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

pcl::ShapeContext3DEstimation< PointInT, PointNT, PointOutT >算子功能分析说明书

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制 | 赵宇轩 | 日期 | 2021-09-25 |
| 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 |
| 2021-09-25 | 1.0 | 初稿完成 | 赵宇轩 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[pcl::ShapeContext3DEstimation< PointInT, PointNT, PointOutT >算子功能分析说明书 1](#_Toc19672)

[修订记录 1](#_Toc25855)

[1.1 pcl::ShapeContext3DEstimation< PointInT, PointNT, PointOutT > 2](#_Toc22202)

[1.1.1 功能介绍 2](#_Toc6186)

[1.1.2 使用场景介绍 2](#_Toc102)

[1.1.3 接口和IR描述 3](#_Toc12478)

[1.1.4 (高性能)实现方案 4](#_Toc22656)

## **pcl::ShapeContext3DEstimation< PointInT, PointNT, PointOutT >**

### 功能介绍

1. 该类实现了3D形状内容描述子(3D shape context descriptor)算法。可用于三维模型的识别、检索等领域。
2. 该类实现的关键接口
   1. 计算初始化接口

bool pcl::ShapeContext3DEstimation<PointInT, PointNT, PointOutT>::initCompute ()

* 1. 计算点云中某点的描述子

bool pcl::ShapeContext3DEstimation<PointInT, PointNT, PointOutT>::computePoint (

     std::size\_t index, const pcl::PointCloud<PointNT> &normals, float rf[9], std::vector<float> &desc)

* 1. 计算点云的特征

void pcl::ShapeContext3DEstimation<PointInT, PointNT, PointOutT>::computeFeature (PointCloudOut &output)

1. 下图是该类的继承关系。

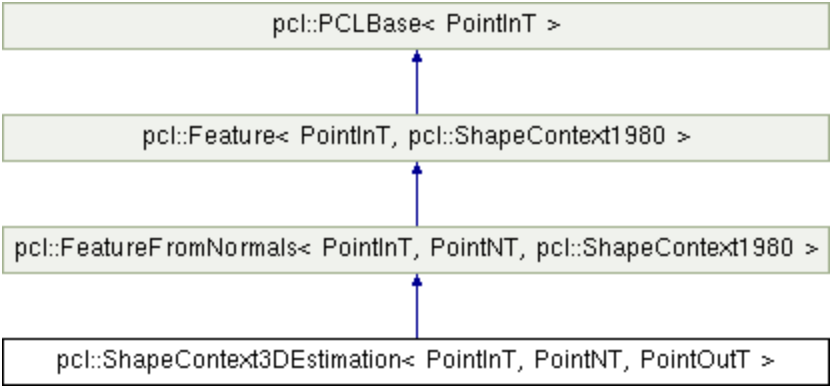


图 1 类的继承关系

### 使用场景介绍

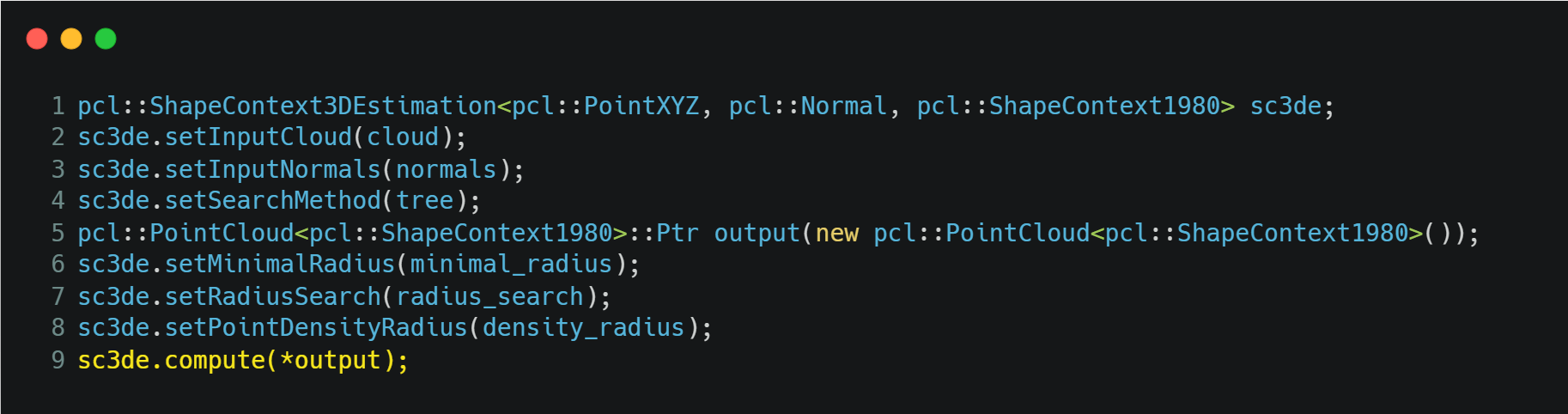


图 2 使用场景

（1）使用场景说明

|  |  |
| --- | --- |
| 行号 | 解释 |
| 1 | 声明一个ShapeContext3DEstimation的实例，记作sc3de |
| 2 | 设置输入的点云，记作cloud |
| 3 | 设置输入的法线，记作，normals |
| 4 | 设置使用kd-tree搜索邻近点 |
| 5 | 声明一个PointCloud的指针，记作output |
| 6~8 | 为sc3de对象设置一系列参数 |
| 9 | 调用compute接口，计算输入点云cloud的特征。其中，compute接口是ShapeContext3DEstimation的父类的父类Feature实现的接口。 |

### 接口和IR描述

1. IR原型定义

IR定义：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Classify** | **Name** | **Type** | **Type Range** | **Default\_value** | **Format** |
| ShapeContext3DEstimation<PointInT, PointNT, PointOutT>::computePoint | Input | index | std::size\_t |  |  |  |
| Input | normals | pcl:PointCloud<PointNT> |  |  |  |
| Input | rf | float[9] |  |  |  |
| Input | desc | std::vector<float> |  |  |  |
| Input | indices\_ | PCLBase<PointInT> |  |  |  |
| Input | search\_radius | double |  |  |  |
| Input | input\_ | PCLBase<PointInT> |  |  |  |
| Input | surface | PointCloudInConstPtr |  |  |  |
| Input | elevation\_bins\_ | std::size\_t |  |  |  |
| Input | radius\_bins\_ | std::size\_t |  |  |  |
| Output | desc | std::vector<float> |  |  |  |
| Output | y1 | bool |  |  |  |

1. 算子的实现接口定义

pcl::ShapeContext3DEstimation<PointInT,PointNT,PointOutT>::computePoint(std::size\_t index, const pcl::PointCloud<PointNT> &normals, float rf[9], std::vector<float> &desc)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| index | std::size | 需要计算描述子的查询点的下标 |
| normals | const pcl::PointClout<PointNT> | 一个指向点云的法向量集合的指针 |
| rf[9] | float | 参考系 |
| desc | std::vector<float> | 用于存放计算好的描述子 |

### (高性能)实现方案

pcl::ShapeContext3DEstimation<PointInT,PointNT,PointOutT>::computePoint(std::size\_t index, const pcl::PointCloud<PointNT> &normals, float rf[9], std::vector<float> &desc)

源码见附件：

|  |  |  |
| --- | --- | --- |
| 行号 | 解释 | Tik接口 |
| 6~7 | 用指向float数组的指针初始化Map |  |
| 11~12 | 定义两个变量（nn\_indices用于存储临近点的下标；nn\_dists用于存储临近点的距离） |  |
| 13 | 调用searchForNeighbors计算临近点的数量（找不到这个函数的源码？），共有3个返回值（neighb\_cnt返回临近点的数量；nn\_indices储存所有临近点的下标；nn\_dists储存所有查询点到临近点的距离） | for\_range |
| 14~19 | 接口的第一个出口。当neighb\_cnt==0时，置descriptor=NaN，rf=0.f，return false |  |
| 21~22 | 得到距查询点最近的点的下标 | for\_range, scalar\_min |
| 25 | 获取查询点（即同心球的球心） | vec\_sub |
| 28~33 | 接口的第二个出口。当查询点的法线不有限时，置descriptor=NaN，rf=0.f，return false |  |
| 34 | 获取查询点的法线 |  |
| 37~39 | 对x轴的全部三个元素用随机数初始化 |  |
| 40~45 | 计算x轴的坐标（查询点(0,0,0)应该作为向量的头）（猜测）  假设x轴与法线正交，X1X2+Y1Y2+Z1Z2=0反推出x轴的坐标(X1,Y1,Z1)  if：法线的z坐标不为0.f  then：Z1=-(X1X2+Y1Y2)/Z2  if：法线的y坐标不为0.f  then：Y1=-(X1X2+Z1Z2)/Y2  if：法线的x坐标不为0.f  then：X1=-(Y1Y2+Z1Z2)/X2 | vec\_mul,  vec\_add,  vec\_rec,  vec\_sub |
| 47 | 对x轴做向量标准化 | vec\_rec |
| 50 | 检查经过计算后的x轴是否与查询点的法线正交 | vec\_mul,  vec\_add,  vec\_sub |
| 53 | 计算y轴：y轴=法向量与x轴做向量积（cross product），由于x轴与法线正交，他们两个再做×积得到y轴，则x、y与法线皆两两正交。 | vec\_mul,  vec\_sub,  vec\_add |
| 56~115 | 遍历所有临近点 | for\_range |
| 58~59 | 首先排除距离为0的临近点 |  |
| 61 | 获取当前临近点的坐标 |  |
| 65 | 获取查询点与当前临近点的距离r（这里用了平方根，尚不清楚具体意义） | scalar\_sqrt |
| 68 | 定义一个变量proj，表示临近点在法平面的投影 |  |
| 69~70 | 计算临近点在法平面的投影proj | vec\_sub,  vec\_mul,  vec\_add |
| 73 | 对投影向量proj做标准化处理 | vec\_rec |
| 76~78 | 计算投影proj与x轴的夹角phi∈[0,360]  pcl::rad2deg：将弧度转换成角度  std::atan2：返回y/x的反正切值，返回值以弧度表示 | vec\_mul,  vec\_sub,  vec\_add,  tik中似乎没有反三角函数和三角函数 |
| 80~83 | 计算临近点与法线（z轴）的夹角theta∈[0,180]  std::acos：反余弦函数，返回值以弧度表示 | vec\_mul,  vec\_sub,  vec\_add,  tik中似乎没有反三角函数和三角函数 |
| 86 | 从所有半径区间(radii\_interval\_)中找出第一个不小于r（第65行定义）的元素，返回值是指向该元素的iterator，记作rad\_min | for\_range, scalar\_min |
| 87 | 从所有theta区间(theta区间的划分与俯仰角elevation相关)中找出第一个不小于theta(在第83行定义)的元素，返回值是指向该元素的iterator，记作theta\_min | for\_range, scalar\_min |
| 88 | 从左右phi区间(phi区间的划分与转动叫azimuth相关)中找出第一个不小于phi(phi表示一个实际的转动角的值，在第77~78行定义并计算出)的元素，返回值是指向该元素的iterator，记作phi\_min | for\_range, scalar\_min |
| 91~93 | 计算rad\_min,theta\_min,phi\_min所在区间的下标 | vec\_sub |
| 96 | 定义一个储存当前临近点的临近点的下标的容器neighbour\_indices |  |
| 97 | 定义一个储存当前临近点距其他临近点的距离的容器neighbour\_distances |  |
| 98 | 调用searchForNeighbour方法，由3个输入、3个输出。3个输入分别是（\*surface：一个静态点云对象指针，尚不清楚具体含义；nn\_indices[ne]：当前临近点的下标；point\_density\_radius：当前临近点搜索其临近点的搜索半径），3个输出分别是（neighbour\_indices：存储当前临近点搜索到的临近点的下标；neighbour\_distances：存储当前临近点距搜索到的临近点的距离； point\_density：返回当前临近点搜索到的临近点的数量） | for\_range |
| 100 | 若当前临近点查询不到point\_density\_radius范围内的其他点，则直接查询下一个临近点。 |  |
| 103 | 计算w，尚不知道具体意义 | vec\_rec,  vec\_mul,  vec\_add |
| 106 | 检查w是否大于等于0 |  |
| 107~110 | 检查w是否合法（不合法的情况：w==infinity；w==NaN） |  |
| 112 | 更新查询点的描述子：descriptor[l][k][j] += w | vec\_mul,  vec\_add |
| 114 | 检查更新后的查询点的描述子descriptor是否大于等于0 |  |
| 118 | 将rf置0，尚不清楚具体意义 |  |
| 119 | 接口运行正确时的出口。return true |  |