YN&$d2CKD\*&X

20240420：从MultitaskLoader读取数据，读到的数据信息键值(类型)对分别为：

meta\_info <class 'dict'> 记录了大量内外参校准信息，信息如下：

T\_vcs2cam <class 'list'>

intrinsics <class 'list'>

distort\_coeffs <class 'list'>

transformats <class 'list'>

ipm\_img\_sizes <class 'list'>

img\_shape <class 'list'>

homo\_transforms <class 'dict'>

cam2local\_rot <class 'list'>

cam2local\_translation <class 'list'>

T\_local2vcs <class 'list'>

persp\_view\_scale <class 'list'>

T\_lidar2vcs <class 'torch.Tensor'>

calib\_path <class 'list'>

fake\_homo\_flag <class 'torch.Tensor'>

aug\_flag <class 'list'>

homography <class 'torch.Tensor'>

offset\_save\_file <class 'list'>

homo\_offset <class 'torch.Tensor'>

temporal\_info <class 'dict'>

timestamp <class 'torch.Tensor'>

pack\_dir <class 'list'> 列表内存了一个路径

img\_paths <class 'list'>列表内存了七个图片路径

have\_lidar\_input <class 'bool'>true or false

object\_token <class 'list'>列表内存了一个路径

pillar\_data <class 'list'>[tensor], tensor shape 为 num\_pillar x 48 x 4(xyzi), i为0~255 测试的代码中pillar\_data长度为5 ，这是代表地平线一帧数据有五个雷达嘛？

pillar\_coordinates <class 'list'>[tensor], tensor shape 为 num\_pillar x 4， 其中前两列全部为0，应该分别代表bs和Z的坐标索引，第三第四列应该分别为XY的pillar坐标索引。

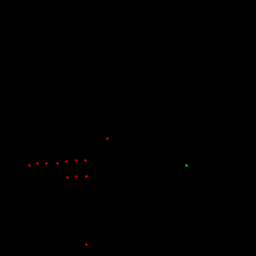
pillar\_num\_points <class 'list'>[tensor], tensor shape 为 num\_pillar, 记录了每一个pillar内有多少个点云。

num\_pillars <class 'list'>[tensor] list长度为5，tensor shape 为 1，记录了每帧点云有多少个pillar

pillar\_shape <class 'list'>[tensor] list长度为1，tensor等于[608, 368, 1](长度像素，宽度像素，高度像素)

annos\_dict <class 'dict'>记录了目标检测的标注{'gt\_boxes': 属性顺序大概率是(x y z w l h yaw), 'gt\_names': np.array([str]), 'gt\_classes': np.array(1 1 1 2 2 3)} gt\_names到gt\_classes存在映射关系。

hm <class 'torch.Tensor'> shape为1 x 3 x 256 x 256 以0，1 rgb的方式记录了真值目标的热力图，从图像上看：



红点为gt\_classes分类为1的类别，绿点为gt\_classes为2的类别，蓝点为gt\_classes为3的类别。从hm的分布结合目标框属性可以推断，该图像上方为x正向，左方为y正向。图像的原点应该在中间偏下侧。即整个图像x正轴长度比负轴长度大。

anno\_box <class 'torch.Tensor'> shape为1 x 500 x 8, 似乎已经把需要回归的值计算出来了？

8个属性分别猜测的计算规则如下：[XY应该是相对于像素角点的偏移，Z正常回归，log(w l h), sin(head), cos(head)]

ind <class 'torch.Tensor'> 1 x 500(计算的应该是256 x 256这个平面的), 计算了gt对应的像素序号，像素序号的排列规则应该是h x 256 + w。结合坐标轴，可以知道xy越大的目标序号越小。

mask <class 'torch.Tensor'> 1 x 500, 0,1 个数等于gt框数量

cat <class 'torch.Tensor'> 1 想500，0，1，2， 3.。。。 gt\_classes从0开始排列。

gt\_boxes\_tasks <class 'torch.Tensor'> 1 x 500 x 7，与annos\_dict中的gt\_boxes信息没区别。

anno\_box\_reg <class 'torch.Tensor'> 1 x 12500 x 8, 没有看懂其含义

ind\_reg <class 'torch.Tensor'> 1 x 12500, 没有看懂其含义

mask\_reg <class 'torch.Tensor'>1 x 12500, 没有看懂其含义

homography\_temporal\_lidar <class 'torch.Tensor'> 1 x 5 x 3 x 3，大部分接近于单位阵

homography\_temporal <class 'torch.Tensor'> 1 x 4 x 3 x 3大部分接近于单位阵

view <class 'dict'> {'front': [1], 'side': [5], 'round': [0], 'narrow': [1]}

img <class 'list'> [5, 3, 512, 960] 五张前视图像

side\_img <class 'list'>[25, 3, 640, 960]

narrow\_img <class 'list'>[5, 3, 512, 960]

return\_latest\_flag <class 'list'>[true] 列表内长度仅一个值

temporal\_clr\_flag <class 'list'>[false] 列表内长度仅一个值

———————————————————————————————————————

MultitaskLoader读取数据时，先返回MultitaskLoaderIterator对象，再调用了其\_\_next\_\_方法。

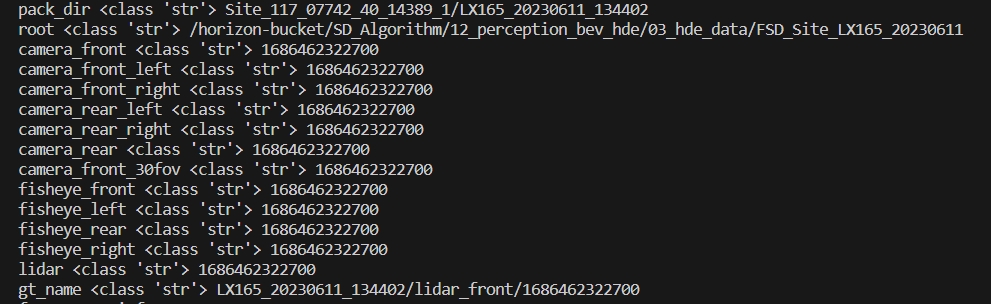
建国代码的训练配置内，stop\_by step, num\_epochs为none， num\_steps为50000。

0426陈磊提供的auto3DV类型数据集记录：

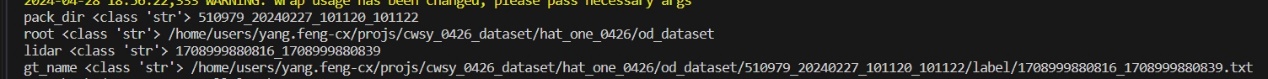
1. 每一帧包含frames、temporal\_info两个key。其中temporal\_info又是一个字典，打印结果为{'num\_frames\_per\_iter': 1}。
2. frames为[<hat\_internal.data.datasets.auto\_3dv.Frame object at 0x7f767d599940>]，是一个长度为1的列表。

20240428--------------------------------：

地平线数据的Frame对象 传入的frame\_sync\_info字典的键值对信息：



陈磊测试的Auto3DV对象 传入的frame\_sync\_info字典的键值对信息：



其实可以看到，这个和陈磊存储的txt所有键值对一样，这说明sync file存了什么，就会读取出来什么

类别 Auto 3DV / Frame / collect 3DV学习记录：

Frame对象中，只要传入camera\_view\_names， img\_load\_size这两个列表，当在sync info中找不到需要读取的图像对象时，或创造用0填充的图像列表返回。从Auto3DV数据集的使用上来看，per\_view\_shape参数对于单激光来说不重要。