目录

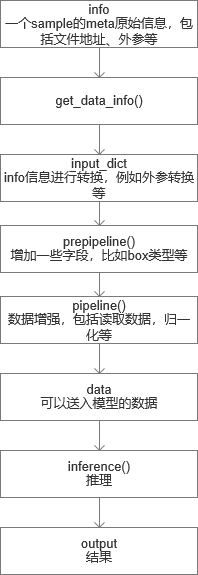
[简介 2](#_Toc181721459)

[Info数据结构构成 2](#_Toc181721460)

[cam字典数据结构 3](#_Toc181721461)

[Output返回结构 5](#_Toc181721462)

## 简介



图表 1 单笔自主数据的构成与输入模型的流程图

## Info数据结构构成

Info数据包含了一个sample中激光pcd路径、时间戳、图片路径和内外参等数据，可以作为civ数据输入到后续处理流程。

通过generate\_info函数，我们可以导入公司采集的传感器的数据，从而生成符合数据流的info

generate\_info函数参数列表如表格 1所示

表格 1 generate\_info函数参数列表

|  |  |  |  |
| --- | --- | --- | --- |
| 成员 | 含义 | 类型 | 举例 |
| sample\_token | Sample的token标识符 | Str | 'ca9a282c9e77460f8360f564131a8af5' |
| lidar\_path | 激光雷达的pcd文件地址 | Str | 'data/v1.0-mini/samples/LIDAR\_TOP/n015-2018-07-24-11-22-45+0800\_\_LIDAR\_TOP\_\_1532402927647951.pcd.bin' |
| sweeps |  | list | [] |
| timestamp | 时间戳 | Int | 1532402927647951 |
| cams | 多个摄像头的信息组成的字典 | Dict | {  ‘CAM\_FRONT’:  ‘CAM\_FRONT\_RIGHT’:  ‘CAM\_BACK\_RIGHT’:  ‘CAM\_BACK’:  ‘CAM\_BACK\_LEFT’:  ‘CAM\_FRONT\_LEFT’:  } |

其中cams是多个多个cam字典构成的字典，单个cam参数的定义在段落0中说明

实际代码用例如下，

|  |
| --- |
| # 构成一个civ的sample的info      info = generate\_info(          sample\_token='ca9a282c9e77460f8360f564131a8af5',          lidar\_path='data/v1.0-mini/samples/LIDAR\_TOP/n015-2018-07-24-11-22-45+0800\_\_LIDAR\_TOP\_\_1532402927647951.pcd.bin',          sweeps="dasdfa",          timestamp=1212321,          cams=cams      ) |

## cam字典数据结构

对于车载多摄像头，每个摄像头输入参数的cam是一个字典，字典的定义如表格 2定义

表格 2 单个cam的字典数据结构与例子

|  |  |  |  |
| --- | --- | --- | --- |
| 成员 | 含义 | 类型 | 举例 |
| data\_path | 摄像头拍摄的图片文件路径 | Str | "data/v1.0-mini/samples/CAM\_FRONT/n015-2018-07-24-11-22-45+0800\_\_CAM\_FRONT\_\_1532402927612460.jpg" |
| type | 摄像头类型 | Str | "CAM\_FRONT" |
| timestamp | 时间戳 | Str | 1532402927612460 |
| sensor2lidar\_rotation | 摄像头坐标系到激光坐标系的旋转矩阵 | np.array(3,3) | np.array(                  [                      [0.99997025, 0.00685271, -0.00354221],                      [0.00340737, 0.01958963, 0.9998023],                      [0.00692074, -0.99978462, 0.0195657],                  ]              ) |
| sensor2lidar\_translation | 摄像头坐标系到激光坐标系的平移向量 | np.array(3) | np.array(                [-0.016138239780634223, 0.43552528694590364, -0.3206717867872868]              ), |
| cam\_intrinsic | 摄像头内参 | np.array(3,3) | np.array(                  [                      [1.26641720e03, 0.00000000e00, 8.16267020e02],                      [0.00000000e00, 1.26641720e03, 4.91507066e02],                      [0.00000000e00, 0.00000000e00, 1.00000000e00],                  ]              ) |

实际例子如下

|  |
| --- |
| "CAM\_FRONT": {              "data\_path": "data/v1.0-mini/samples/CAM\_FRONT/n015-2018-07-24-11-22-45+0800\_\_CAM\_FRONT\_\_1532402927612460.jpg",              "type": "CAM\_FRONT",              "timestamp": 1532402927612460,              "sensor2lidar\_rotation": np.array(                  [                      [0.99997025, 0.00685271, -0.00354221],                      [0.00340737, 0.01958963, 0.9998023],                      [0.00692074, -0.99978462, 0.0195657],                  ]              ),              "sensor2lidar\_translation": np.array(                  [-0.016138239780634223, 0.43552528694590364, -0.3206717867872868]              ),              "cam\_intrinsic": np.array(                  [                      [1.26641720e03, 0.00000000e00, 8.16267020e02],                      [0.00000000e00, 1.26641720e03, 4.91507066e02],                      [0.00000000e00, 0.00000000e00, 1.00000000e00],                  ]              ),          }, |

## Output返回结构

网络输出的内容又三种，分别是boxes\_3d,score\_3d和labels\_3d

FUTR3d网络默认会输出300个可能对象，我们需要通过对score\_3d的置信度设置一个阈值，最终输出高于阈值的对象

表格 3 output数据结构

|  |  |  |  |
| --- | --- | --- | --- |
| 成员 | 含义 | 类型 | 举例 |
| boxes\_3d | 检测到的对象的三维框 | LIDARInstance3DBoxes |  |
| score\_3d | 检测到的物体的置信度 | Tensor | 维度300  tensor([0.9268, 0.8631, 0.8447, 0.8371,…]) |
| labels\_3d | 检测到的物体的分类标签 | Tensor | 维度300  tensor([5, 5, 5, 1, 5, 5, 5, 5,…]) |

对于网络输出的output中的boxes\_3d，属于LIDARInstance3DBoxes,其定义如下

表格 4 LIDARInstance3DBoxes 数据结构

|  |  |  |  |
| --- | --- | --- | --- |
| 成员 | 含义 | 类型 | 维度举例 |
| Bev | 2D BEV box of each box with rotation | Tensor | 300x5 |
| nearest\_bev | 2D BEV box of each box without rotation | Tensor | 300x5 |
| Bottom center | center of each box | Tensor | 300x3 |
| Bottom\_height |  | Tensor | 300 |
| center | center of each box | Tensor | 300x3 |
| Corners |  | Tensor | 300x8x3 |
| Height | height of each box | Tensor | 300 |
| Top\_height | top height of each box | Tensor | 300 |
| Volume | volume of each box | Tensor | 300 |
| Yaw | heading | Tensor | 300 |

## 最终输出json

最终的输出也以json文件的形式保存，json的形式如

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | center(x, y, z) | size(dx, dy, dz) | rotation(rot) | label | score |
| 1 | [1.0, 2.0, 0.0] | [4.0, 5.0, 2.0] | [0.5] | 1 | 0.95 |
| 2 | [2.5, 3.0, 0.1] | [3.0, 4.0, 1.5] | [0.4] | 2 | 0.88 |

|  |
| --- |
| [      {          "center": [              7.967964172363281,              28.03937530517578,              -0.9864397048950195          ],          "size": [              0.5959441661834717,              2.060821056365967,              1.0682649612426758          ],          "rotation": -3.0433590412139893,          "label": 5,          "score": 0.9267691373825073      },      {          "center": [              10.483543395996094,              15.859001159667969,              -1.675168514251709          ],          "size": [              0.6759464740753174,              1.9952363967895508,              1.0604634284973145          ],          "rotation": 3.064009428024292,          "label": 5,          "score": 0.8631176948547363      },  ] |