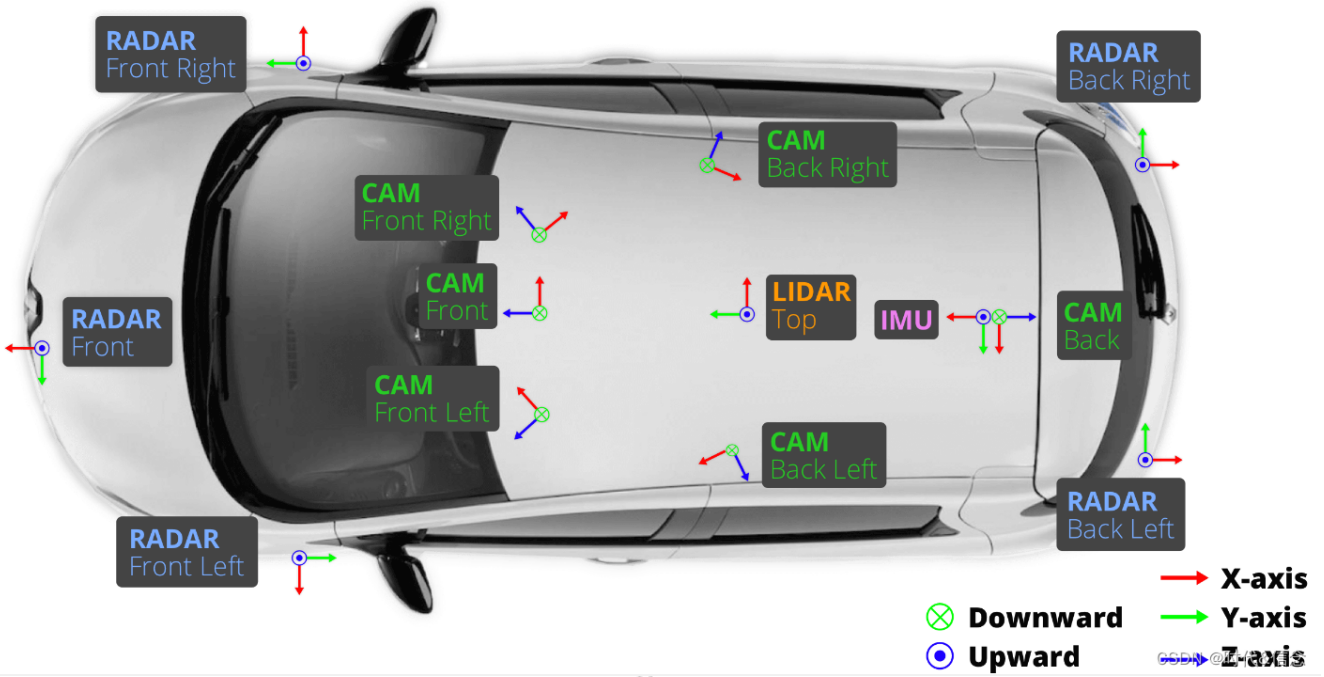
# nuScenes数据集介绍

nuScenes采集设备一共配备了6个相机、1个LiDAR、5个RADAR，如下图所示。值得注意的是，相比KITTI和Waymo，nuScenes提供了**360度的相机视野**，所以该数据集可以用来做360度场景的多传感器融合。

注意：nuScenes的内外参链接：各数据集-传感器配置及内外参




各传感器参数如下表：

（1）6个相机的分辨率都是1600x900，除了背后的相机FOV为110度，其他的5个相机的FOV为70度，前置相机和侧面的相机的视野中线角度为55度。也就是说相机覆盖了360度，但是会有重叠部分。相机的采集速率是12Hz。

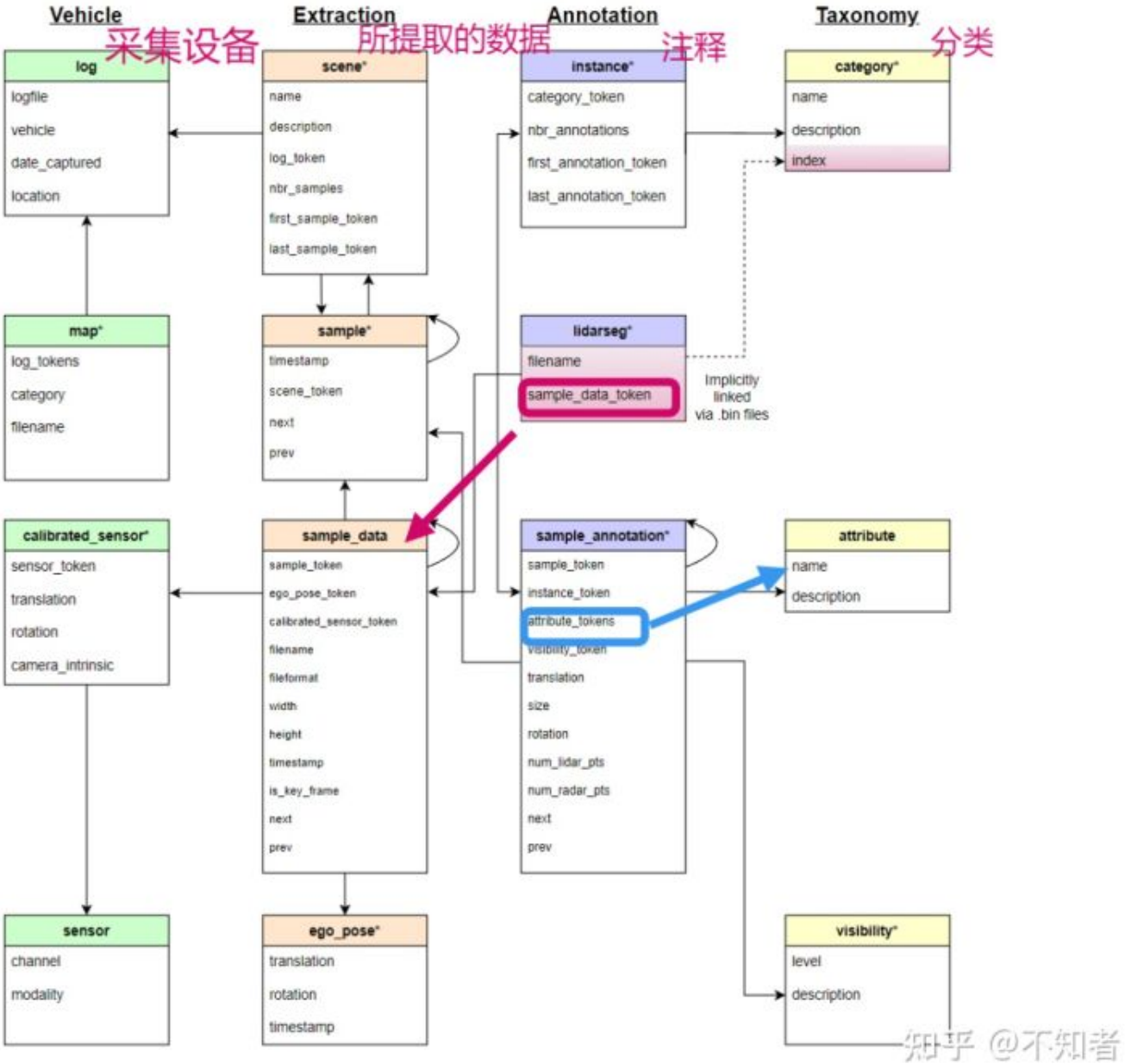
（2）用了一个32线的LiDAR。采集速率是20Hz。注意到nuScenes的采集速度是很快的(KITTI是2Hz，Waymo是10Hz)，所以该数据集可以用**点云稠密化**来做增强(下面会说)。

（3）5个毫米波雷达测试距离≤250m范围，采集速率是77GHz，FMCW（发射调频连续波），13Hz捕获频率，±0.1km / h的精度。

（4）全球定位系统，IMU惯导，航姿参考系统（AHRS）。0.2度航向，0.1度横滚/俯仰，20mm RTK定位，1000Hz更新速率。

## 一、数据集介绍

数据集的结构由以下表格展示。




从左到右四列分别是Vehicle（采集数据所用的交通工具）、Extraction（所采集的对象）、Annotation（标注）和Taxonomy（分类）。

从上到下各行表示各种级别的对象，以Extraction为例，一个scene表示从日志中提取的20秒长的连续帧序列（A scene is a 20s long sequence of consecutive frames extracted from a log.）而一个sample表示一个2Hz的带注释的关键帧（A sample is an annotated keyframe at 2 Hz.）也就是从一个scene中提取出的一帧。而sample_data则更进一步的表示在某时刻（timestamp）的传感器数据，例如图片，点云或者Radar等等。每一个sample_data都与某个sample相联系，如果其is_key_frame值为True，那么它对应的时间戳应当与它所对应的关键帧非常接近，它的数据也会与该关键帧时刻的数据较为接近。

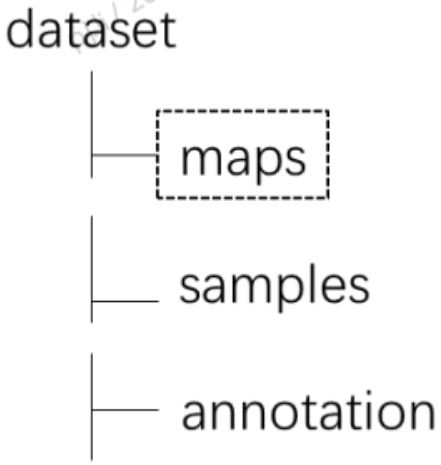
由此可以发现，一个上一级的对象可以包含多个下一级的对象，或者说表示更加广义层面的信息。例如一个scene中可以有多个sample（scene的一个属性为nbr_samples=number of samples）。又比如sample和sample_data的关系，sample只是表示某个采样的时刻，而sample_data则表示该时刻具体传感器采样得到了什么数据，在一个sample对应的关键帧附近，多次采集了各时间戳的sample_data。

| nuScenes dataset | | | |
|---|---|---|---|
| 数据文件列表 | Vehicle | log | 采集数据的相关日志，包含车辆信息、所采集的数据（类型？）、采集地点等信息 |
| | | map | 地图数据，自上而下获得。包含map_category、filename：存储map_mask掩码数据的bin file地址 |
| | | calibrated_sensor | 一个特定传感器在一个特定车辆上的矫正参数。包含translation、rotation、camera_intrinsic（内在相机校准） |

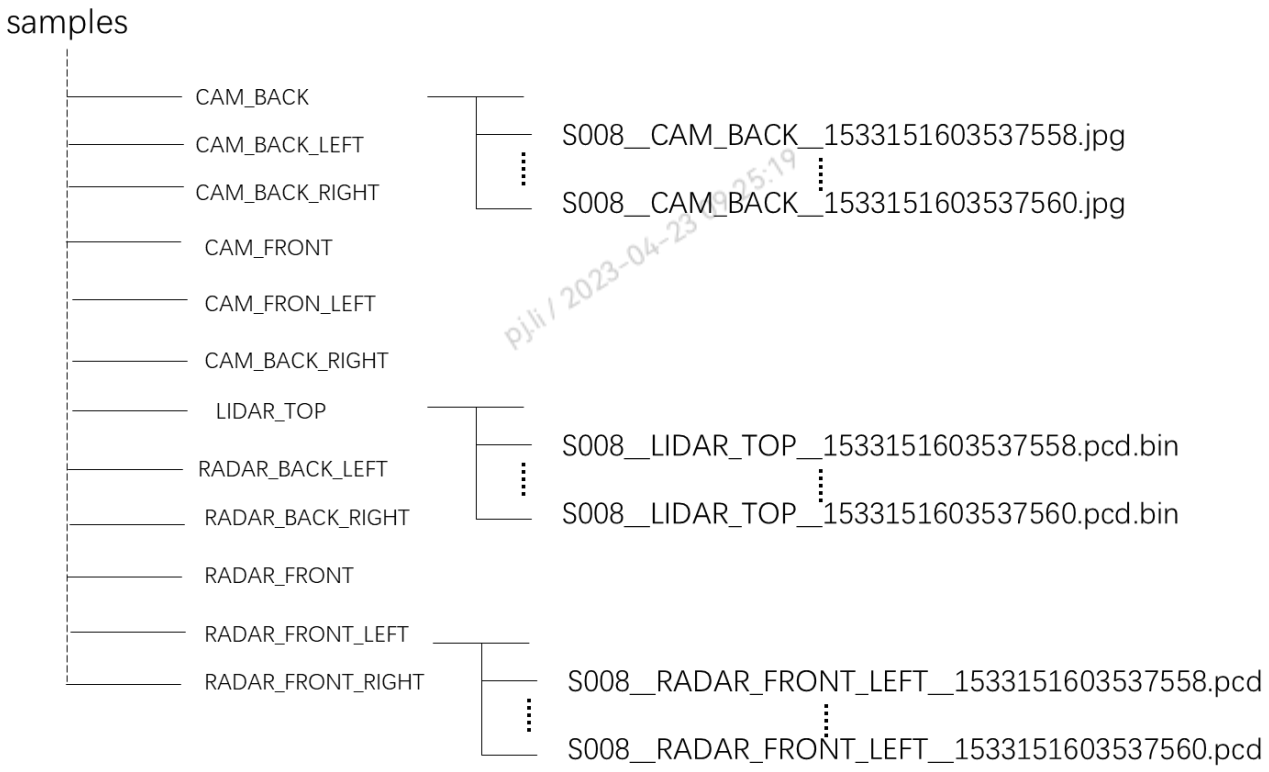
| | | | |
|---|---|---|---|
| | | sensor | 传感器。包含频道（channel）、modality（{camera，lidar，radar}） |
| | Extraction | scene | 采集场景 |
| | | sample | 一个2Hz的带注释的关键帧，也就是从一个scene中提取出的一帧。包含时间戳、对应的scene_token、下一个sample和上一个sample的token |
| | | sample_data | 某时刻（timestamp）的传感器数据，例如图片，点云或者Radar等等。包含ego_pose_token（指向某一ego pose）、calibrated_sensor_token（指向特定的calibrated sensor block）、数据存储路径、时间戳、is_key_frame值等 |
| | | ego_pose | 采集所用的车辆在特定时间戳的pose。基于激光雷达地图的定位算法给出（见nuScenes论文）。包含坐标系原点和四元坐标（需要读论文了解具体含义）、时间戳 |
| | Annotation | instance | 对象注释实例，包含对象类别、注释数量、初次和末次注释的token |
| | | lidarseg（目录中未见） | 标注信息和点云之间的映射关系。filename为对应的bin file的名称，该bin文件以numpy arrays的形式存储了nuScenes-lidarseg labels 标签。sample_data_token为某个sample data的token key，该sample data为对应的bin标签文件所标注的、且is_key_frame值为True的sample_data数据 |
| | | sample_annotation | 样本标注。包含所对应的sample token、instance token、attribute token、visibility token、bounding box 中心点坐标以及长宽高、lidar points数量、下一个和上一个标注 |
| | Taxonomy | category | 对象分类集合，包含human、vehicle等。子类以human.pedestrain.adult的形式表示 |
| | | attribute | 一些关于样本的可能会改变的性质，Example: a vehicle being parked/stopped/moving, and whether or not a bicycle has a rider |
| | | visibility | 可见性评估，0-40%，40%-60%，60%-80%以及80%-100% |

## 二、数据集标注格式

大致框架与nuScenes相同，数据采集依然是scene的形式，一个scene的视频时长为20s，然后对每一个scene的数据做timestamp处理。整体文件结构如下如所示（maps的文件夹是否需要加进去）**需要**:




文件夹结构




数据存储文件结构

其中数据文件命名规则为：S008表示第8个场景scene中的数据；中间的LIDAR_TOP表示传感器的类型；后面的1533151603537560表示文件的名字，可以用截取视频的当前时间表示。

标注文件annotations，采用coco的存储的json文件存储的方式。

**json文件表示方法：**

```
{
   'images'
   'annotations'
   'categories'
   'videos'
   'attributes'
}
```

**#### json说明**

Images：保存了图像相关的信息

Annotations：包含了当前场景中的关于目标和车辆本身的信息

Categories：类别属性声明

Videos：场景属性声明

Attributes：目标状态声明

Pointclouds：激光雷达的数据

```
{"images": [{"id": 1, # 图像的id
```

"file_name":"samples/CAM_FRONT/S008_CAM_FRONT__1533151603512404.jpg",#图像的路径

"calib": [[1252.8131103515625, 0.0, 826.588134765625, 0.0], [0.0, 1252.8131103515625, 469.9846496582031, 0.0], [0.0, 0.0, 1.0, 0.0]],# 当前传感器的内参

"video_id": 1, #所属场景

"frame_id": 1, #当前相机所拍摄的图像属性id

"sensor_id": 1,#传感器的token

"sample_token": "3e8750f331d7499e9b5123e9eb70f2e2", # sample的token

"trans_matrix": [[-0.471156096007008,-0.47107062410861184, .....], [0.0, 0.0, 0.0, 1.0]], # 传感器相对于全局坐标变化矩阵

"velocity_trans_matrix": [[-0.017214572464080702,.....], [0.0, 0.0, 0.0, 1.0]],

# 传感器相对于全局坐标系旋转矩阵

"width": 1600, # 当前图像的宽

"height": 900, #图像的高

"pose_record_trans": [599.849775495386, 1647.6411294309523, 0.0], #车辆原始的tralation

"pose_record_rot": [-0.9687876119182126, -0.004506968075376869, -0.00792272203393983, 0.24772460658591755], #车辆原始四元素数据

"cs_record_trans": [1.72200568478, 0.00475453292289, 1.49491291905], #传感器原始tranlation数据

"cs_record_rot": [0.5077241387638071, -0.4973392230703816, 0.49837167536166627, -0.4964832014373754],#传感器原始rotation数据

"radar_pc": [[4.8049437558445955,

.....

.....3.0]],# 当前相机传感器对应的雷达点云数据

"camera_intrinsic": [[1252.8131021185304, 0.0, 826.588114781398], [0.0, 1252.8131021185304, 469.9846626224581], [0.0, 0.0, 1.0]]},#当前相机内参

. ....

],

"annotations": [{"id": 1, #有效 box 的 id

"image_id": 1,#当前box所属图像id

"category_id": 6,#box的所属类别

"dim": [1.778, 0.621, 0.647], #box的长宽高

"location": [-7.516707974170363, 2.3902318792386197, 36.525215135341675], #box的中心点

"depth": 36.525215135341675, #box的深度

"occluded": 0,

"truncated": 0,

"rotation_y": 1.794254246073804, #绕相机坐标系y轴旋转角（观测角）

"amodel_center": [568.765380859375, 521.4768676757812], #box在图像中的中心点位置

"iscrowd": 0, #道路拥挤程度或可以表示当前box的目标的可见度

"track_id": 1,#被检测目标物体box的id，用于跟踪

"attributes": 3, #当前box的目标的状态

"velocity": [-0.8272804720015601, 0.6514334151510506, 0.1158991965606086], #目标在全局坐标系的速度

"velocity_cam": [-0.18670421959392292, -0.10339086971218908, -1.0376140584730447, 0.0],# 目标在相机坐标系中的速度

"bbox": [553.4947253760256, 490.5584879921071, 30.329964533936845, 62.24330538453637],#box数据

"area": 1887.8372447879883, #box覆盖面积

"alpha": 1.9972966284425855}, .....],

"categories": [{"name": "car", "id": 1},

{"name": "truck", "id": 2},

{"name": "bus", "id": 3},

{"name": "trailer", "id": 4},

{"name": "construction_vehicle", "id": 5},

{"name": "pedestrian", "id": 6},

{"name": "motorcycle", "id": 7},

{"name": "bicycle", "id": 8},

{"name": "traffic_cone", "id": 9},

{"name": "barrier", "id": 10}],

"videos": [{"id": 1, "file_name": "scene-0103"},

{"id": 2, "file_name": "scene-0916"}],

"attributes": {"": 0,

"cycle.with_rider": 1,

"cycle.without_rider": 2,

"pedestrian.moving": 3,

"pedestrian.standing": 4,

"pedestrian.sitting_lying_down": 5,

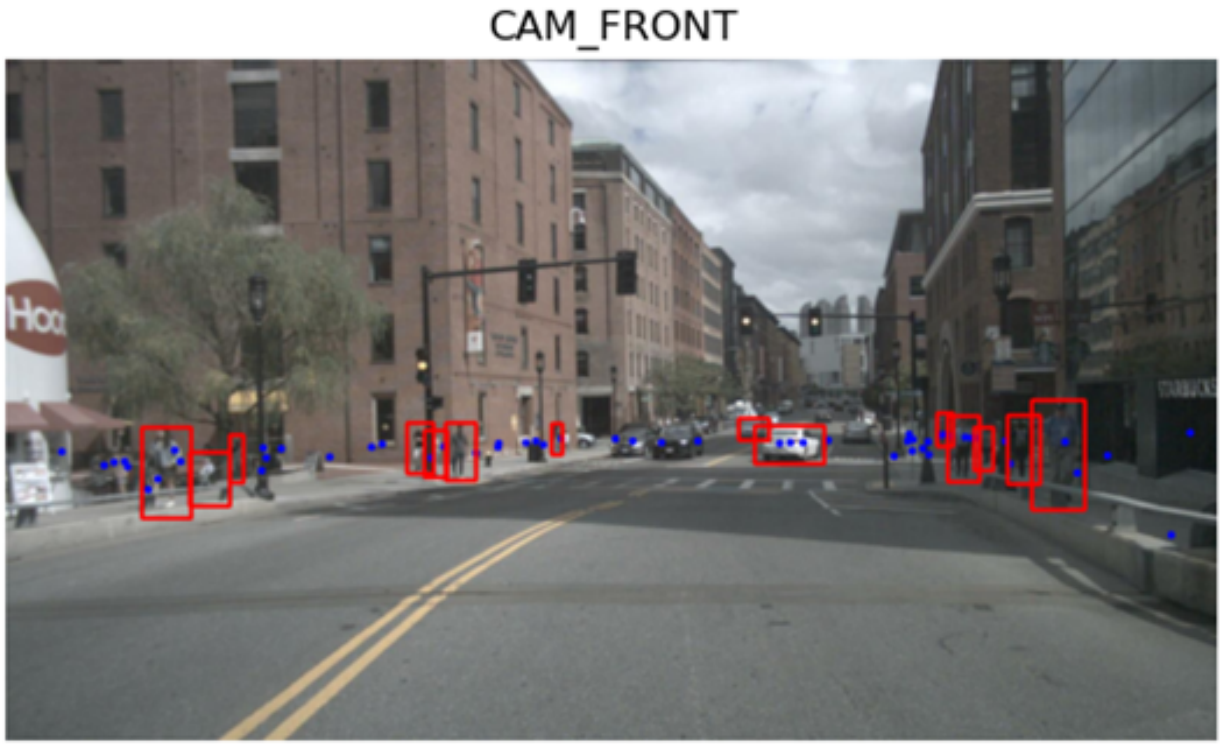
"vehicle.moving": 6,

"vehicle.parked": 7,

"vehicle.stopped": 8},

"pointclouds": [.....]}}

**转换成coco格式可视化：**




点云数据可视化

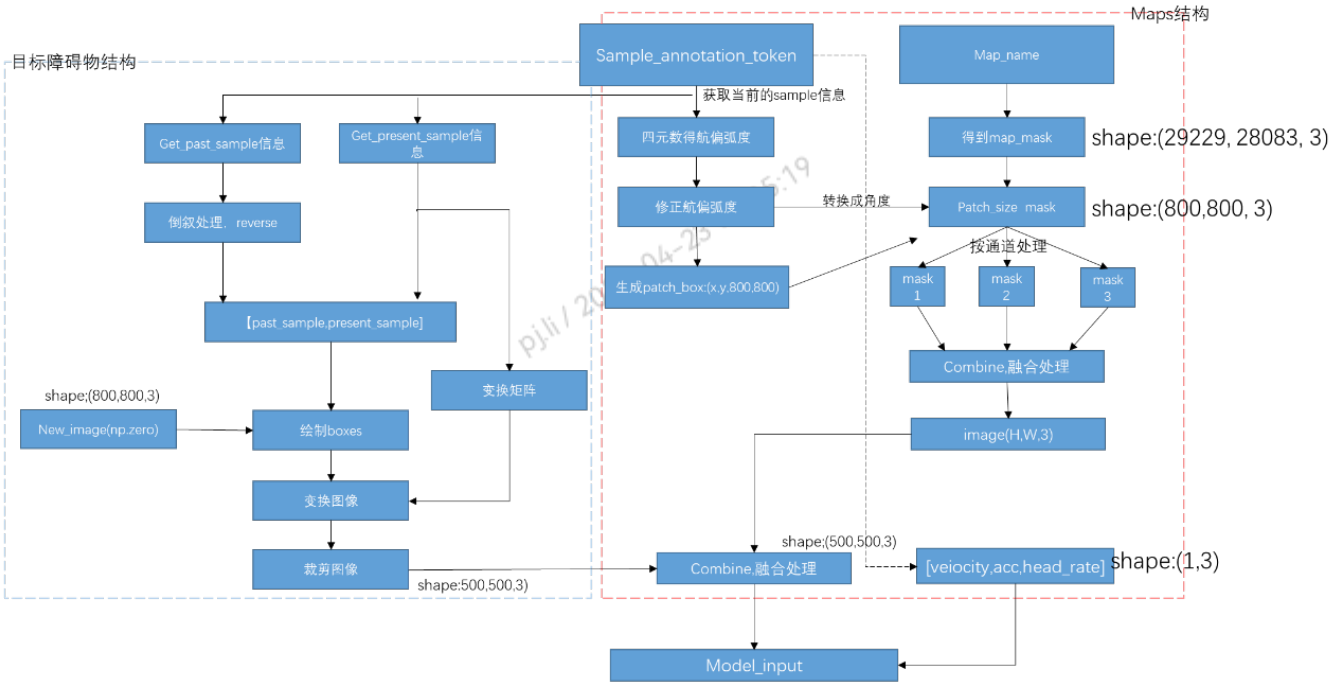

COCO的Json格式box3D可视化


原nuScenes数据格式box3D可视化

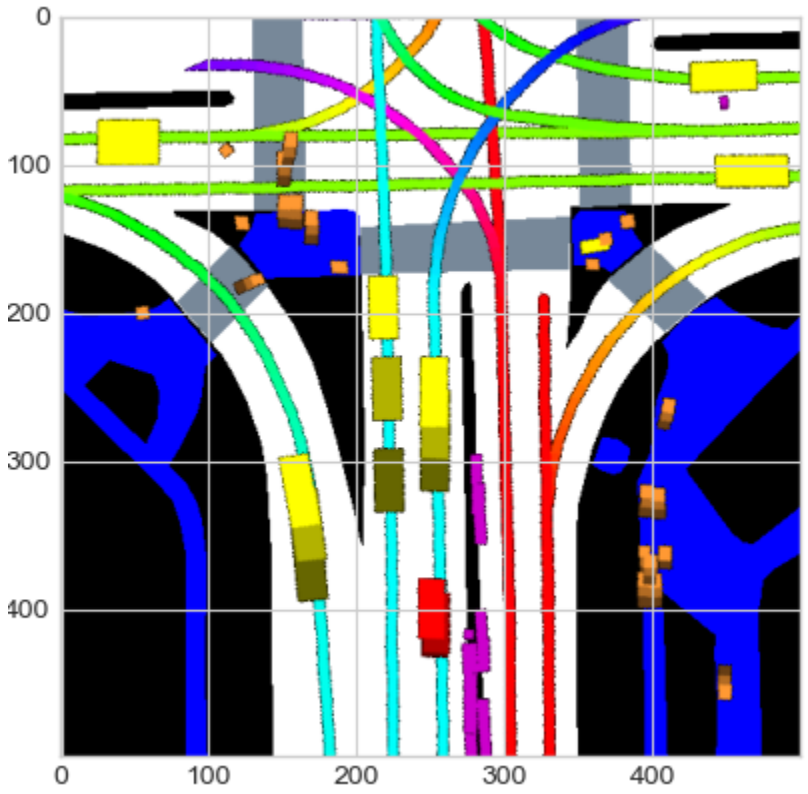
## 三、数据集使用

参考链接：https://zhuanlan.zhihu.com/p/508912923

地图数据相关：https://candyguo.github.io/blog-post-16/




预处理结构图



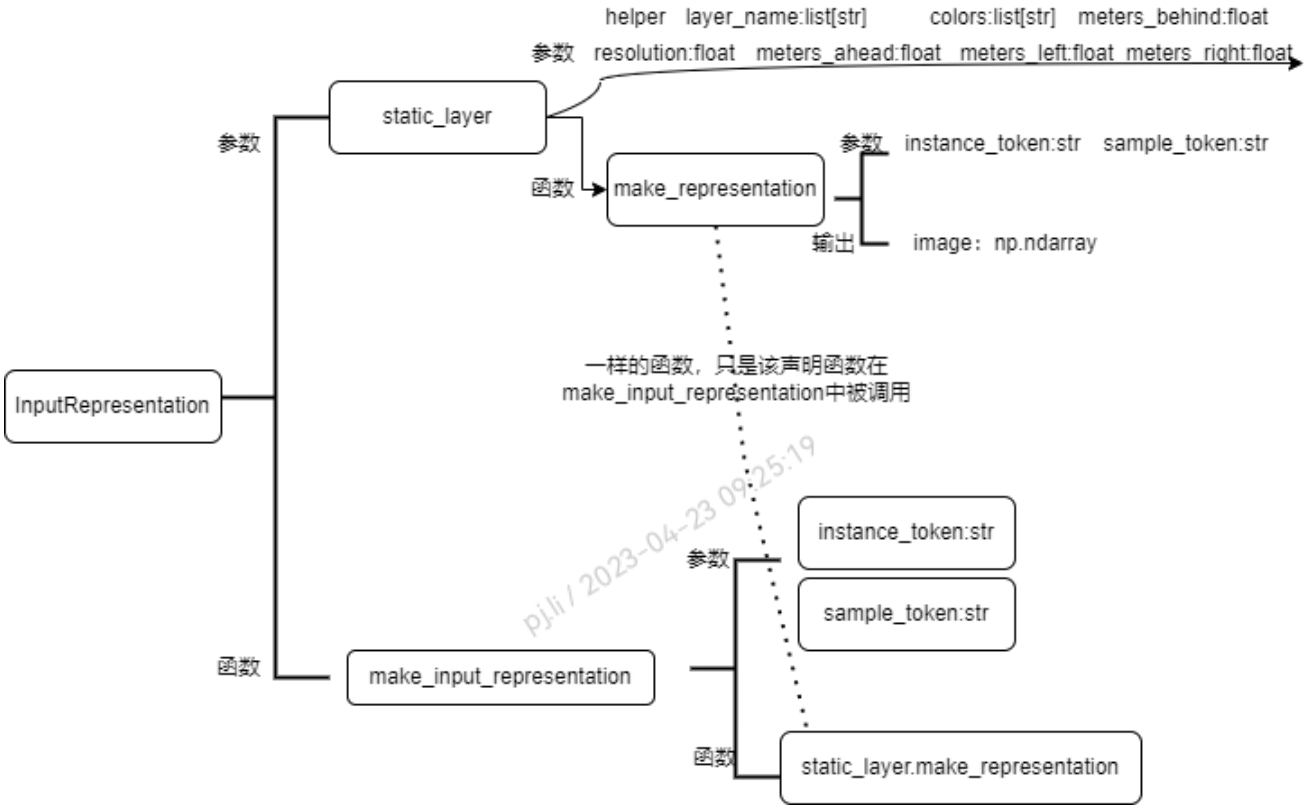

图像数据预处理可视化

```
======
image_shape: torch.Size([1, 3, 500, 500])
veiocity,acc,head_rate: tensor([[1.9737, 3.0657, 0.0000]])
```

最后输入模型的数据形式

## 四、nuScenes地图接口和pgp算法地图接口

nuScenes的地图接口



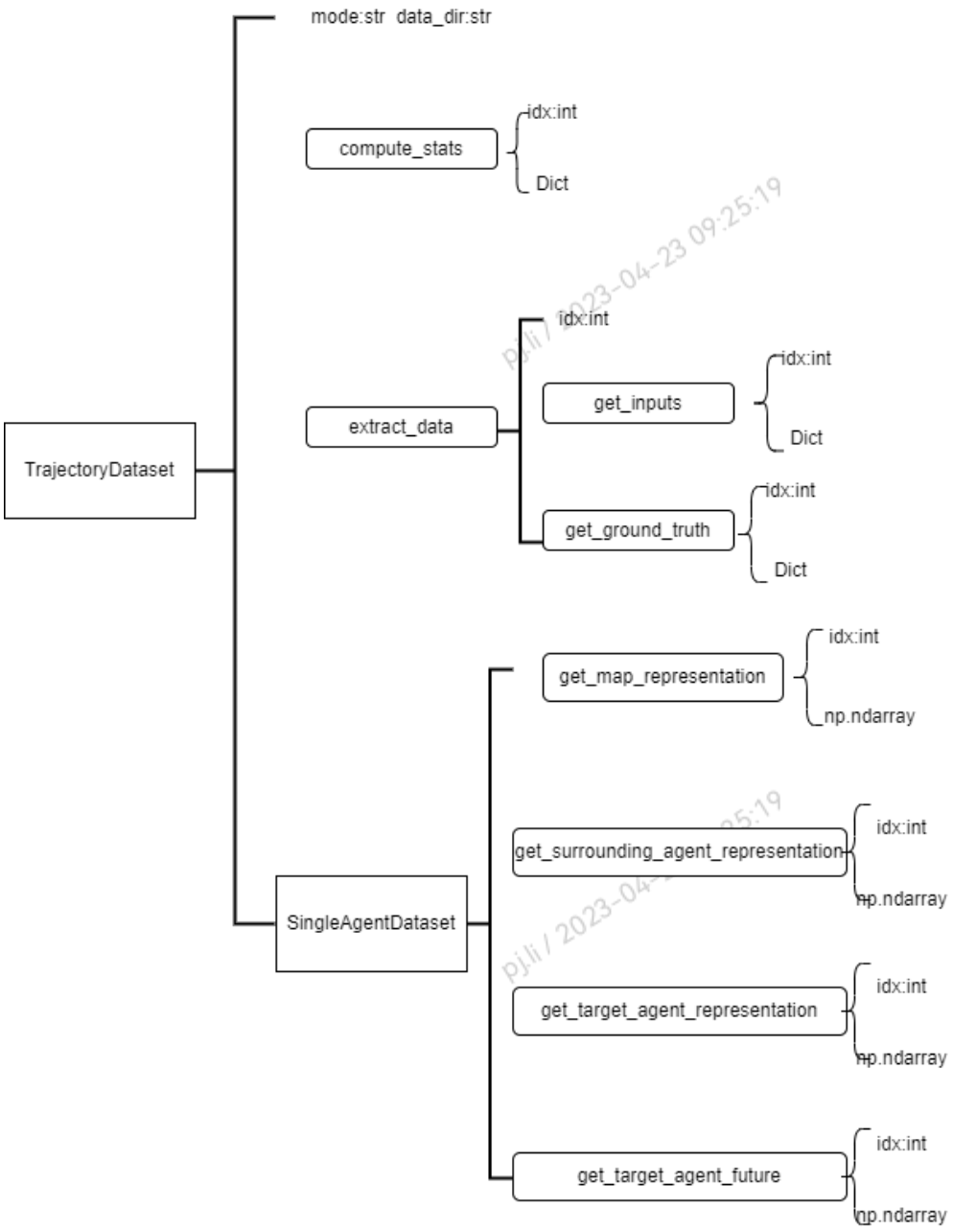

地图接口定义

InputRepresentation

| 输入 | | | |
|---|---|---|---|
| static_layer | helper | | nuScenes内置函数 |
| | layer_names | List[str] | |
| | colors | List[str] | |
| | meters_behind | float | |
| | meters_left | float | |
| | meters_right | float | |
| | resolution | float | |
| | meters_ahead | float | |
| | make_representation | | |
| make_input_representation | | | |
| | instance_token | str | 实例的token |
| | | | |

| | | | |
|---|---|---|---|
| | sample_token | str | sample的token |
| | static_layer.make_representation | | |
| make_presentation | | | |
| | instance__token | str | |
| | samplt_token | str | |
| 输出: | mtp | np.ndarray | 最后得到的预处理map |

算法最终的地图接口




# 预处理整体接口

**inputs**

1. instance_token:str
2. sample_token
3. map_representation:dict
4. surrounding_agent_representation:dict
5. target_agent_representation:tensor
6. agent_node_masks:dict
7. init_node:tensor
8. node_seq_gt:tensor

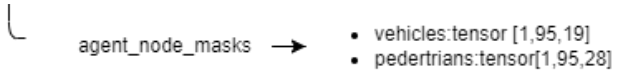
→

map_representation →

- lane_node_feats:tensor [1,95,20,6]
- lane_node_masks:tensor [1,95,20,6]
- s_next:tensor [1,95,10]
- edge_type:tensor [1,95,10]

surrounding_agent_representation →

- vehicles:tensor [1,19,5,5]
- vehivle_masks:tensor [1,19,5,5]
- pedestrians:tensor [1,28,5,5]
- predestrian_masks:tensor [1,28,5,5]




**label**

1. traj[batchsize,tf*2,2]
2. evf_gt

TrajectoryDataset

这里的idx是表示token_list的索引值，然后处理成instance和sample的token。

| 输入 | | |
|---|---|---|
| TrajectoryDataset | mode | str |
| | data_dir | str |
| | compute_stats | |
| | extract_data | |
| | singAgentDataset | |
| | | |
| extract _stats | idx | int |
| | get_inputs | |
| | get_ground_truth | |
| get_inputs | idx | int |
| get_ground_truth | idx | int |
| _ | | |
| compute_stats | idx | int |
| | | |
| singleAgentDataset | | |
| | get_map_represebtation | |
| | get_surrounding_agent_representation | |
| | get_target_agent_representation | |
| | get_target_agent_future | |
| get_map_represebtation | idx | int |
| get_surrounding_agent_representation | idx | int |
| get_target_agent_representation | idx | int |
| get_target_agent_future | idx | int |
| 输出: | | |
| | inputs | Dict |
| | ground_truth | Dict |
| Inputs | | |
| | map_representation | Dict |
| | node_seg_gt | np.ndarray |

| | | |
|---|---|---|
| map_representation | | |
| | lane_node_feats | np.ndarray |
| | lane_node_masks | np.ndarray |
| | s_next | np.ndarray |
| | edge_type | np.ndarray |
| ground_truth | | |
| | traj | np.ndarray |
| | evf_gt | np.ndarray |

## 五、针对nuScenes地图中lane、ped_crossing、stop_line和stop_line的表达形式

其中图像中绿色字体表示：在预处理阶段没有用到的信息

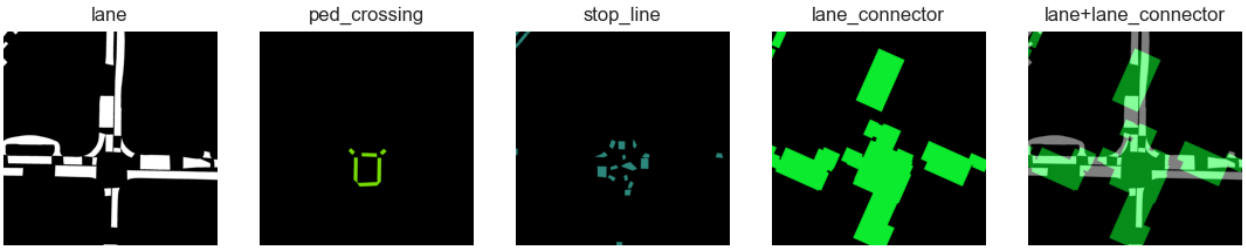
# 针对nuScenes数据地图的map_name.json文件说明

**这里只介绍和lane、ped_crossing和stop_line有关的token信息**

convas_edge：[宽，高]　　地图的图像大小，但是json中的表示方法宽和高都缩小了10倍

- polygon:多边形
  - token：表示当前的信息token
  - exterior_node_tokens:表示区域,用node_token
  - holes：表示其中有node_token。
- node:点
  - token
  - x：坐标点x
  - y：坐标点y
- road_segment:道路分割区域
  - token
  - polygon_token:多边形token
  - is_intersection：是否为路口，True/False
  - drivable_area_token:表示可驾驶区域
- *road_block:路障*
  - toekn
  - polygon_token
  - from_edge_line_token
  - to_edge_line_token
- lane:车道
  - token
  - polygon_token
  - *lane_type:"CAR"表示车道的类型*
  - *from_edge_line_token*
  - *to_edge_line_token*
  - *left_lane_divider_segments:*
    - *node_token*
    - *segment_type:"DOUBLE_DASHED_WHITE"分割的类型*
  - *right_lane_divider_segments:与上面相同*
- ped_crossing:人行道
  - token
  - polygon_token
  - *road_segment_tokenegment_token*
- walkway：人行横道
  - token
  - polygon


## 对应的可视化效果，截取map中一段进行可视化




## 六，对nuscenes和pgp算法障碍物接口

nuScenes障碍物接口

InputRepresentation

| 输入 | | | |
|---|---|---|---|
| agent | helper | | nuScenes内置函数 |
| | seconds_of_history | float | |
| | frequency_in_hz | float | default:2 |
| | meters_behind | float | |
| | meters_left | float | |
| | meters_right | float | |

| | | | |
|---|---|---|---|
| | resolution | float | |
| | meters_ahead | float | |
| | color_mapping | tuple[int,int,int] | |
| | make_representation | | |
| make_input_representation | | | |
| | instance_token | str | 实例的token |
| | sample_token | str | sample的token |
| | agent.make_representation | | |
| make_presentation | | | |
| | instance__token | str | |
| | samplt_token | str | |
| 输出: | img | np.ndarray | 最后得到的预处理map |

算法最终的障碍物接口

TrajectoryDataset

这里的idx是表示token_list的索引值，然后处理成instance和sample的token。

| 输入 | | |
|---|---|---|
| TrajectoryDataset | mode | str |
| | data_dir | str |
| | compute_stats | |
| | extract_data | |
| | singAgentDataset | |
| | | |
| extract _stats | idx | int |
| | get_inputs | |
| | get_ground_truth | |
| get_inputs | idx | int |
| get_ground_truth | idx | int |
| _ | | |
| compute_stats | idx | int |
| | | |
| singleAgentDataset | | |
| | get_map_representation | |
| | get_surrounding_agent_representation | |
| | get_target_agent_representation | |
| | get_target_agent_future | |
| get_map_represebtation | idx | int |
| get_surrounding_agent_representation | idx | int |
| get_target_agent_representation | idx | int |

| | | |
|---|---|---|
| get_target_agent_future | idx | int |
| 输出: | | |
| | inputs | dict |
| | ground_truth | dict |
| Inputs | | |
| | surrounding_agent_representation | |
| | agent_node_masks | |
| | target_agent_representation | np.ndarray |
| | init_node | np.ndarray |
| surrounding_agent_representation | | |
| | vehicles | np.ndarray |
| | vehicles_masks | np.ndarray |
| | predestrians | np.ndarray |
| | predestrian_masks | np.ndarray |
| agent_node_masks | | |
| | vehicles | |
| | predestrians | |
| ground_truth | | |
| | traj | np.ndarray |
| | evf_gt | np.ndarray |