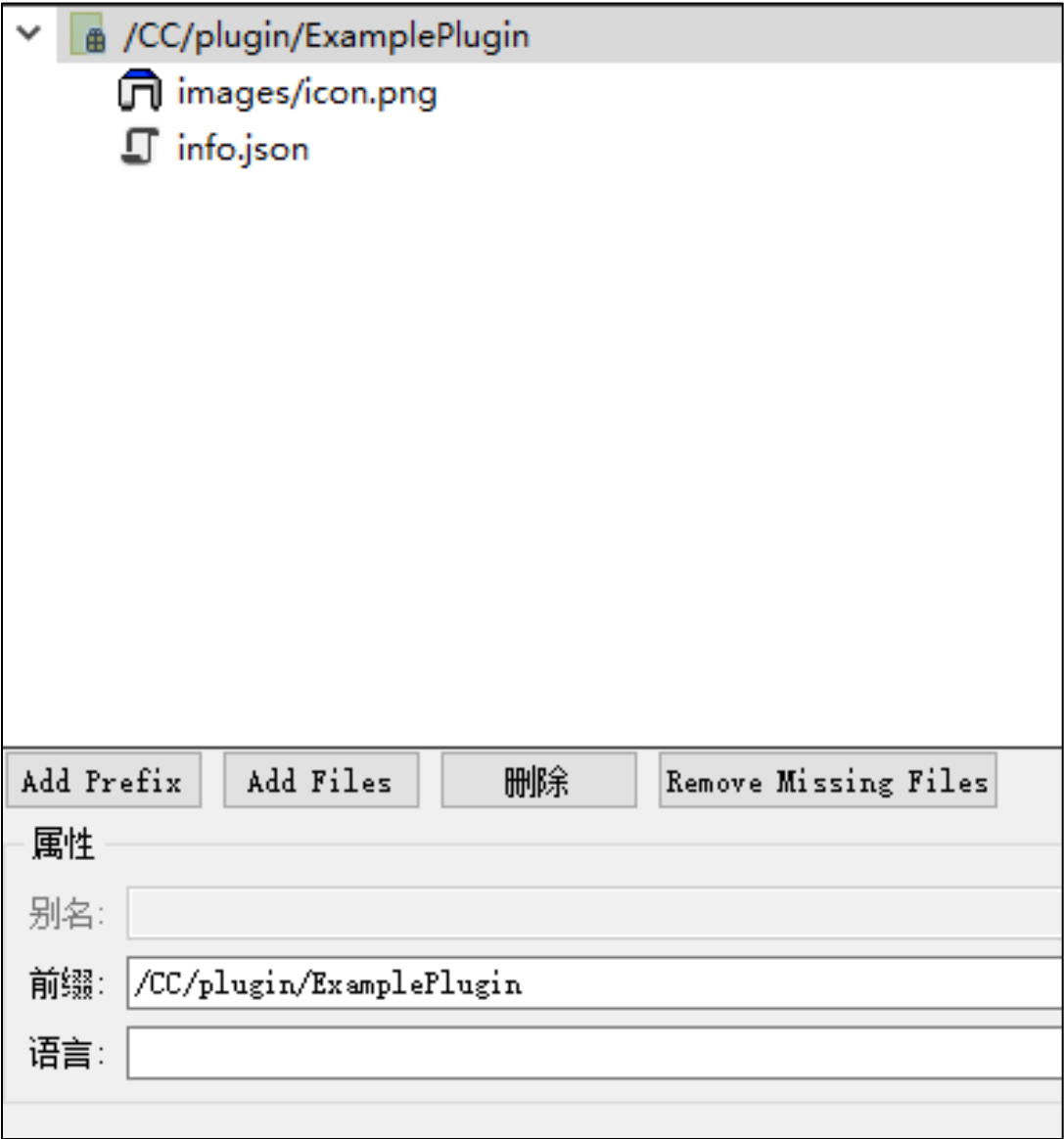
-  images → 按钮图标
-  CMakeLists.txt → CMAKE文件
-  ExamplePlugin.cpp → 主程序
-  ExamplePlugin.h → 头文件
-  ExamplePlugin.qrc → 资源文件
-  info.json →

```
1 {
2     "type" : "Standard",
3     "name" : "Example (Standard Plugin)",
4     "icon" : ":/CC/plugin/ExamplePlugin/images/icon.png",
5     "description": "This is a description of the marvelous Example plugin. It does nothing.",
6     "authors" : [
7         {
8             "name" : "Daniel Girardeau-Montaut",
9             "email" : "daniel.girardeau@gmail.com"
10        }
11    ],
12    "maintainers" : [
13        {
14            "name" : "Andy Maloney",
15            "email" : "asmaloney@gmail.com"
16        },
17        {
18            "name" : "Example NoEmail"
19        }
20    ],
21    "references" : [
22        {
23            "text" : "The unsuccessful self-treatment of a case of "writer's block"",
24            "url" : "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1311997/"
25        },
26        {
27            "text" : "Sword swallowing and its side effects",
28            "url" : "http://www.bmj.com/content/333/7582/1285"
29        },
30        {
31            "text" : "I thought of creating this wonderful plugin while I was hiking up &#x26f0; Mont Blanc"
32        }
33    ]
34 }
35
```

插件编写具体步骤

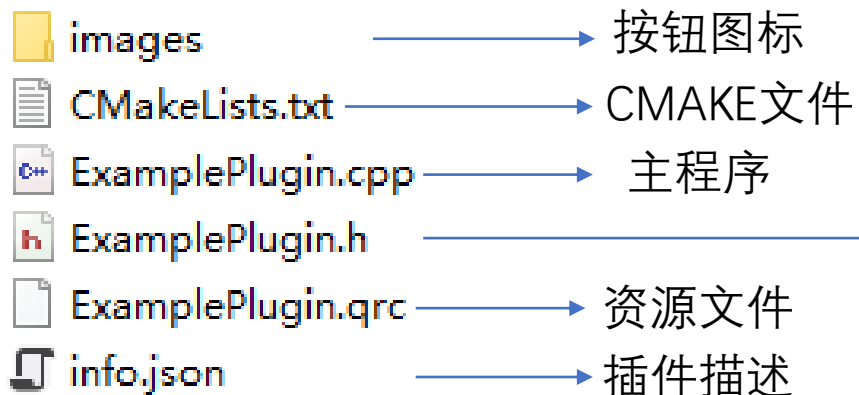
第三步 修改资源文件

-  images —————> 按钮图标
-  CMakeLists.txt —————> CMAKE文件
-  ExamplePlugin.cpp —————> 主程序
-  ExamplePlugin.h —————> 头文件
-  ExamplePlugin.qrc —————>
-  info.json —————> 插件描述



插件编写具体步骤

第四步 修改头文件与主程序



```
#include "ccStdPluginInterface.h"

/** Example qCC plugin
/** Replace 'ExamplePlugin' by your own plugin class name throughout and then
    check 'ExamplePlugin.cpp' for more directions.

    Each plugin requires an info.json file to provide information about itself -
    the name, authors, maintainers, icon, etc..

    The one method you are required to implement is 'getActions'. This should
    return all actions (QAction objects) for the plugin. CloudCompare will
    automatically add these with their icons in the plugin toolbar and to the
    plugin menu. If your plugin returns several actions, CC will create a
    dedicated toolbar and a sub-menu for your plugin. You are responsible for
    connecting these actions to methods in your plugin.

    Use the ccStdPluginInterface::m_app variable for access to most of the CC
    components (database, 3D views, console, etc.) - see the ccMainAppInterface
    class in ccMainAppInterface.h.
**/
class ExamplePlugin : public QObject, public ccStdPluginInterface
{
    Q_OBJECT
    Q_INTERFACES(ccStdPluginInterface)

    // Replace "Example" by your plugin name (IID should be unique - let's hope your plugin name is unique ;)
    // The info.json file provides information about the plugin to the loading system and
    // it is displayed in the plugin information dialog.
    Q_PLUGIN_METADATA(IID "cccorp.cloudcompare.plugin.Example" FILE "info.json")

public:
    explicit ExamplePlugin( QObject *parent = nullptr );
    ~ExamplePlugin() override = default;

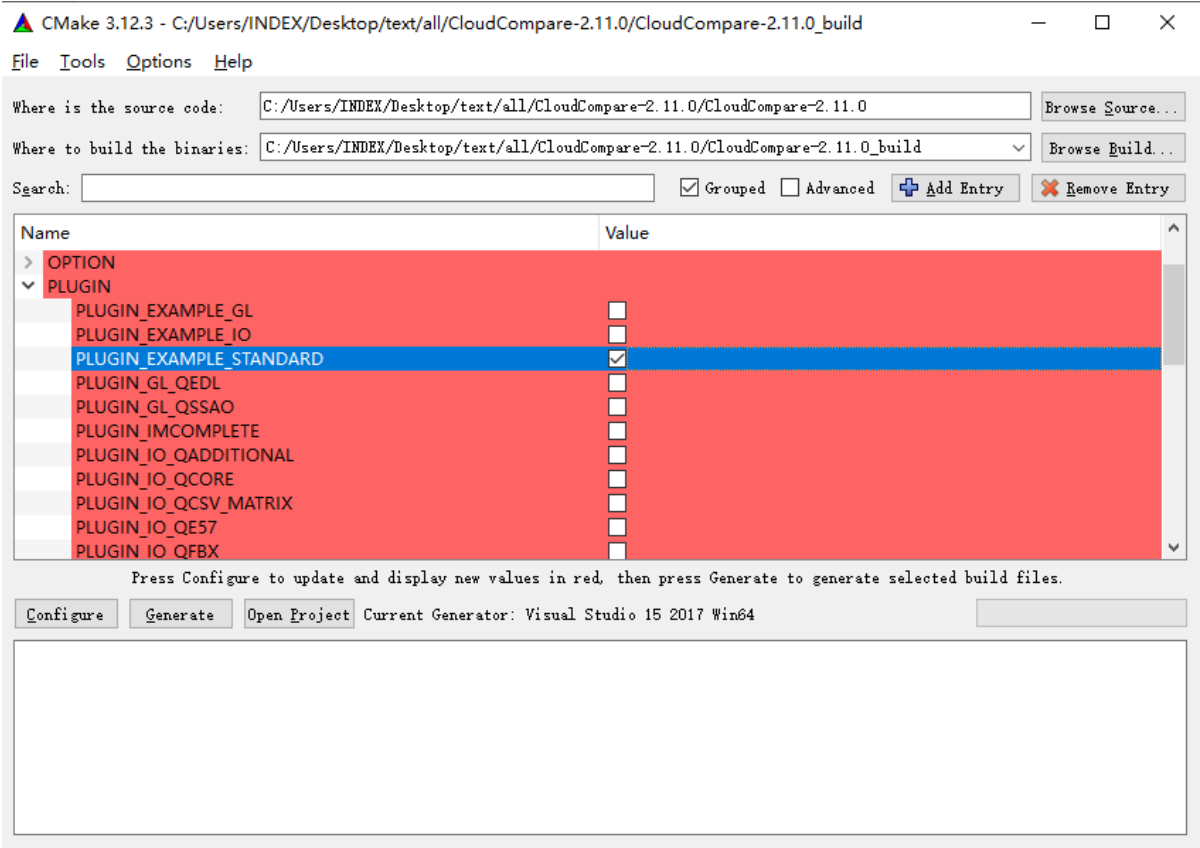
    // inherited from ccStdPluginInterface
    void onNewSelection( const ccHObject::Container &selectedEntities ) override;
    QList<QAction *> getActions() override;

private:
    /*** ADD YOUR CUSTOM ACTIONS HERE ***/
    void doAction();

    /** Default action
    /** You can add as many actions as you want in a plugin.
        Each action will correspond to an icon in the dedicated
        toolbar and an entry in the plugin menu.
    **/
    QAction* m_action;
};
```

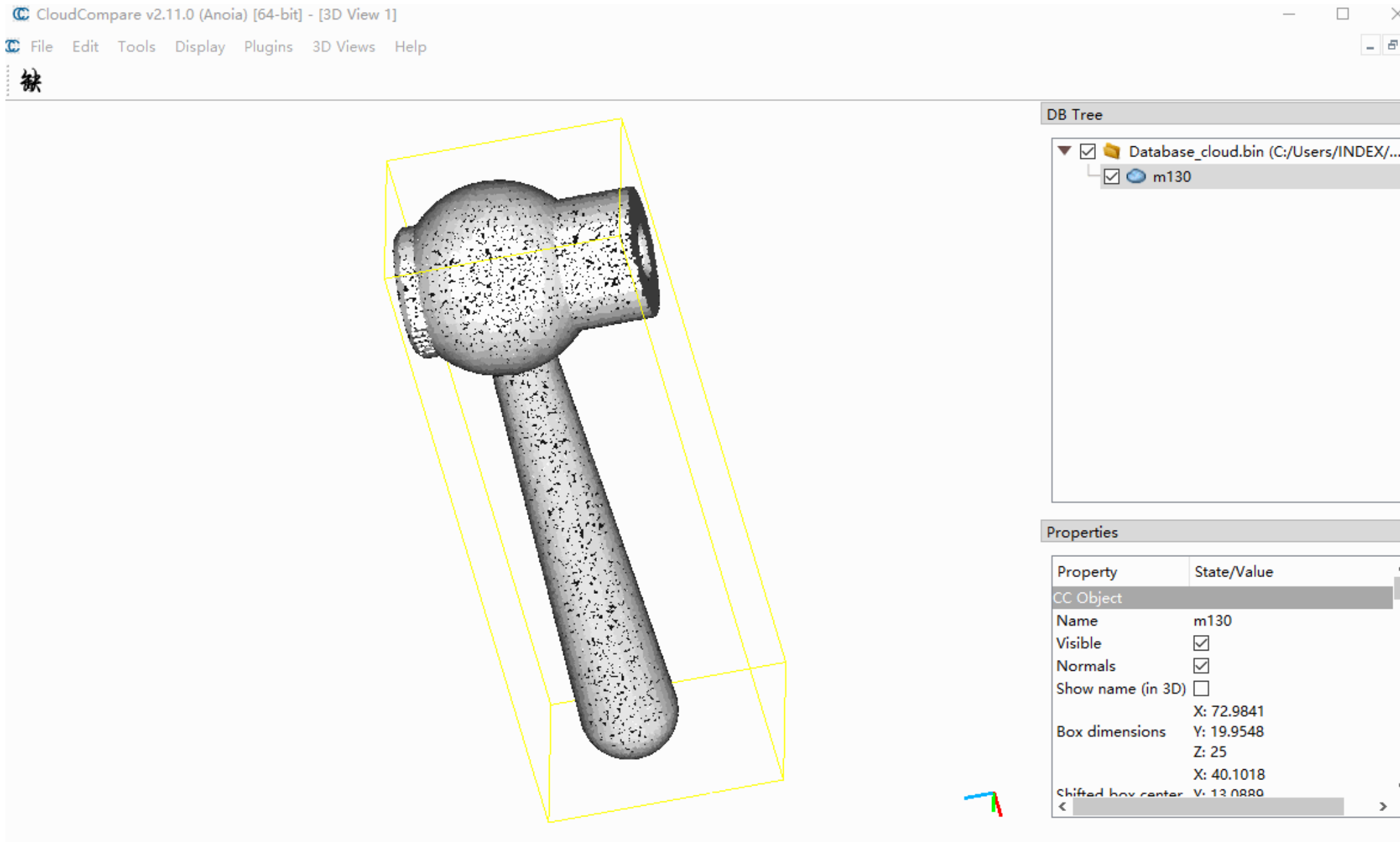

插件编写具体步骤

第五步 编译生成插件



名称	修改日期	类型	大小
 ExamplePlugin.dll	2020/6/16 19:33	应用程序扩展	34 KB
 ExamplePlugin.exp	2020/6/16 19:33	Exports Library ...	2 KB
 ExamplePlugin.lib	2020/6/16 19:33	Object File Library	3 KB

亲手实践写一个批量随机切割点云的插件





恳请各位专家同学批评指正

分享人：李子宽

日期：2020年6月19日



视觉点云



个人微信公众号

微信扫描二维码,加入“点云PCL”免费知识星球, 获取分享ppt, 与更多同行研究者们交流。备注“姓名+学校/公司+研究方向”, 否则不予通过。



星主: Particle

星球: 点云PCL



知识星球

长按扫码预览社群内容
和星主关系更近一步

扫码关注微信公众号

分享使人进步

Sharing makes progress

定期线上分享

免费知识星球

分享原创好文

