|  |  |
| --- | --- |
| 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 |  | 日期 | yyyy-mm-dd |
| 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 |
| yyyy-mm-dd | 1.0 | 初稿完成 |  |
| yyyy-mm-dd | 1.1 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[pcl::ShapeContext3DEstimation< PointInT, PointNT, PointOutT >算子功能分析说明书 1](#_Toc9289)

[修订记录 1](#_Toc10009)

[1.1 xxx(原始算子名) 2](#_Toc26433)

[1.1.1 功能介绍 2](#_Toc20910)

[1.1.2 使用场景介绍 2](#_Toc17324)

[1.1.3 接口和IR描述 2](#_Toc1044)

[1.1.4 (高性能)实现方案 3](#_Toc16677)

[1.2 grid\_sampler 3](#_Toc2754)

[1.2.1 功能介绍 3](#_Toc30832)

[1.2.2 接口和IR描述 7](#_Toc10504)

[1.2.3 输入 7](#_Toc24264)

[1.2.4 处理 7](#_Toc2463)

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

### 功能介绍

此处，主要介绍该算子的原始功能及其接口和参数信息(可截图)。

对于**复杂难懂**的计算类算子，必须给出细节详细说明，尽量给出示意图、公式和示例代码等。

### 使用场景介绍

最好给出1-2个示例，简答易懂的可酌情略过。

### 接口和IR描述

此处包括两部分，第一个是算子的IR原型定义，第二个是算子实现接口定义(python)。

1. IR原型定义

IR定义：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Op** | **Classify** | **Name** | **Type** | **Type Range** | **Default\_value** | **Format** |
| AaBbCc | Input | x1 | tensor | BasicType |  | 1D |
| Input | x2 | tensor | BasicType |  | ND |
| Output | y1 | tensor | IndexNumberType |  | 1D |
| Output | y2 | tensor | IndexNumberType |  | 1D |
| Attr | xx | int |  | 1 |  |

注意：

1. Op名字的命名必须满足大驼峰，不含下划线；比如AaBbCc
2. Classify分为Input（必选输入）、OptionalInput（可选输入）、DynamicInput（必选动态输入）、Output（必选输出）、 OptionalOutput（可选输出）、DynamicOutput（必选动态输出）、Attr（可选属性）、RequiredAttr（必选属性）。如何区分输入和属性呢？参数的size固定或者数值，type固定，它也可定义为属性。
3. 针对Name栏，如果有多个输入，可定义为x1，x2…；同理，如果有多个输出，可定义为y1,y2,…；但如果有特定含义且有习惯缩写的，优先采用，且遵循命名缩写规则。
4. 针对Type栏，非属性类基本上都是Tensor
5. 针对Type Range栏，参看算子IR定义字典V1.0表格DType&Format页
6. 针对Format栏，给出该输出输入的维度信息。一般给出ND(1D/2D)即可，如果输入有特定明确格式要求需明确，比如NCHW，NHWC，HWCN。
7. 算子的实现接口定义

模板如下：

def xx(\_yy\_zz) (x1, x2, y1, y2, default\_value=default\_value, kernel\_name="xx\_yy\_zz"):

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| x1 | dict | 针对参数x1的说明，比如输入图片image1，及其他信息 |
| x2 | dict | 针对参数x2的说明，比如输入图片image2，及其他信息 |
| y1 | list\_int | 针对输出tensor y1的说明。可能有shape和dtype |
| y2 | int | 针对输出tensor y2的说明 |
| default\_name | bool | 针对默认字段参数的说明，包括默认取值 |
| kernel\_name | str | kernel的名称，含义清晰且能表达该算子的含义即可。 |

注意：

1. xx(\_yy\_zz)命名必须满足小驼峰(加下划线)
2. 函数参数，遵循小驼峰(加下划线)，且有实际的含义，比如input\_x，intput\_y，output等。可根据IR中的Name延续过来，但必须满足命名规则。
3. 类型，根据输入参数实际给出，比如dict，tuple，list\_int，int, bool等
4. 说明，给出每一个参数的含义，包括尽量shape和type信息。如果有默认值，需要给出。

### (高性能)实现方案

基于华为昇腾芯片，描述一下它的实现方案。简单的算子可省略。

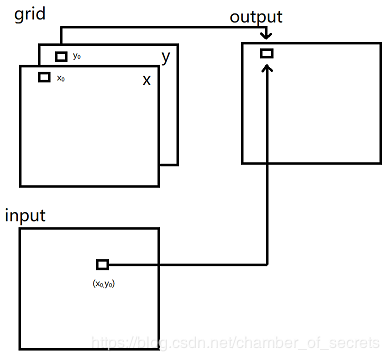
## grid\_sampler

### 功能介绍

该算子源于空间变换网络(Spatial Transformation Networks)，它包括三个模块：Localisation Network、grid generator(affine grid)、grid sampler.

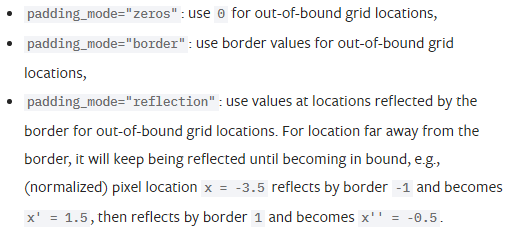
grid\_sampler(Tensor input, Tensor grid, int interpolation\_mode, int padding\_mode, bool align\_corners)

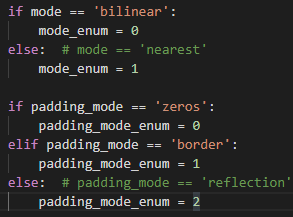
**原始功能**：Given an input and a flow-field grid, computes the output using input values and pixel locations from grid。提供一个input的Tensor以及一个对应的flow-field网格(比如光流，体素流等)，然后根据grid中最后一维提供的坐标信息(这里指input中pixel的坐标)，将input中对应位置的像素值填充到grid指定的位置，得到最终的输出。它一般是跟放射网格生成器affine\_grid\_generator配合使用。



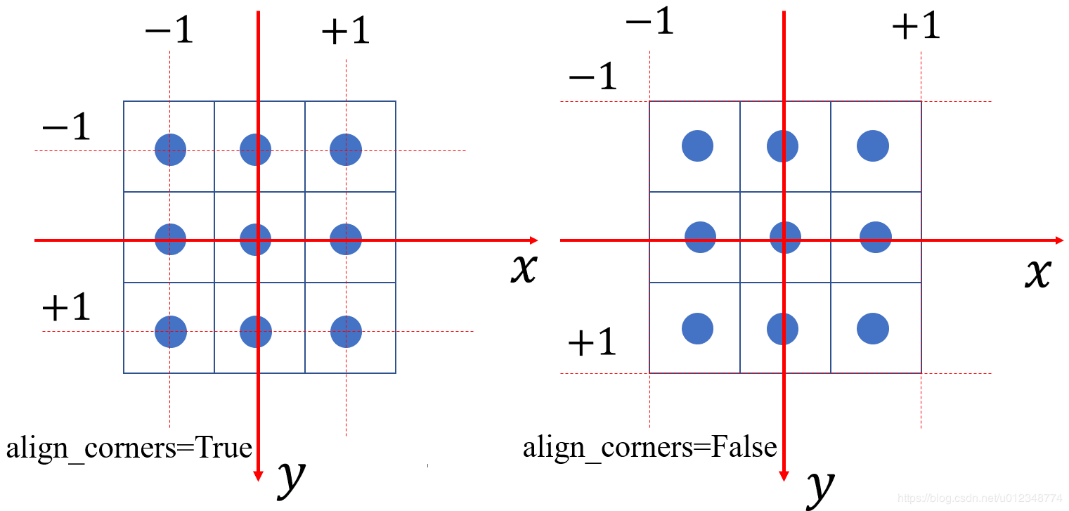
输入参数：

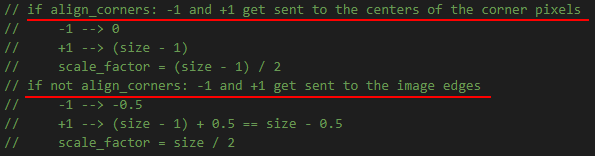
* **input**: (Tensor)the input image or feature-map. For 4-D case, the shape is NCHW; For 5-D case, the shape is NCDHW.
* **grid:** (Tensor)specifies the sampling pixel locations normalized by the input spatial dimensions. 因此，它的值范围主要在[-1, 1]之间。例如，如果x=-1,y=-1,那么它表示输入的左上角；如果x=1,y=1，那么它表示为输入的右下角。
* **interpolation\_mode:** (int)计算输出值的插值模式。有最近邻nearest:1和双线性bilinear:0。默认是bilinear. 枚举对应关系：pytorch-1.5.0\torch\nn\functional.py
* **padding\_mode:**(int)针对网格grid外数值的padding模式。





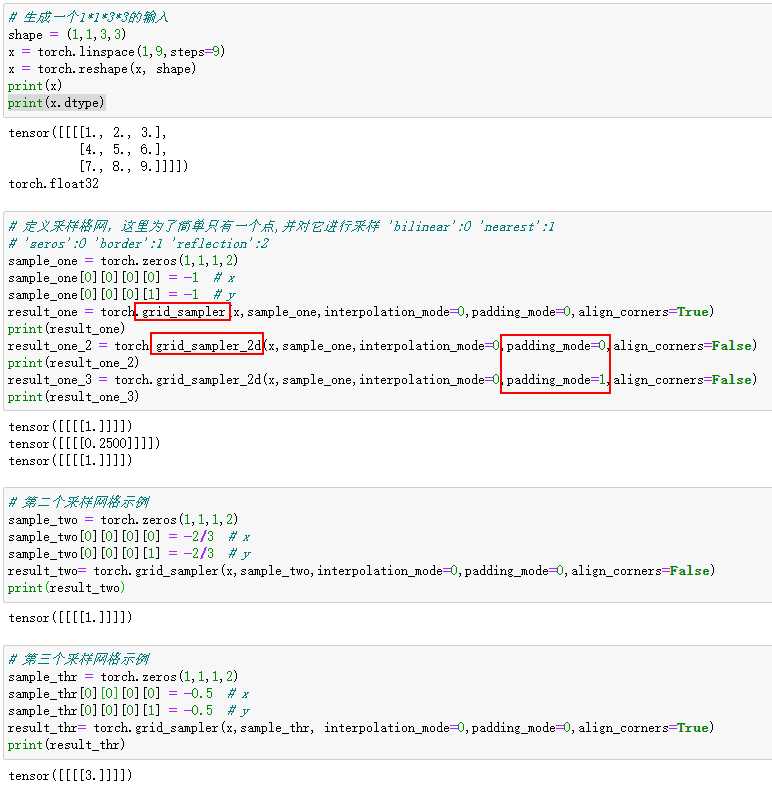
* **align\_corner:**(bool,optional)在几何学上，认为输入的像素点是小正方形而不是一个点。如果它设置为True，极限值extrema（也即-1和1）是指输入corner pixels的中心点。如果设置为False，是指输入corner pixels的corner points。这个参数值跟interpolate()的用法一样。当align\_corners=True时，坐标归一化范围是图像四个角点的中心;当align\_corners=False时，坐标归一化范围是图像四个角点靠外的角点;为了更好的说明这个情况，我画了一个大小为3×33×33×3影像进行说明，如下，其中每一个方格代表一个像素，并且像素坐标在方格中央。后面有具体代码示例给出详细的测试说明。





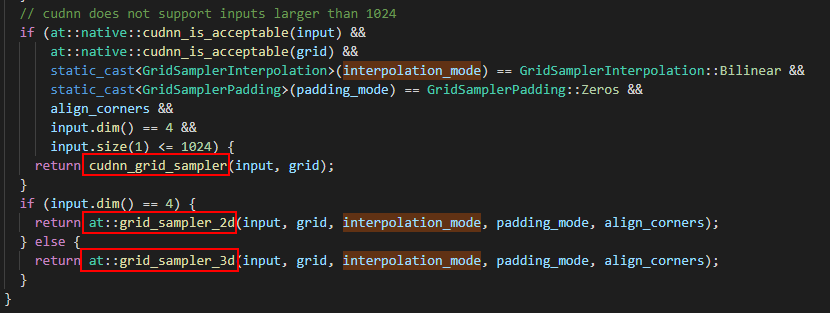
Note注意：grid\_sampler是cudnn\_grid\_sampler,grid\_sampler\_2d,grid\_sampler\_3d的总入口，它对输入参数的shape先做校验，然后在派发dispatch给其中一个区执行。

Pytorch算子功能示例代码：



在采样网格1中，在result\_one\_2中当padding\_mode=zero,align\_corners=False时，左上角位置(-1,-1)最后的输出0.25是对像素点1和bound外padding为0的双线性插值；在result\_one\_3中当padding\_mode=’border’,align\_corners=True时，左上角位置(-1,-1)最后的输出1是对像素点1和bound外padding为1的双线性插值。

实现路径：pytorch-1.5.0\aten\src\ATen\native\GridSampler.cpp



|  |  |  |  |
| --- | --- | --- | --- |
| 入口函数 | 分支函数 | 需要的条件 | 备注 |
| grid\_sampler | cudnn\_grid\_sampler | 1.interpolation\_mode=’bilinear’  2.padding\_mode=’zeros’  3.align\_corners=True  4.input.dim()=4  5.intput.size(1)<=1024，也即是Channel<=1024 threads block | 最后，它调用cudnn算子cudnnSpatialTfSamplerForward。  它的条件限制，也是因为cuDNN接口条件原因。 |
| grid\_sampler\_2d | input.dim()=4 | 如果不满足cuDNN条件，针对4D情况，调用grid\_sampler\_2d |
| grid\_sampler\_3d | 其他 | 如果不满足cuDNN条件，针对5D情况，调用grid\_sampler\_3d |

### 接口和IR描述

无IR定义和描述

### 输入

见接口定义

### 处理

增加一个python适配层。把grid\_sampler\_2d和grid\_sampler\_3d的实现囊括进去，用统一的一套接口和参数。