

磊磊哈哈
码龄6年
暂无认证

94
原创

15万+
周排名

118万+
总排名

10万+
访问

等级

1352
积分

911
粉丝

105
获赞

34
评论

701
收藏

私信

关注

搜博文文章

热门文章

PCL点云学习六（边缘提取）
@ 4164

gtsam 学习十（ISAM2 理论）
@ 3685

PNP问题学习
@ 2613

gtsam 学习九（imu 因子）
@ 2598

获取激光雷达数据
@ 2556

最新评论

PCL点云库 学习六(提取、条件剔除、半...
十一·先生·大佬，我用条件剔除的方法对深度相机返回的点云进行高度上的剔除，但...
捷联惯导系统学习6.4(平方根滤波)
weixin_57937717: 想请问一下用待定系数法求出rk有两个值，代到量测更新中求Δ...
gtsam库 学习一（因子图理论基础）
一大碗面加俩蛋·贝叶斯定理写错了bro
捷联惯导系统学习2.1（反对称矩阵）
qq_41364482: 唉，网上这么多博客，没一个把行列式和矩阵分清的，，，
PCL 点云库学习四（双边滤波器）
小灰墩: 有序点云只是说点云存储结构像图片像素一样有width和height，并不是点云...

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

FreeRTOS 笔记二（中断配置和临界段）

freertos 笔记一（HAL 库移植）

pcl 学习论文阅读（Towards 3D Object Maps for Autonomous Household Robots）

2021年 39篇

2020年 105篇

PCL 库学习二（查找、点云压缩、octree）

转载磊磊哈哈于 2021-03-19 21:08:32 发布272 收藏 5

激光slam 学习 专栏收录该内容

27 订阅28 篇文章订阅专栏

来自点云PCL库到精通到入门

1. 查找

• K临近

```
1 kdtree.nearestKSearch (searchPoint, K, pointIndex, pointSquareDistance)
2 //searchPoint 查询点
3 //K 最临近点的个数
4 //pointIndex 查询到的临近值
5 //pointSquareDistance 存储的临近距离
6 //return 搜索到的点的个数
```

• R半径搜索

```
1 kdtree.radiusSearch(searchPoint,R,RpointIndex,RpointSquareDistance,20){
2 //searchPoint 查询点
3 //R 最大半径
4 //RpointIndex 查询到的临近值
5 //RpointSquareDistance 存储的临近距离
6 //20 最多搜索到的点
7 //return 搜索到的点的个数
```

实例：

```
1 #include <pcl/point_types.h>
2 #include <pcl/point_cloud.h>
3 #include <pcl/kdtree/kdtree_flann.h>
4 #include <vector>
5 #include <ctime>
6 #include <iostream>
7
8 using namespace std;
9 using namespace pcl;
10 int main(int argc, const char** argv) {
11     srand(time(NULL));
12     //创建数据
13     pcl::PointCloud<PointXYZ>::Ptr cloud(new PointCloud<PointXYZ>);
14     cloud->width=100;
15     cloud->height=20;
16     cloud->points.resize(cloud->width*cloud->height);
17     for(int i=0;i<cloud->points.size();i++){
18         cloud->points[i].x=1024.0f*rand()/(RAND_MAX+1.0f);
19         cloud->points[i].y=1024.0f*rand()/(RAND_MAX+1.0f);
20         cloud->points[i].z=1024.0f*rand()/(RAND_MAX+1.0f);
21     }
22
23     //创建kd树对象
24     KdTree<FLANN::PointXYZ> kdtree;
25     //创建搜索空间
26     kdtree.setInputCloud(cloud);
27
28     //查询点，随机随机
29     PointXYZ searchPoint;
30     searchPoint.x=1024.0f*rand()/(RAND_MAX+1.0f);
31     searchPoint.y=1024.0f*rand()/(RAND_MAX+1.0f);
32     searchPoint.z=1024.0f*rand()/(RAND_MAX+1.0f);
33
34     //k临近搜索
35     int K=10;
36     vector<int> pointIndex(K); //存储查询点的k临近索引
37     vector<float> pointSquareDistance(K); //存储临近点的平方距离
38
39     //输出查询点的信息
40     cout<<"X : "<<searchPoint.x<<"Y: "<<searchPoint.y<<"Z: "<<searchPoint.z<<endl;
41
42
43     //K临近搜索
44     cout<<" K临近搜索" <<std::endl;
45     cout<<"K"<<K<<endl;
46     if (kdtree.nearestKSearch (searchPoint, K, pointIndex, pointSquareDistance) >0 )
47     {
48         //searchPoint 查询点
49         //K 最临近点的个数
50         //pointIndex 查询到的临近值
51         //pointSquareDistance 存储的临近距离
52         //return 搜索到的点的个数
53         for (size_t i=0; i<pointIndex.size (); ++i){
54             std::cout<<"第1个"<<i;
55             std::cout<<"点的索引值: "<<pointIndex[i]<<endl;
56             cout<<"X:"<< cloud->points[ pointIndex[i] ].x <<endl;
57             cout<<"Y:"<< cloud->points[pointIndex[i] ].y <<endl;
58             cout<<"Z:"<<cloud->points[pointIndex[i] ].z <<endl;
59             cout<<" 平方距离: "<<pointSquareDistance[i] <<std::endl;;
60         }
61     }
62     //R半径搜索
63     float R=256.0f* rand () / (RAND_MAX +1.0f);
64     vector<int> RpointIndex; //存储查询点的R半径索引
65     vector<float> RpointSquareDistance; //存储临近点的平方距离
66     cout<<" R半径搜索" <<std::endl;
67     cout<<"R"<<R<<endl;
68     if(kdtree.radiusSearch(searchPoint,R,RpointIndex,RpointSquareDistance,20)){
69         //searchPoint 查询点
70         //R 最大半径
71         //RpointIndex 查询到的临近值
72         //RpointSquareDistance 存储的临近距离
73         //20 最多搜索到的点
74         //return 搜索到的点的个数
75         for(int i=0;i<RpointIndex.size();i++){
76             std::cout<<"第1个"<<i;
77             std::cout<<"索引值:"<<RpointIndex[i]<<endl;
78             std::cout<<"X:"<<cloud->points[RpointIndex[i]].x<<endl;
79             std::cout<<"Y:"<<cloud->points[RpointIndex[i]].y<<endl;
80             std::cout<<"Z:"<<cloud->points[RpointIndex[i]].z<<endl;
81             std::cout<<"距离:"<<RpointSquareDistance[RpointIndex[i]]<<endl;
82         }
83     }
```

分类专栏

Freertos

arm_linux

C++/C

2篇

学习

17篇

makefile 学习

2篇

ROS

BP

opencv

5篇

任务

1篇

```
84     return 0;
85 }
86
```

2. 压缩

```
1 #include <pcl/point_types.h>
2 #include <pcl/point_cloud.h>
3 #include <pcl/io/ply_io.h>
4 #include <pcl/compression/octree_pointcloud_compression.h>
5 #include <iostream>
6 #include <pcl/octree/octree_pointcloud.h>
7 #include <pcl/visualization/cloud_viewer.h>
8 #include <fstream>
9 using namespace std;
10 using namespace pcl;
11 int main(int argc, const char** argv) {
12
13     pcl::PointCloud<PointXYZRGB>::Ptr cloud(new pcl::PointCloud<PointXYZRGB>());
14     //读取ply数据
15     io::loadPLYFile("/home/n1/notes/pcl/compress/init.ply",*cloud);
16     io::compression_Profiles_e compressionProfile = pcl::io::MANUAL_CONFIGURATION;
17     //压缩选项
18     // 通过设置:
19     //   pointResolution:      点采样率, 即精度
20     //   octreeResolution:    八叉树分辨率, 八叉树最小块
21     //   doVoxelGridDownDownSampling: 是否开启体素滤波的下采样率
22     //   iFrameRate:          编码率, 每隔30次进行一次I编码, 中间帧使用P编码
23     //   colorBitResolution:   颜色所占bit;
24     //   doColorEncoding:      是否开启颜色编码
25     //   LOW_RES_ONLINE_COMPRESSION_WITHOUT_COLOR,
26     //   LOW_RES_ONLINE_COMPRESSION_WITH_COLOR,
27
28     //   MED_RES_ONLINE_COMPRESSION_WITHOUT_COLOR,
29     //   MED_RES_ONLINE_COMPRESSION_WITH_COLOR,
30
31     //   HIGH_RES_ONLINE_COMPRESSION_WITHOUT_COLOR,
32     //   HIGH_RES_ONLINE_COMPRESSION_WITH_COLOR,
33
34     //   LOW_RES_OFFLINE_COMPRESSION_WITHOUT_COLOR,
35     //   LOW_RES_OFFLINE_COMPRESSION_WITH_COLOR,
36
37     //   MED_RES_OFFLINE_COMPRESSION_WITHOUT_COLOR,
38     //   MED_RES_OFFLINE_COMPRESSION_WITH_COLOR,
39
40     //   HIGH_RES_OFFLINE_COMPRESSION_WITHOUT_COLOR,
41     //   HIGH_RES_OFFLINE_COMPRESSION_WITH_COLOR,
42
43     //   COMPRESSION_PROFILE_COUNT,
44     //   MANUAL_CONFIGURATION
45     bool showStatistics=true;
46     io::OctreePointCloudCompression<PointXYZRGB>* PointCloudEncoder;//通过八叉树进行点云压缩
47
48     //压缩
49     PointCloudEncoder<new pcl::io::OctreePointCloudCompression<pcl::PointXYZRGB>(compressionProfile, showStatistics);
50     //io::OctreePointCloudCompression<> 参数
51     //   compressionProfile_arg 参数设置
52     //   自定义compressionProfile_arg参数
53     //   octreeResolution_arg 分辨率
54     //   doVoxelGridDownDownSampling_arg 是否开启体素滤波的下采样率
55     //   iFrameRate_arg:        编码率, 每隔30次进行一次I编码, 中间帧使用P编码
56     //   doColorEncoding_arg     启动颜色编码
57     //   colorBitResolution_arg  RGB每一位的所占位数
58     //   showStatistics_arg      是否将压缩相关的统计信息打印到标准输出上
59     std::stringstream compressedData;//输入输出流
60     pcl::PointCloud<pcl::PointXYZRGB>::Ptr compresscloud(new pcl::PointCloud<pcl::PointXYZRGB>());
61     //压缩点云
62     PointCloudEncoder->encodePointCloud(cloud,compressedData);
63     //cloud
64     //compressedData 二进制流文件
65
66     //解压缩点云
67     PointCloudEncoder->decodePointCloud(compressedData, compresscloud);
68     cout<<"compressedData:size"<<sizeof(compressedData)<<endl;
69     fstream out("/home/n1/notes/pcl/compress/compress.txt");
70     string data;
71     compressedData->data;
72     out.write(data.c_str(),sizeof(data));
73     pcl::PLYWriter write;
74     write.write("/home/n1/notes/pcl/compress/compress.ply",*compresscloud,false,false);
75     delete(PointCloudEncoder);
76     return 0;
77 }
```

3. octree

```
1 #include <pcl/pcl_macros.h>
2 #include <pcl/point_cloud.h>
3 #include <pcl/octree/octree.h>
4 #include <iostream>
5 using namespace std;;
6 using namespace pcl;
7 int main(int argc, const char** argv) {
8     srand( (unsigned int)time(NULL));
9     PointCloud<PointXYZ>::Ptr cloud(new PointCloud<PointXYZ>());
10     cloud->width=200;
11     cloud->height=20;
12     cloud->resize(cloud->width*cloud->height);
13     for(int i=0;i<cloud->size();i++){
14         cloud->points[i].x=1024.0f*rand()/(RAND_MAX+1.0f);
15         cloud->points[i].y=1024.0f*rand()/(RAND_MAX+1.0f);
16         cloud->points[i].z=1024.0f*rand()/(RAND_MAX+1.0f);
17     }
18     float resolution=128.0f;
19     //resolution octree最低的分辨率
20     octree::IOctreePointCloudSearch<PointXYZ> octree(resolution);
21     //设置点云输出
22     octree.setInputCloud(cloud);
23     //构建八叉树
24     octree.addPointsFromInputCloud();
25     PointXYZ searchPoint;//查找点
26     searchPoint.x=1024.0f*rand()/(RAND_MAX+1.0f);
27     searchPoint.y=1024.0f*rand()/(RAND_MAX+1.0f);
28     searchPoint.z=1024.0f*rand()/(RAND_MAX+1.0f);
29
30     vector<int> pointIndexVec;//查询序号值
31     //体素临近搜索
32     if(octree.voxelSearch(searchPoint,pointIndexVec)){
33         //查询点          searchPoint
34         //查询点的序号    pointIndexVec
35     }
```

```
35     cout<<"x:"<<searchPoint.x<<"y:"<<searchPoint.y<<"z:"<<searchPoint.z<<endl;
36     cout<<"pointIndexVec.size():"<<pointIndexVec.size()<<endl;
37     for(size_t i=0;i<pointIndexVec.size();i++){
38         cout<<"x:"<<cloud->points[pointIndexVec[i]].x<<"y:"<<cloud->points[pointIndexVec[i]].y<<"z:"<<cloud->
39     }
40 }
41 }
42 //K搜索
43 vector<int> pointKsearchIdl;
44 vector<float> pointKsquare;
45 int K=10;
46 if(octree.nearestKSearch(searchPoint,K,pointKsearchIdl,pointKsquare)>0){
47     cout<<"x:"<<searchPoint.x<<"y:"<<searchPoint.y<<"z:"<<searchPoint.z<<endl;
48     cout<<"pointKsearchIdl.size():"<<pointKsearchIdl.size()<<endl;
49     for(size_t i=0;i<pointKsearchIdl.size();i++){
50         cout<<"x:"<<cloud->points[pointKsearchIdl[i]].x
51         <<"y:"<<cloud->points[pointKsearchIdl[i]].y
52         <<"z:"<<cloud->points[pointKsearchIdl[i]].z
53         <<"距离:"<<pointKsquare[i]<<endl;
54     }
55 }
56 //R搜索
57 vector<int> pointRsearchIdl;
58 vector<float> pointRsquare;
59 int R=256.0f* rand () / (RAND_MAX +1.0f);
60 if(octree.radiusSearch(searchPoint,R,pointRsearchIdl,pointRsquare,20)>0){
61     cout<<"x:"<<searchPoint.x<<"y:"<<searchPoint.y<<"z:"<<searchPoint.z<<endl;
62     cout<<"pointRsearchIdl.size():"<<pointRsearchIdl.size()<<endl;
63     for(size_t i=0;i<pointRsearchIdl.size();i++){
64         cout<<"x:"<<cloud->points[pointRsearchIdl[i]].x
65         <<"y:"<<cloud->points[pointRsearchIdl[i]].y
66         <<"z:"<<cloud->points[pointRsearchIdl[i]].z
67         <<"距离:"<<pointRsquare[i]<<endl;
68     }
69 }
70 octree.approxNearestSearch
71 return 0;
72 }
73 }
```

©2022 CSDN 皮肤主题：数字20 设计师：CSDN官方博客 返回首页

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务
中国互联网举报中心 Chrome商店下载 账号管理 规范 版权与免责声明 版权申诉 出版物许可证 营业执照 ©1999-2023北京创新乐知网络技术有限公司



磊磊哈哈

关注

0



0



0



专栏目录

Beta



举报