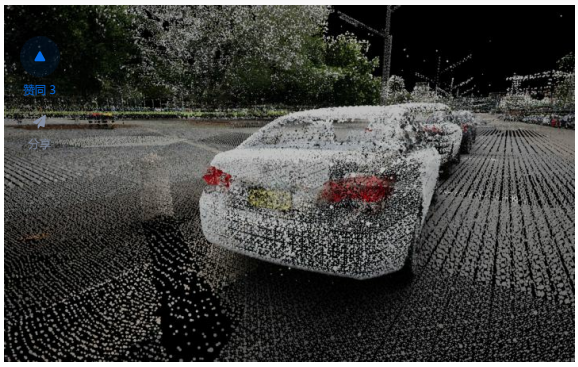


【PCL随记】三维点云体素下采样滤波器

岸边散步的鱼儿
不见高山，不显平地

前言

获取的点云通常有噪声点与离群点，其中噪声点是由于设备精度、操作者经验、环境、配准操作过程等因素的影响，点云不可避免地出现了噪声点；而由于受到视线遮挡、障碍物等因素的影响，点云也会出现离群点，这些点远离主体点云，距离被测点云较远；并且点云还会受到孔洞、数据压缩等影响。

滤波处理常常是点云处理中预处理的第一步，而滤波处理就是要针对上面的四个问题，进行预处理，将噪声点、离群点、孔洞和数据压缩进行定制处理，具体如何处理依据后续所要进行的应用，比如：点云配准、特征提取、曲面重建等

点云滤波通常分为三类：

- 常用滤波
- 采样滤波
- 裁减滤波

PCL中提供的典型滤波器：

- 直通滤波器
- 体素滤波器
- 统计滤波器
- 半径滤波器
- 条件滤波器
- 模型滤波器
- 投影滤波器
- 索引滤波器
- 高斯滤波
- 双边滤波
- 中值滤波

需要点云滤波处理的情况：

1. 点云数据的密度不规则，需要进行平滑
2. 有离群点出现，需要去除
3. 大量数据，需要进行下采样
4. 噪声数据，需要去除

遇到需要点云滤波处理时，处理方法如下：

1. 按后续需求对点进行去除和过滤
2. 利用滤波算法修改点的部分属性
3. 对大量数据进行下采样

VoxelGrid滤波器

该滤波器主要使用体素化网络方法实现下采样，可以减少点的数量。

不仅减少了点的数量，还保持了点云的形状特征，常用于提高配准、曲面重建、形状识别等应用中

PCL的VoxelGrid类为输入的点云数据创建一个三维体素栅格，在每个体素内，用体素中所有点的重心来近似显示体素中的其他点，此时该体素内所有点都用一个重心点最终表示，虽然这比使用体素中心逼近的方法慢，但是对曲面的表示更准确。

需要使用的头文件：

```
#include <pcl/filters/voxel_grid.h>
```

实例化体素滤波器对象：

```
VoxelGrid<PointXYZ> vg;
```

设置体素滤波器的一些属性：

```
vg.setInputCloud (cloud);  // 输入要滤波的点云
vg.setLeafSize (0.01f, 0.01f, 0.01f);  // 滤波时，创建的体素大小为1cm的立方体
vg.filter (*filtered_cloud);
```

完整可视化代码：

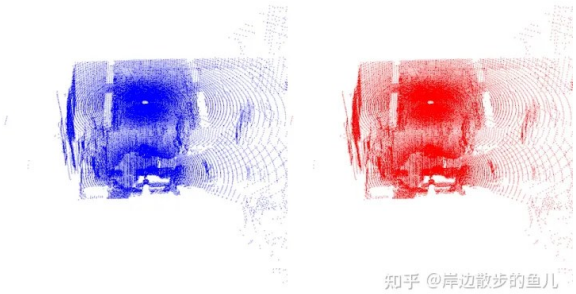
```
#include <iostream>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>
#include <pcl/filters/voxel_grid.h>
#include <pcl/visualization/pcl_visualizer.h>

using namespace std;
using namespace pcl;

void
visualize (PointCloud<PointXYZ>::Ptr source, PointCloud<PointXYZ>::Ptr target)
```

```
{
  visualization::PCLVisualizer viewer ("Point Cloud Viewer");

  // 创建两个显示窗口
  int v1, v2;
  viewer.createViewPort (0, 0.0, 0.5, 1.0, v1);
  viewer.createViewPort (0.5, 0.0, 1.0, 1.0, v2);

  // 设置背景颜色
  viewer.setBackgroundColor (255, 255, 255, v1);
  viewer.setBackgroundColor (255, 255, 255, v2);

  // 给点云添加颜色
  visualization::PointCloudColorHandlerCustom<PointXYZ> source_color (source, 0, 0, 2

  visualization::PointCloudColorHandlerCustom<PointXYZ> target_color (target, 255, 0,


  // 添加点云到显示窗口
  viewer.addPointCloud (source, source_color, "source cloud", v1);
  viewer.addPointCloud (target, target_color, "target cloud", v2);


  while (!viewer.wasStopped ())
  {
    viewer.spinOnce (100);
    boost::this_thread::sleep (boost::posix_time::microseconds(100000));
  }

}

int
main(int argc, char** argv)
{
  PointCloud<PointXYZ>::Ptr cloud (new pcl::PointCloud<PointXYZ>);
  PointCloud<PointXYZ>::Ptr filtered_cloud (new pcl::PointCloud<PointXYZ>);

  io::loadPCDFile ("room_scan1.pcd", *cloud);
  // 输出滤波前点的个数
  cout << "滤波前有:" << cloud->points.size () << "个点" << endl;

  VoxelGrid<PointXYZ> vg;
  vg.setInputCloud (cloud);
  vg.setLeafSize (0.1, 0.1, 0.1);
  vg.filter (*filtered_cloud);

  cout << "滤波后有:" << filtered_cloud->points.size () << "个点" << endl;

  visualize (cloud, filtered_cloud);

  return 0;
}
```

实验效果：

体素大小设置为：0.01

```
vg.setLeafSize (0.01, 0.01, 0.01);
```

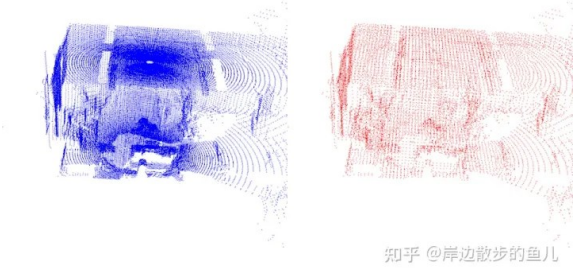


点数的比较：

```
滤波前有:112586个点
滤波后有:45161个点
```

体素大小设置为：0.1

```
vg.setLeafSize (0.1, 0.1, 0.1);
```

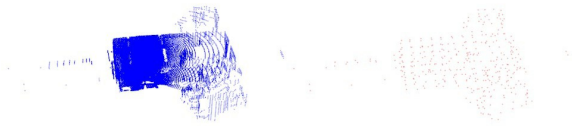


点数的比较：

```
滤波前有:112586个点
滤波后有:13490个点
```

体素大小设置为：1

```
vg.setLeafSize (1, 1, 1);
```



点数的比较：

```
滤波前有:112586个点
滤波后有:384个点
```

从上面的三个实验可以看到，体素大小设置得越大，体素化的栅格越大，下采样后的点越少，体素大小设置得越小，体素化的栅格越小，保留的点越多

点的密度大小和整齐程度不同，虽然点的数量变少了，但是其所含有的形状特征与空间结构信息与原始点云无异。

以上，就是PCL中的体素下采样滤波器

编辑于 2022-06-08 23:11

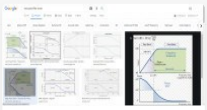
滤波器 点云库PCL 点云配准

写下你的评论...

发布

还没有评论，发表第一个评论吧

推荐阅读



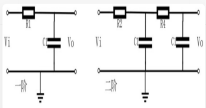
[控制相关] 滤波器1 & 推荐个电子信息相关的网站

szyyy 发表于汽车工程师...

第十三课 基于Rao-Blackwellized粒子滤波器...

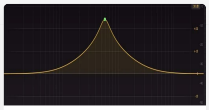
动机至今为止，我们主要解决的是基于特征的SLAM，比如说基于卡尔曼滤波器的SLAM和FastSLAM，在栅格地图方面，我们也学习了已知位姿的地图构建，但利用原始里程计得到的机器人位姿通常会存...

疏影暗香 发表于机器人建图



什么是二阶滤波器？有什么优点？

工程师看海 发表于硬件工程师...



天大教授在美被捕 国产滤波器谈何反扑

与非网