|  |  |
| --- | --- |
| 产品名称 | 密级 |
|  | 机密 |
| 产品版本 | 共 页 |
|  |

pcl::BoundaryEstimation< PointInT, PointNT, PointOutT >算子功能分析说明书

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制 | 赵宇轩 | 日期 | 2021-10-06 |
| 审核 |  | 日期 | yyyy-mm-dd |
| 批准 |  | 日期 | yyyy-mm-dd |



华为技术有限公司

版权所有 侵权必究

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 修订版本 | 修改描述 | 作者 |
| 2021-10-06 | 1.0 | 初稿完成 | 赵宇轩 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[pcl::BoundaryEstimation< PointInT, PointNT, PointOutT >算子功能分析说明书 1](#_Toc10239)

[修订记录 1](#_Toc20034)

[目录 1](#_Toc18835)

[1.1 pcl::BoundaryEstimation< PointInT, PointNT, PointOutT > 2](#_Toc5955)

[1.1.1 功能介绍 2](#_Toc4476)

[1.1.2 使用场景介绍 3](#_Toc21098)

[1.2 bool pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (const pcl::PointCloud<PointInT> &cloud, int q\_idx, const pcl::Indices &indices, const Eigen::Vector4f &u, const Eigen::Vector4f &v, const float angle\_threshold) 5](#_Toc13443)

[1.2.1 接口功能 5](#_Toc17531)

[1.2.2 接口和IR描述 5](#_Toc17146)

[1.2.3 (高性能)实现方案 5](#_Toc2228)

[1.3 bool pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (const pcl::PointCloud<PointInT> &cloud, const PointInT &q\_point, const pcl::Indices &indices, const Eigen::Vector4f &u, const Eigen::Vector4f &v, const float angle\_threshold) 7](#_Toc20115)

[1.3.1 接口功能 7](#_Toc26564)

[1.3.2 接口和IR描述 7](#_Toc3061)

[1.3.3 (高性能)实现方案 7](#_Toc32644)

[1.4 void pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::computeFeature (PointCloudOut &output) 9](#_Toc13859)

[1.4.1 接口功能 9](#_Toc2380)

[1.4.2 接口和IR描述 9](#_Toc27120)

[1.4.3 (高性能)实现方案 9](#_Toc15388)

[1.5 inline void getCoordinateSystemOnPlane (const PointNT &p\_coeff, Eigen::Vector4f &u, Eigen::Vector4f &v) 10](#_Toc8708)

[1.5.1 接口功能 10](#_Toc28918)

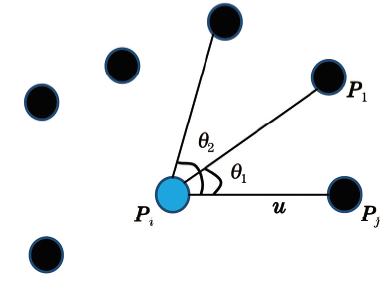
[1.5.2 接口和IR描述 10](#_Toc81)

[1.5.3 (高性能)实现方案 10](#_Toc8986)

## pcl::BoundaryEstimation< PointInT, PointNT, PointOutT >

### 功能介绍

1. 基于法线实现点云边界提取
2. 3D点云的边缘点往往位于点云的最外围，判断点云P的边缘点集的过程如下：
   1. 记点云中的任意点为Pi，对于Pi设置其搜索半径r，并将搜索半径r范围内的邻域点记作N(Pi)，即
   2. 设定曲面方程：
   3. 对于Pi及其对应的邻域点集N(Pi)，计算N(Pi)到曲面的距离di：
   4. 根据Pi及其法向量n，做出切平面Φ，将N(Pi)内的点投影到切平面Φ上，记为N(Pi’)。在N(Pi’)中选取一点Pj’，将n作为w轴
   5. 将u×w作为v轴，以Pi为坐标中心构建局部坐标系，记作(Pi,u,v,w)。
   6. 分别计算集合N(Pi’)中其他点Pj’到点Pi的向量PiPj’与坐标轴u的顺时针夹角Ω=(Θ1,Θ2,Θ3,...Θn),并对相邻夹角两两做差，记差的集合Ω’=(Θ1’,Θ2’,Θ3’,...Θn’)，其中Θn-1’=Θn-Θn-1
   7. 将集合Ω’中的元素降序排列，找出其中最大的夹角Θmax’，当Θmax’大于阈值时（代码中阈值=PI/2），认为该点为边缘点。



1. 该类实现的关键接口
   1. 判断某点是否在给定点集投影到平面上的投影区域的边缘。

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (

       const pcl::PointCloud<PointInT> &cloud, int q\_idx,

       const pcl::Indices &indices,

       const Eigen::Vector4f &u, const Eigen::Vector4f &v,

       const float angle\_threshold)

* 1. 判断某点是否在给定点集投影到平面上的投影区域的边缘。

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (

       const pcl::PointCloud<PointInT> &cloud, const PointInT &q\_point,

       const pcl::Indices &indices,

       const Eigen::Vector4f &u, const Eigen::Vector4f &v,

       const float angle\_threshold)

* 1. 计算点集是否属于平面的边界上，返回包含边界点的点云

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::computeFeature (PointCloudOut &output)

1. 下图是该类的继承关系。

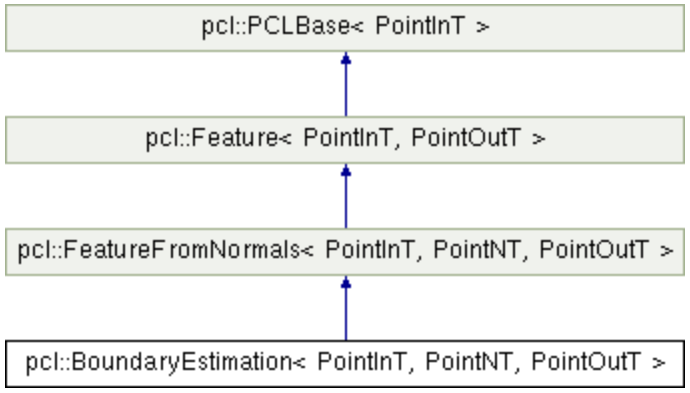


图 1 类的继承关系

### 使用场景介绍

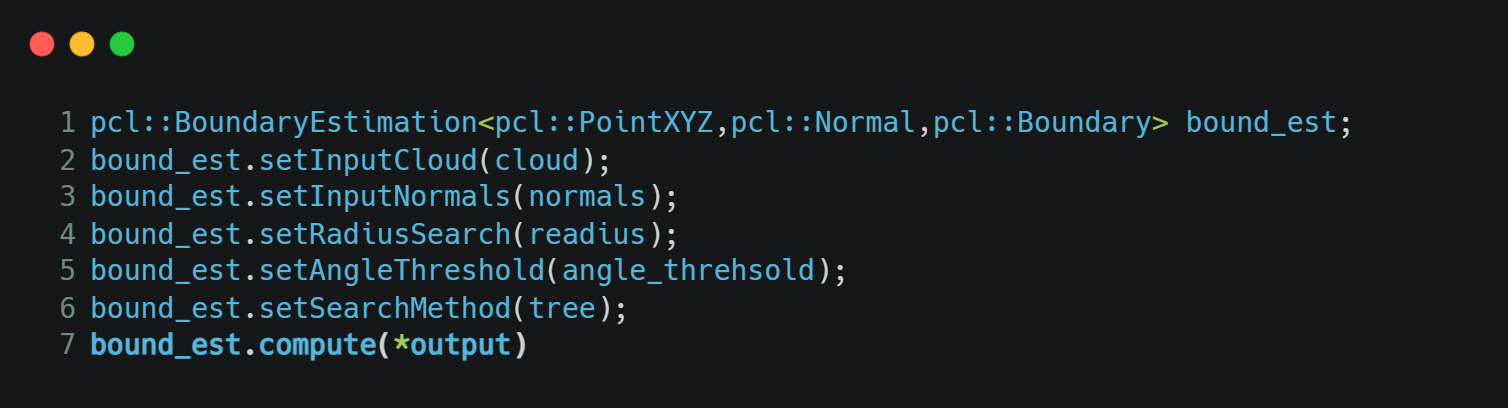


图 2 使用场景

（1）使用场景说明

|  |  |
| --- | --- |
| 行号 | 解释 |
| 1 | 声明一个BoundaryEstimation的实例，记作bound\_est |
| 2~6 | 设置一系列参数 |
| 7 | 调用compute接口，计算输入点云cloud的特征。其中，compute接口是BoundaryEstimation的父类的父类Feature实现的接口。 |

## bool pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (const pcl::PointCloud<PointInT> &cloud, int q\_idx, const pcl::Indices &indices, const Eigen::Vector4f &u, const Eigen::Vector4f &v, const float angle\_threshold)

### 接口功能

判断某点是否在给定点集投影到平面上的投影区域的边缘。

### 接口和IR描述

1. IR原型定义

IR定义：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Classify** | **Name** | **Type** | **Type Range** | **Default\_value** | **Format** |
| isBoundaryPoint | Input | cloud | pcl::PointCloud<PointInT> |  |  |  |
| Input | q\_idx | int |  |  |  |
| Input | indices | pcl::Indices |  |  |  |
| Input | u | Eigen::Vector4f |  |  |  |
| Input | v | Eigen::Vector4f |  |  |  |
| Input | angle\_threshold | float |  |  |  |
| Output | isBoundaryPoint | bool |  |  |  |

1. 算子的实现接口定义

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (const pcl::PointCloud<PointInT> &cloud, int q\_idx, const pcl::Indices &indices, const Eigen::Vector4f &u, const Eigen::Vector4f &v, const float angle\_threshold)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| cloud | pcl::PointCloud<PointInT> | 输入点云 |
| q\_idx | int | 查询点的下标 |
| indices | pcl::Indices | 查询点的临近点集的下标 |
| u | Eigen::Vector4f | u方向 |
| v | Eigen::Vector4f | v方向 |
| angle\_threshold | float | 阈值角度 |

### (高性能)实现方案

|  |  |
| --- | --- |
| 该接口需要用到的Tik接口 | 同接口中调用的其他接口 |
| 无法用Tik接口实现的部分 | 同接口中调用的其他接口 |

行号见源码附件：

|  |  |  |
| --- | --- | --- |
| 行号 | 解释 | Tik接口 |
| 7 | 调用重载的isBoundaryPoint接口 |  |

## bool pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (const pcl::PointCloud<PointInT> &cloud, const PointInT &q\_point, const pcl::Indices &indices, const Eigen::Vector4f &u, const Eigen::Vector4f &v, const float angle\_threshold)

### 接口功能

判断某点是否在给定点集投影到平面上的投影区域的边缘。

### 接口和IR描述

1. IR原型定义

IR定义：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Classify** | **Name** | **Type** | **Type Range** | **Default\_value** | **Format** |
| isBoundaryPoint | Input | cloud | pcl::PointCloud<PointInT> |  |  |  |
| Input | q\_point | PointInT |  |  |  |
| Input | indices | pcl::Indices |  |  |  |
| Input | u | Eigen::Vector4f |  |  |  |
| Input | v | Eigen::Vector4f |  |  |  |
| Input | angle\_thresholded | float |  |  |  |
| Output | isBoundaryPoint | bool |  |  |  |

1. 算子的实现接口定义

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::isBoundaryPoint (const pcl::PointCloud<PointInT> &cloud, const PointInT &q\_point, const pcl::Indices &indices, const Eigen::Vector4f &u, const Eigen::Vector4f &v, const float angle\_threshold)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| cloud | pcl::PointCloud<PointInT> | 输入点云 |
| q\_point | PointInT | 查询点 |
| indices | pcl::Indices | 查询点的临近点集的下标 |
| u | Eigen::Vector4f | u方向 |
| v | Eigen::Vector4f | v方向 |
| angle\_thresholded | float | 阈值角度 |

### (高性能)实现方案

|  |  |
| --- | --- |
| 该接口需要用到的Tik接口 | vec\_mul，vec\_sub，vec\_add |
| 无法用Tik接口实现的部分 | 三角函数 |

行号见源码附件：

|  |  |  |
| --- | --- | --- |
| 行号 | 解释 | Tik接口 |
| 25 | 向量相减 | vec\_sub |
| 29 | 向量点乘、计算反正切 | vec\_mul、vec\_add  tik中似乎没有与三角函数相关的接口 |

## void pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::computeFeature (PointCloudOut &output)

### 接口功能

计算点集是否属于平面的边界上，返回包含边界点的点云。

### 接口和IR描述

1. IR原型定义

IR定义：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Classify** | **Name** | **Type** | **Type Range** | **Default\_value** | **Format** |
| computeFeature | Output | output | PointCloudOut |  |  |  |

1. 算子的实现接口定义

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::computeFeature (PointCloudOut &output)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| output | PointCloudOut | 输出点云 |

### (高性能)实现方案

|  |  |
| --- | --- |
| 该接口需要用到的Tik接口 | 同接口中调用的其他接口 |
| 无法用Tik接口实现的部分 | 同接口中调用的其他接口 |

行号见源码：

|  |  |  |
| --- | --- | --- |
| 行号 | 解释 | Tik接口 |
| 17 | 调用searchForNeighbors接口 | 尚未找到searchForNeighbors接口的源码 |
| 27 | 调用getCoordinateSystemOnPlane接口 |  |
| 30 | 调用isBoundaryPoint接口 |  |
| 52 | 调用isBoundaryPoint接口 |  |

## inline void getCoordinateSystemOnPlane (const PointNT &p\_coeff, Eigen::Vector4f &u, Eigen::Vector4f &v)

### 接口功能

获取由某平面法线所定义的u-v-n坐标系。

（该函数是内联函数，定义在boundary.h文件中）

### 接口和IR描述

1. IR原型定义

IR定义：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Classify** | **Name** | **Type** | **Type Range** | **Default\_value** | **Format** |
| getCoordinateSystemOnPlane | Input | p\_coeff | PointNT |  |  |  |
| Output | u | Eigen::Vector4f |  |  |  |
| Output | v | Eigen::Vector4f |  |  |  |

1. 算子的实现接口定义

pcl::BoundaryEstimation<PointInT, PointNT, PointOutT>::computeFeature (PointCloudOut &output)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| p\_coeff | PointNT | 平面系数（包含平面法线） |
| u | Eigen::Vector4f | 作为结果输出的轴u |
| v | Eigen::Vector4f | 作为结果输出的轴v |

### (高性能)实现方案

|  |  |
| --- | --- |
| 该接口需要用到的Tik接口 | vec\_mul，vec\_sub，vec\_add |
| 无法用Tik接口实现的部分 | 无 |

行号见源码：

|  |  |  |
| --- | --- | --- |
| 行号 | 解释 | Tik接口 |
| 17 | 计算与当前向量正交的单位向量 | vec\_sub、vec\_mul |
| 7 | 向量叉乘 | vec\_mul、vec\_sub、vec\_add |