转载 数据智能笔记 于 2020-01-17 14:46:17 发布 1044 收藏 21 版权
分类专栏： PCL学习 文章标签： PCL

1.时间计算

程序运行时间计算：以秒输出"程序段"的运行时间。

```
#include <pcl/console/time.h>  // 头文件

// 程序段

pcl::console::TicToc time; time.tic();

+程序段 + cout<<time.toc()/1000<<"s"<<endl;

```

2. 实现类似pcl::PointCloud::Ptr和pcl::PointCloud的两个类相互转换

```
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>

pcl::PointCloud<pcl::PointXYZ>::Ptr cloudPointer(new pcl::PointCloud<pcl::PointXYZ>);
pcl::PointCloud<pcl::PointXYZ> cloud;
cloud = *cloudPointer;
cloudPointer = cloud.makeShared();
```

3.查找 点云 的x，y，z的极值

```
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/common/common.h>

pcl::PointCloud<pcl::PointXYZ>::Ptr cloud;
cloud = pcl::PointCloud<pcl::PointXYZ>::Ptr (new pcl::PointCloud<pcl::PointXYZ>);
pcl::io::loadPCDFile<pcl::PointXYZ> ("your_pcd_file.pcd", *cloud);    //读取点云
pcl::PointXYZ minPt, maxPt;
pcl::getMinMax3D (*cloud, minPt, maxPt);                              //计算
```

4.根据保存点的索引，从原点云中拷贝点到新点云

```
#include <pcl/io/pcd_io.h>
#include <pcl/common/impl/io.hpp>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>

// 新建点云对象cloud用于存放原点云
pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZ>);
// 打开点云 保存在cloud中
pcl::io::loadPCDFile<pcl::PointXYZ>("your_pcd_file.pcd", *cloud);
// 新建对象用于存放拷贝点云
pcl::PointCloud<pcl::PointXYZ>::Ptr cloudOut(new pcl::PointCloud<pcl::PointXYZ>)
// 索引值
std::vector<int > indexs = { 1, 2, 5 };
// 执行计算
pcl::copyPointCloud(*cloud, indexs, *cloudOut);
```

5. 从点云里删除和添加点

```
#include <pcl/io/pcd_io.h>
#include <pcl/common/impl/io.hpp>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>

pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZ>);

pcl::io::loadPCDFile<pcl::PointXYZ>("your_pcd_file.pcd", *cloud);

pcl::PointCloud<pcl::PointXYZ>::iterator index = cloud->begin();
//删除第一个
cloud->erase(index);
index = cloud->begin() + 5;
cloud->erase(cloud->begin());//删除第5个
pcl::PointXYZ point = { 1, 1, 1 };
//在索引号为5的位置1上插入一点，原来的点后移一位
cloud->insert(cloud->begin() + 5, point);
cloud->push_back(point);//从点云最后面插入一点
std::cout << cloud->points[5].x;//输出1
```

如果删除的点太多建议用上面的方法拷贝到新点云，再赋值给原点云，如果要添加很多点，建议先resize，然后用循环向点云里的添加。

6. 对点云进行全局或局部变换

```
#include <pcl/io/pcd_io.h>
#include <pcl/common/impl/io.hpp>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>
#include <pcl/common/transforms.h>


        pcl::PointCloud<pcl::PointXYZ>::Ptr cloud (new pcl::PointCloud<pcl::PointXYZ>);
        pcl::io::loadPCDFile("path/.pcd",*cloud);
        //全局变化
        //构造变化矩阵
        Eigen::Matrix4f transform_1 = Eigen::Matrix4f::Identity();
        float theta = M_PI/4;   //旋转的度数，这里是45度
        transform_1 (0,0) = cos (theta);  //这里是绕的Z轴旋转
        transform_1 (0,1) = -sin(theta);
        transform_1 (1,0) = sin (theta);
        transform_1 (1,1) = cos (theta);
        //   transform_1 (0,2) = 0.3;   //这样会产生缩放效果
        //   transform_1 (1,2) = 0.6;
        //    transform_1 (2,2) = 1;
        transform_1 (0,3) = 25; //这里沿X轴平移
        transform_1 (1,3) = 30;
        transform_1 (2,3) = 380;
        pcl::PointCloud<pcl::PointXYZ>::Ptr transform_cloud1 (new pcl::PointCloud<pcl::PointXYZ>);
```

目录

分类专栏

```
        pcl::transformPointCloud(*cloud,*transform_cloud1,transform_1);  //不言而喻

        //局部
        //第一个参数为输入，第二个参数为输入点云中部分点集索引，第三个为存储对象，第四个是变换矩阵。
        pcl::transformPointCloud(*cloud,pcl::PointIndices indices,*transform_cloud1,matrix);
```

7. 链接两个点云字段（两点云大小必须相同）

```
pcl::PointCloud<pcl::PointXYZ>::Ptr cloud (new pcl::PointCloud<pcl::PointXYZ>);
        pcl::io::loadPCDFile("your pcd file.pcd",*cloud);

        // 新建法线估计对象
        pcl::NormalEstimation<pcl::PointXYZ,pcl::Normal> ne;
        // 输入点云
        ne.setInputCloud(cloud);
        // 新建查找对象
        pcl::search::KdTree<pcl::PointXYZ>::Ptr tree (new pcl::search::KdTree<pcl::PointXYZ>());
        // 设置查找方法
        ne.setSearchMethod(tree);

        pcl::PointCloud<pcl::Normal>::Ptr cloud_normals(new pcl::PointCloud<pcl::Normal>());
        ne.setKSearch(8);
        //ne.setRadisuSearch(0.3);
        ne.compute(*cloud_normals);
        pcl::PointCloud<pcl::PointNormal>::Ptr cloud_with_nomal (new pcl::PointCloud<pcl::PointNormal>);
        pcl::concatenateFields(*cloud,*cloud_normals,*cloud_with_nomal);
```

8. 从点云中删除无效点

pcl中的无效点是指：点的某一坐标值为nan。

```
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
#include <pcl/filters/filter.h>
#include <pcl/io/pcd_io.h>

using namespace std;
typedef pcl::PointXYZRGBA point;
typedef pcl::PointCloud<point> CloudType;

    int main (int argc,char **argv)
    {
            CloudType::Ptr cloud (new CloudType);
            CloudType::Ptr output (new CloudType);


            pcl::io::loadPCDFile(argv[1],*cloud);
            cout<<"size is:"<<cloud->size()<<endl;


            vector<int> indices;
            pcl::removeNaNFromPointCloud(*cloud,*output,indices);
            cout<<"output size:"<<output->size()<<endl;


            pcl::io::savePCDFile("out.pcd",*output);

            return 0;
    }
```

9. 将xyzrgb格式转换为xyz格式的点云

```
#include <pcl/io/pcd_io.h>
#include <ctime>
#include <Eigen/Core>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>

using namespace std;
typedef pcl::PointXYZ point;
typedef pcl::PointXYZRGBA pointcolor;

int main(int argc,char **argv)
{
        pcl::PointCloud<pointcolor>::Ptr input (new pcl::PointCloud<pointcolor>);
        pcl::io::loadPCDFile(argv[1],*input);


        pcl::PointCloud<point>::Ptr output (new pcl::PointCloud<point>);
        int M = input->points.size();
        cout<<"input size is:"<<M<<endl;

        for (int i = 0;i <M;i++)
        {
                point p;
                p.x = input->points[i].x;
                p.y = input->points[i].y;
                p.z = input->points[i].z;
                output->points.push_back(p);
        }
        output->width = 1;
        output->height = M;

        cout<< "size is"<<output->size()<<endl;
        pcl::io::savePCDFile("output.pcd",*output);

}
```

10. flann kdtree 查询k近邻

```
//平均密度计算
        pcl::KdTreeFLANN<pcl::PointXYZ> kdtree;  //创建一个快速k近邻查询，查询的时候若该点在点云中，则第一个近邻点是其本身
        kdtree.setInputCloud(cloud);
        int k =2;
        float everagedistance =0;
        for (int i =0; i < cloud->size()/2;i++)
        {
                vector<int> nnh ;
                vector<float> squaredistance;
                //  pcl::PointXYZ p;
                //   p = cloud->points[i];
                kdtree.nearestKSearch(cloud->points[i],k,nnh,squaredistance);
                everagedistance += sqrt(squaredistance[1]);
                //   cout<<everagedistance<<endl;
        }

        everagedistance = everagedistance/(cloud->size()/2);
```

```
        cout<<"everage distance is : "<<everagedistance<<endl;
```

```
#include <pcl/kdtree/kdtree_flann.h>
        pcl::KdTreeFLANN<pcl::PointXYZ> kdtree; //创建KDtree
        kdtree.setInputCloud (in_cloud);
        pcl::PointXYZ searchPoint; //创建目标点，（搜索该点的近邻）
        searchPoint.x = 1;
        searchPoint.y = 2;
        searchPoint.z = 3;

        //查询近邻点的个数
        int k = 10; //近邻点的个数
        std::vector<int> pointIdxNKNSearch(k); //存储近邻点集的索引
        std::vector<float>pointNKNSquareDistance(k); //近邻点集的距离
        if (kdtree.nearestKSearch(searchPoint,k,pointIdxNKNSearch,pointNKNSquareDistance)>0)
        {
                for (size_t i = 0; i < pointIdxNKNSearch.size (); ++i)
                        std::cout << "    "   <<   in_cloud->points[ pointIdxNKNSearch[i] ].x
                                  << " " << in_cloud->points[ pointIdxNKNSearch[i] ].y
                                  << " " <<in_cloud->points[ pointIdxNKNSearch[i] ].z
                                  << " (squared distance: " <<pointNKNSquareDistance[i] << ")" << std::endl;
        }

        //半径为r的近邻点
        float radius = 40.0f;  //其实是求的40*40距离范围内的点
        std::vector<int> pointIdxRadiusSearch;  //存储的对应的平方距离
        std::vector<float> a;
        if ( kdtree.radiusSearch (searchPoint, radius, pointIdxRadiusSearch, a) > 0 )
        {
          for (size_t i = 0; i < pointIdxRadiusSearch.size (); ++i)
                  std::cout << "    "   <<   in_cloud->points[ pointIdxRadiusSearch[i] ].x
                            << " " <<in_cloud->points[ pointIdxRadiusSearch[i] ].y
                            << " " << in_cloud->points[ pointIdxRadiusSearch[i] ].z
                            << " (squared distance: " <<a[i] << ")" << std::endl;
        }
```

二. ply点云文件格式解读

后缀命名为.ply格式文件，常用的点云数据文件。ply文件不仅可以存储**点**数据，而且可以存储**网格**数据. 用emacs打开一个ply文件，观察表头，如果表头element face的值为0,ze则表示该文件为点云文件，如果element face的值为某一正整数N，则表示该文件为网格文件，且包含N个网格。

所以利用pcl读取 ply 文件，不能一味用pcl::PointCloud<PointT>::Ptr cloud (new pcl::PointCloud<PintT>)来读取。
在读取ply文件时候，首先要分清该文件是点云还是网格类文件。如果是点云文件，则按照一般的点云类去读取即可，官网例子.就是这样。

如果ply文件是网格类，则需要通过

```
pcl::PolygonMesh mesh;
pcl::io::loadPLYFile(argv[1],mesh);
pcl::io::savePLYFile("result.ply", mesh);
```

读取。（官网例子之所以能成功，是因为它对模型进行了细分处理，使得网格变成了点）

1. 计算点的索引

例如sift算法中，pcl无法直接提供索引（主要原因是sift点是通过计算出来的，在某些不同参数下，sift点可能并非源数据中的点，而是某些点的近似），若要获取索引，则可利用以下函数：

```
void getIndices (pointcloud::Ptr cloudin, pointcloud keypoints, pcl::PointIndices::Ptr indices)
{
        pcl::KdTreeFLANN<pcl::PointXYZ> kdtree;
        kdtree.setInputCloud(cloudin);
        std::vector<float>pointNKNSquareDistance; //近邻点集的距离
        std::vector<int> pointIdxNKNSearch;

        for (size_t i =0; i < keypoints.size();i++)
        {
                kdtree.nearestKSearch(keypoints.points[i],1,pointIdxNKNSearch,pointNKNSquareDistance);
                // cout<<"the distance is:"<<pointNKNSquareDistance[0]<<endl;
                // cout<<"the indieces is:"<<pointIdxNKNSearch[0]<<endl;

                indices->indices.push_back(pointIdxNKNSearch[0]);

        }

}
```

其思想就是：将原始数据插入到flann的kdtree中，寻找keypoints的最近邻，如果距离等于0,则说明是同一点，提取索引即可。

2. 计算质心

```
    Eigen::Vector4f centroid;  //质心
    pcl::compute3DCentroid(*cloud_smoothed,centroid); //估计质心的坐标
```

3. 从网格提取顶点（将网格转化为点）

```
#include <pcl/io/io.h>
#include <pcl/io/pcd_io.h>
#include <pcl/io/obj_io.h>
#include <pcl/PolygonMesh.h>
#include <pcl/point_cloud.h>
#include <pcl/io/vtk_lib_io.h>//loadPolygonFileOBJ所属头文件;
#include <pcl/io/vtk_io.h>
#include <pcl/io/ply_io.h>
#include <pcl/point_types.h>
using namespace pcl;
int main(int argc,char **argv)
{
    pcl::PolygonMesh mesh;
    //   pcl::io::loadPolygonFileOBJ(argv[1], mesh);
    pcl::io::loadPLYFile(argv[1],mesh);
    pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZ>);
    pcl::fromPCLPointCloud2(mesh.cloud, *cloud);
    pcl::io::savePCDFileASCII("result.pcd", *cloud);
    return 0;
```

```
}
```

以上代码可以从.obj或.ply面片格式转化为点云类型。

数据智能笔记 关注

PCL点云文件生成与读取 fandq1223的博客 1万+
提要 PCL中创造了一种用于描述空间点集的文件 - PCD.关于PCD的简介，可以参考这里 - http://pointclouds.org/documentation/tutorials/pcd_file_for…

pcl测试常用操作 平平酱的博客 632
标题一、代码常用1、代码运行时间二、cc常用1、两个点之间的距离 一、代码常用 1、代码运行时间 #include <ctime> clock_t start=clock(); clock_t end…

评论 1 条 写评论

zhr8686782 热评 好强

数据智能笔记
码龄5年 暂无认证

47	5万+	96万+	8万+	
原创	周排名	总排名	访问	等级

1273	127	62	26	380
积分	粉丝	获赞	评论	收藏

私信 关注

搜博主文章

热门文章

无人机数据处理—Pix4Dmapper解析 9379
无人机倾斜摄影测量影像处理关键技术 5766
Python爬虫数据存储之TXT文本 4215
VS2017+PCL+QT+VTK开发环境搭建 4155
C#之CAD二次开发(10) 用户交互之选择集 4029

最新评论

C#之CAD二次开发(16) 表格操作
眉目清秀: 这不是B站视频的源代码吗？
Python爬虫数据存储之TXT文本
m0_64314660: 弱弱的问一句，报错的位置是不是find那里？
VS2017+PCL+QT+VTK开发环境搭建
wang_nn: 博主您好，编译好的VTK能发我一份吗，自己编译了好多次，QVtkWidge…
VS2017+PCL+QT+VTK开发环境搭建
qq_42683295: 博主您好，您能把cmake编译好的VTK文件发我一份吗，谢谢您！我…
PCL报错记录（三）：VS2017+PCL1.8.…
一棵会开花的树。: 拒绝访问怎么办

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

C# ObjectArx CAD二次开发环境搭建
AutoCAD .NET 二次开发实例(7) 输出CAD文本到Excel和TxT
C#之CAD二次开发(19) 组合条件的选择集过滤

2020年 31篇 2019年 20篇

io获取 pcl_PCL读取PCD文件的数据_weixin_39750598的博客 12-22
pcl::PointCloud<:pointxyz>::Ptr cloud(new pcl::PointCloud<:pointxyz>);// //*打开点云文件 if (pcl::io::loadPCDFile<:pointxyz>("rabbit.pcd", *cloud) == -1) { …

PCL教程-点云配准之正态分布变换算法(NDT)_SOC罗三炮的博客_p… 1-3
pcl::PointCloud<pcl::PointXYZ>::Ptrinput_cloud(newpcl::PointCloud<pcl::PointXYZ>); if(pcl::io::loadPCDFile<pcl::PointXYZ> ("room_scan2.pcd", *input_…

PCL中一些常用函数总结 最新发布 weixin_43834466的博客 32
PCL常用方法总结

点云库PCL的使用 A_L_A_N 1万+
相关头文件： #include <pcl/io/io.h> #include <pcl/io/pcd_io.h> #include <pcl/point_types.h> #include <pcl/visualization/cloud_viewer.h> 声明和定义点云…

保存点云数据_PCL入门系列三——PCL进行数据读写_TRLU番的葡萄的博客… 12-31
pcl::io::savePCDFileASCII("test_pcd.pcd", cloud); std::cerr <<"Saved "<< cloud.points.size() <<" data points to test_pcd.pcd."<< std::endl; for(size_ti =0…

PCL学习笔记->基本用法3(在你的项目中使用PCL) JunJun 3148
在自己的项目中使用PCL 本教程说明了如何在自己的项目中使用PCL。 先决条件 我们假设您已经在计算机上下载，编译并安装了PCL。 项目设定 假设该…

pcl的使用基础，python-pcl安装 daoer_sofu的专栏 5832
PCL

点云学习----PCL读写点云 liuxiaojiangtobeno1的博客 2073
1.读取pcd文件： int main() { //定义点云对象 pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZ>); //加载读取点云数据到cloud…

PCL：读取指定路径下的pcd点云 | 将点云保存至指定路径 NOIF 3296
本文进一步介绍了点云读写的内容，读取指定路径下的点云，将点云保存至指定路径下。

pcl处理pcd文件的问题 harden1013的博客 768
问题1：pcd可以用pcl_viewer等点云可视化工具查看，但无法用文本形式打开查看； 问题2：采集的pcd中存在大量nan点数据，在配准等操作时报错如下…

【PCL】【PCL实践】【PCL的使用学习记录】 qq_45954434的博客 902
全文代码参考【slam十四讲第二版】【课本例题代码向】【第五讲~相机与图像】【opencv3.4.1安装】【OpenCV、图像去畸变、双目和RGB-D、遍历图…

PCL-点云读入 qq_40149035的博客 772
1.pcl点云读入代码 #include <iostream> #include <pcl/io/pcd_io.h> #include <pcl/point_types.h> // 里面是pcl库点云的数据类型 int main(int argc,char** a…

PCL IO 操作 月照银海似蛟龙的博客 442
PCL IO 操作PCD文件介绍 PCD文件介绍 PCL以PCD文件形式保存点云。 （主要是为了保存n维点类型） 其文件格式如下： 每一个PCD文件都包含一个文…

PCL估计点云的表面法向量 热门推荐 wolfcsharp的博客 1万+
PCL估计点云的表面法向量估计点云表面法向量的方法理论基础法线确定方法法线方向确定选择合适的邻域尺度PCL估计表面法线代码实现用OpenMP加…

python读取pcd文件_（一）读取PCD文件 weixin_39795419的博客 2795
下面是一个简单的读取PCD文件并显示的代码：#include #include #include #include #include void main(){/* Create Point Cloud */pcl::PointCloud::Ptr clo…

PCL读取PCD文件的数据 aituochang1886的博客 1688
1.pcd文件——rabbit.pcd 链接：https://pan.baidu.com/s/1v6mjPjwd7flqUSjIIGTIGQ 提取码：zspx 新建项目pcl rabbit.pcd 和pcl.cpp在同一目录下 2.读取…

PCL教程指南-读取PCD文件,写入PCD文件,合并点云 qq_41795143的博客 2055
PCL教程指南-读取PCD文件,写入PCD文件,合并点云 一.读取PCD文件 读取PCD应用pcl::io::loadPCDFile<pcl::PointT>(pcd文件,点云对象)函数，针对官方…

IO weixin_30955617的博客 69
Transform a point cloud 1 #include <iostream> 2 #include <pcl/io/pcd_io.h> 3 #include <pcl/io/ply_io.h> 4 #include <pcl/point_cloud.h> 5 #include <pcl/…

点云库PCL学习笔记 -- 输入输出IO -- 6.PCL中记录时间长度TicToc 类和系统Time 类 杰尼君的博客 423
点云库PCL学习笔记 -- 输入输出IO -- 6.PCL中记录时间长度TicToc 类和系统Time 类 PCL库中用于记录时间长度的方法 第一种：TicToc 类的方法 添加头…

python读取pcd文件_PCL读取PCD文件的数据 weixin_39873741的博客 1062
1.pcd文件——rabbit.pcd新建项目pclrabbit.pcd 和pcl.cpp在同一目录下2.读取文件(1)显示数据#include#include#includeint main(int argc, char**argv) {//…

PCL 记录时间长度 — TicToc 类 AoboSir.com 4325
原博文链接在我的官方网站，网址是：http://www.aobosir.com/blog/2017/02/08/pcl-console-time-TicToc/对于PCL在Windows和Linux上的环境的搭建请参…

“相关推荐”对你有帮助么？

非常没帮助 没帮助 一般 有帮助 非常有帮助