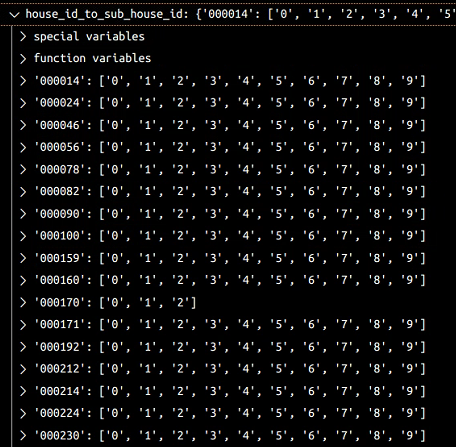
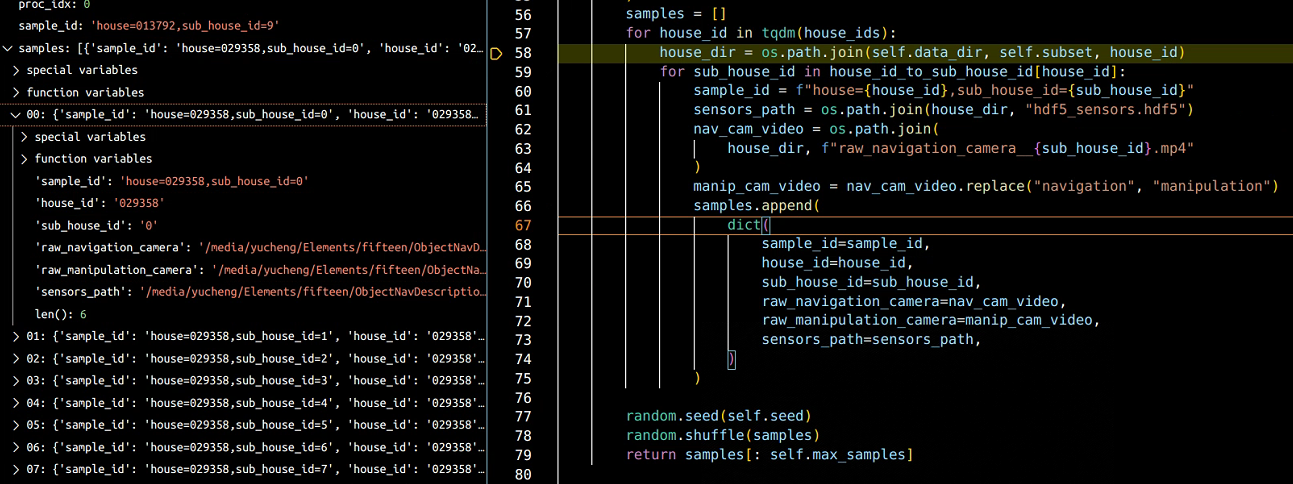
*\*这份文档记录spoc数据集的解析和制作*

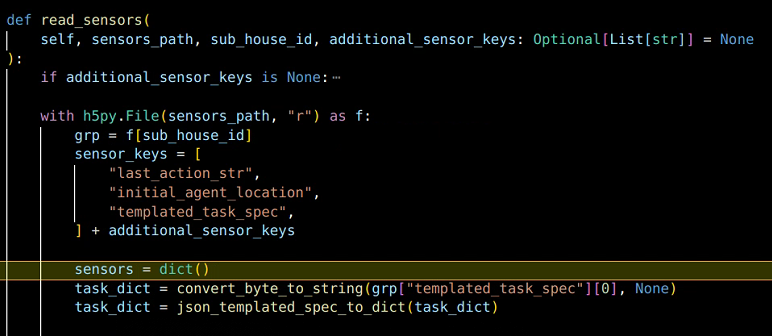
**ChoresMultitaskDataset(Dataset)中成员:self.datasets是一个由一系列ChoresDataset(Dataset)构成的list。ChoresDataset的初始化时要用ChoresDataReader的partial\_load\_sample函数来读取house\_id\_to\_sub\_ house\_id:**

****

**整合到一个叫sample的list中，其中每个item包含，house\_id, sub\_house\_id, 相应的视频文件位置和传感器文件位置**

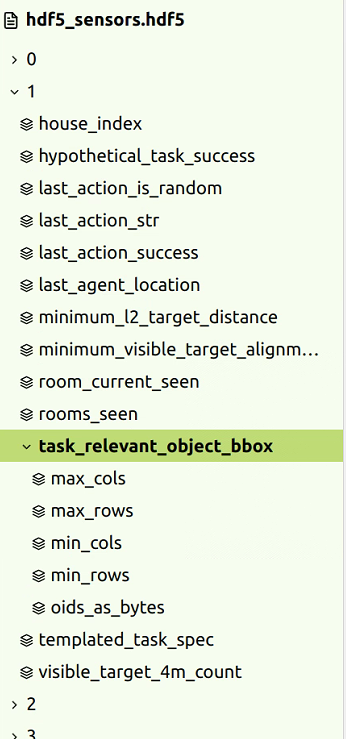


***ChoresDataset的\_\_getitem\_\_()***

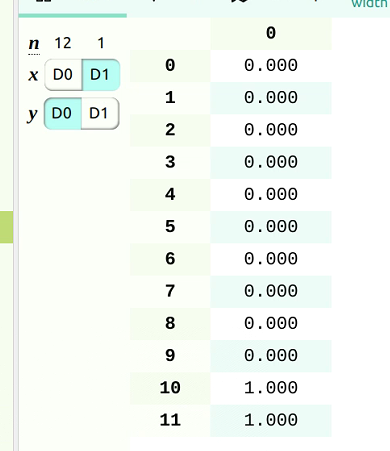


**首先要用reader里的read\_sensor函数读hdf5文件，f[sub\_house\_id]：**

**hdf5文件结构：**

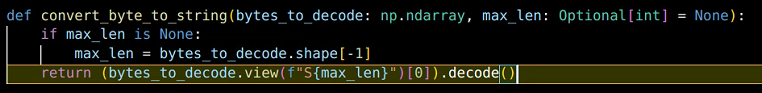


* **house\_index: 房子的索引，每个step都一样的 shape:[step]，这是仿真引擎中的step**
* **hypothetical\_task\_success: 在当前step指示任务是否已完成，比如下面这个跑了12个step的sample，在倒数两个step的时候任务完成了**



* **last\_action\_is\_random: shape:[step]**
* **last\_action\_str: shape: [step, 200]**

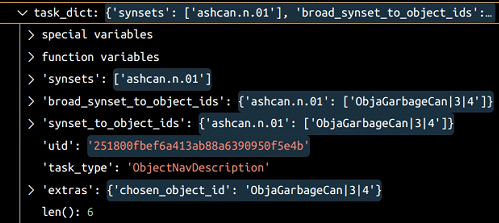
**这里存是200维的uint32但这里用.view(“S200”).decode转回了字符串，除了最后一个step对应的是end三个字符，其他step对应的都只有第一个数字不是0对应一个字符，所以last\_action读进来是[step]的字符，然后根据window的大小取[arg.sliding\_window]大小的action，last\_action用[:-1]从last\_action\_str取，然后action真值对last\_action用[1:]取即可**



* **last\_action\_success: shape:[step]**
* **last\_agent\_location(float64): shape:[step, 6]**



* **minimum\_l2\_target\_distance: [step]**
* **minimum\_visible\_target\_alignment: [step]，如果目标物体可以通过转过一定角度移到视野中心，则是需要转过的角度；否则是-1**
* **templated\_task\_spec：也是用.view(“S{max\_length}”).decode转成字符串，记录了一些当前任务的信息，被用来生成任务文本**

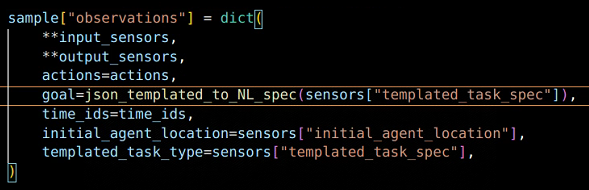


**在处理actions对应的time\_ids时，把time\_ids随机地后移了一段，文中说作用是：  
we train with a limited temporal context window (e.g. 100), but the model uses all past observations during inference. To enable using a larger context window during deploy ment**

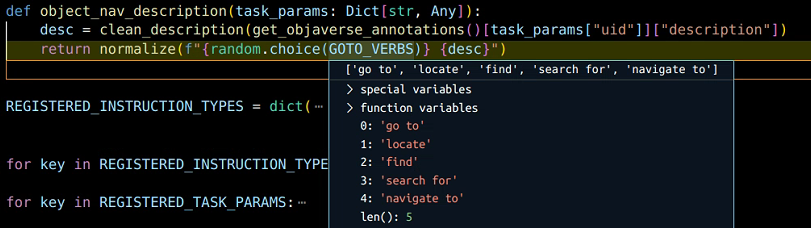
**读取视频是用torchvision.io中的read\_video函数读的，视频读出来是[frames, C, H, W]的tensor。**



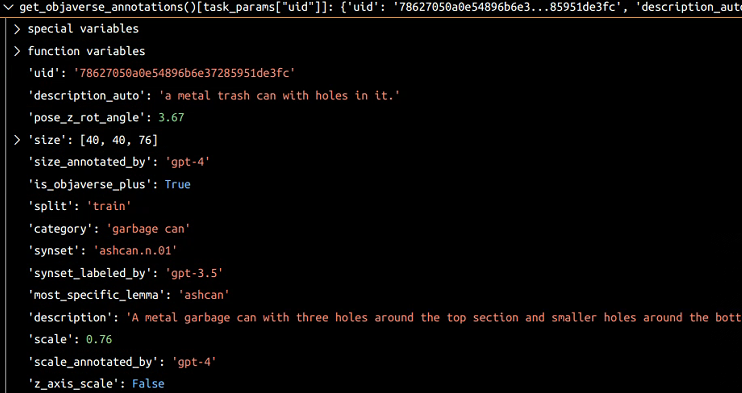
**最后信息全存进observation里**

****

**这里的json\_templated\_to\_NL\_spec把刚才的任务信息转成相应任务文本**

****

**get\_objaverse\_anntations()会从objaverse\_assets物品集的annotation,json,gz中读出里面写好的全部物品，然后根据uid指定该物品，并取其description属性，加上go to之类的前缀返回，这是objnavdescription任务的goal**

****

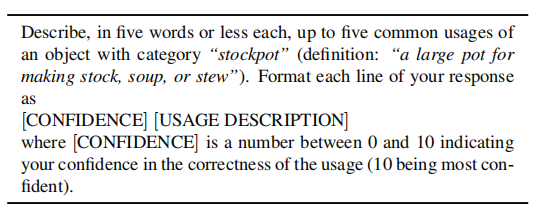
***关于数据集的生成与标注***

**场景（scene）用的是AI2THOR，在该场景中还加入了Objaverse的assets以增加多样性(2k->40k)，这些Objaverse的assets是有各自的metadata的（relevance to household environments，object types grounded in the WordNet 2022 hierarchy, instance descriptions, and size in meters），把Objaverse的assets和AI2THOR原有的assets配对，保留了四万一千多个unique assets。**

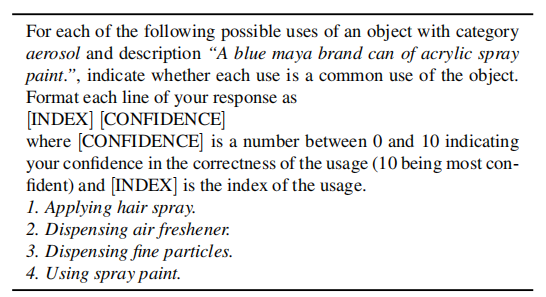
**为了进一步增加多样性，使用ProcTHOR来生成200k个不同的house，每个house有1-8个房间。ProcTHOR生成的房屋具有特定的布局，而且有更多得到assets，这些assets还有更详细的anno**

**生成的assets叫ObjaverseHome，加上原有的THOR，共被分为863个WordNet synsets，也就是863个类，这些类都有相应的hypernyms，所以当描述的物体比较generic时也可以通过hypernym来retrive到相应的种类，作者给这些asset都添加了额外标注（scale, standard pose, descriptions, category, and nearest WordNet synset）。**

**接着用GPT3.5给这些个WordNet synset生成相应的affordance，863个synset每个synset用GPT生成5个unique的affordance，共4315个affordance，GPT的query如下：**



**接着，再用GPT给这4315个生成的affordance来进行confidence评分，GPT的query如下：**



**最后，用GPT-3’s ada给前面的affordance生成embedding，利用这些embedding生成cosine similarity>0.96的更多affordance cluster，一共生成了343个这种cluster且平均每个cluster包含2.78个affordance。所以最后应该是一共有4315+343\*2.78个affordance。**

***Goal的分类***

1. **ObjectType goal：给定指令去到一个能最精确解释该synset的lemma处，可能有多个**
2. **ObjectRoom goal：类①，但是限定该goal必须在任务开始时的房间里，也表明这种东西在别的房间也有**
3. **RelativeAtrribute goal：共六种比较，最高最低、最大最小、最远最近（相对于房间中某一个unique的anchor）,也表明这个东西在某个房间里不止一个**
4. **Affordance goal：这个好理解，不过一般提供稍微generic一点的hypernym，然后agent要找到既满足hypernym又满足某affordance的目标**
5. **LocalReference goal：同一房间存在多于一个的某种物体时，通过“near”和“on top of”来指定哪一个**
6. **Description goal：给定目标物体的open-vocabulary description，这个description是前面用GPT通过相似词嵌入生成的那些额外的affordance，只针对ObjaverseHome里的assets生成了。**
7. **Room goal：目标是去到某个房间类型，一个house里可以有多个**

***Sample的方法***

**首先，为了使任务的target是valid的，设置了一个filter，navigation任务的filter会确保物体的高度不大于1.1m，物体的bbox对角线小于10cm，中间维最小4cm；在manipulation任务里拿的物体bbox最大维小于50cm**

1. **RoomVisit：从House里随机采出一个room再在room里随机采一个loc**
2. **ObjNav：这里为了平衡不同synsets的分布，有一个计数器来记录synset被采次数，然后每次采样根据synset的采样次数的倒数赋予分布权重，另外还要过nav的filter**
3. **Fetch: 同②，但要过manip的filter**
4. **Pickup: 类③，但agent生在一个能操作目标物且带有正确manip camera朝向的位置**
5. **ObjNavRoom: 类②，但要求目标synset在house中有多个instance，且所在的room of a certain type需要是同一个room。同时在其他type的room里也需要有至少一个instance**
6. **ObjNavRelAttr: 先定义那六个相对属性，最大最小最高最低最远最近，然后过在count每个target type同时还count每个属性的采样次数，物品的分布类⑤但不要求别的类型房间也有，从采样次数最低的属性开始试**
7. **ObjNavAfford: target synset设为scene里基本synset的hypernym，然后按照ObjNav的方法采样**
8. **ObjNavLocalRef: 类似⑥，定义near和on，且避免以地板墙面等为参考**
9. **ObjNavDesc: 类似ObjNav但限定assets源于ObjaverseHome因为只有它们有NL的description，且要求一个scene里只有一个目标**
10. **RoomNav：计数不同类型房间的采样次数**

***Planners***

**所有的planner都是由两个subroutine组合成的，一个naviagtion subroutine，用于近似跟踪最短轨迹，在stuck时能recompute；另一个Pick-up subroutine，分为旋转、伸出、张开爪子登七个步骤。这个线程还包含一个重试的选择，比如在20%的random choice下在东西没拿起来后再做一次尝试以增加鲁棒性**

**所以在maniskill里要做出这么一个数据集需要什么？**

1. **把ObjaverseHome里的assets加载进ManiSkill**
2. **把assets根据house的json文件里的layout给加载到ManiSkill形成scene**
3. **写一个子线程planner来对ManiSkill里的agent进行最短路径控制**
4. **Record下来相机和前面提到的一些过程性信息**

**但如果说要形成一个新的数据集：**

1. **assets还是使用Objaverse的情况下，必须在ManiSkill环境下搭一个ProcTHOR，以生成大量的新屋子**
2. **assets还是使用Objaverse的情况下，用GPT来给各种asset生成各种Description和affordance等额外的metadata**
3. **根据生成的scene\_config文件load各种house**
4. **Planner进行最短路径控制**
5. **Record**