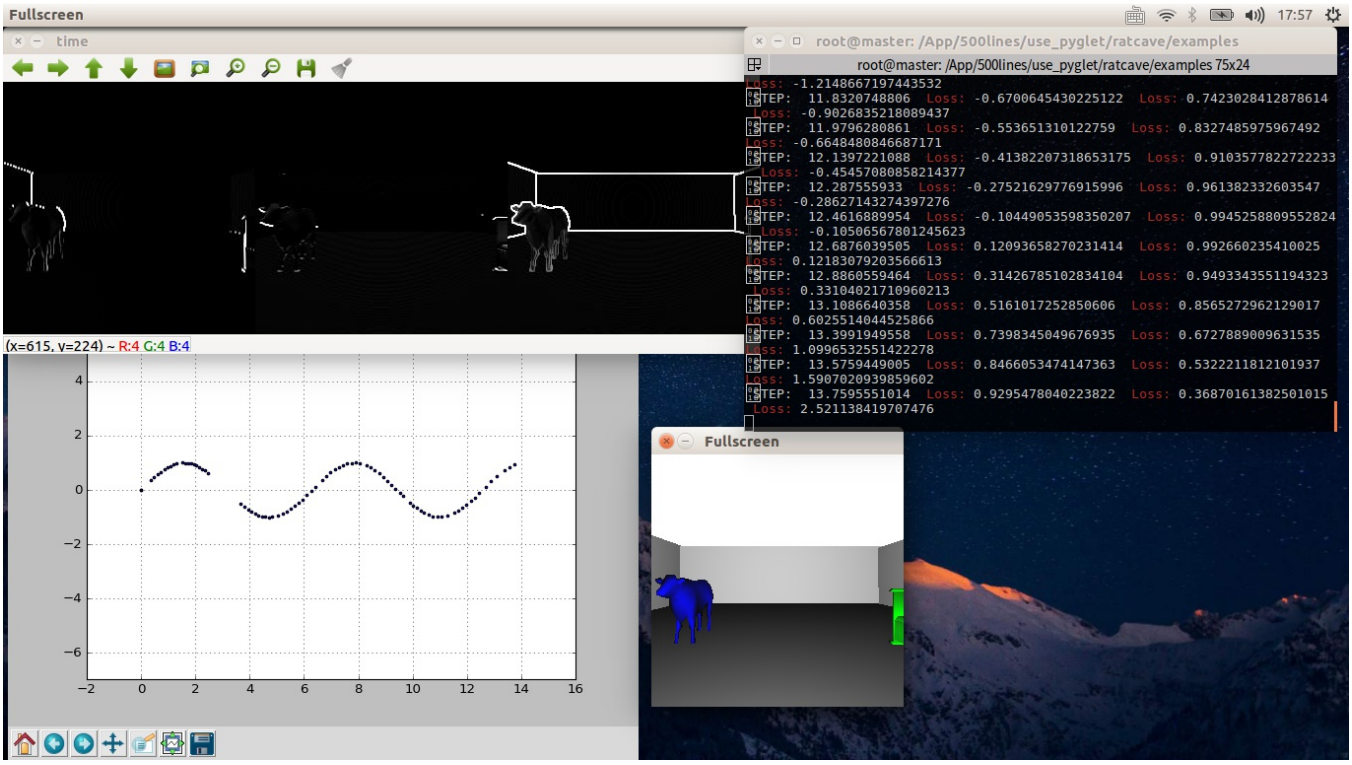
3D虚拟交互式强化学习环境构建工具Fuzzy World



> “ 这个工具不是为了开发而开发的，相反，这个工具是因为"自然语言和视觉共同作用来实现复杂推理"这个想法而设计的。

Fuzzy World的特点

这个工具取名为Fuzzy World的原因:其构建、其逻辑、其训练方式的自由度很大。

Fuzzy World是一个**基于3D虚拟环境的交互学习工具**,Agent可以通过视觉接口和语言交互接口探索环境的隐藏规则(通常是复杂的)并构建实体的联系。

真正应该关心的是:它和之前的**基于3D虚拟环境的Agent学习库**有什么不一样?实际上已经有足够多的关于这类任务的工具:

- **Gazebo:**机器人运动仿真环境，他的物理逻辑非常完美，很适合用来仿真制造业的机器人。总之感觉比较偏机械硬件。
- **RobSchool:**基于 OpenAI Gym 环境的 Roboschool，是一个教机器人如何协调运动(跑步，躲避，跳跃)的虚拟环境，基于强化学习。

- **XWorld:**最接近我们目前做的这个工具，但是XWorld主要还是和语言结合用来做Navigation的。看下文会发现，XWorld推崇无监督或弱监督学习，而我们的理念属于强监督强化学习。

综合来看，**“可交互”、“视觉与自然语言结合”、“训练方式不限于强化学习”**这几点全满足的库目前还没有，所以Fuzzy World这个小工具就是为这个任务设计的。

Fuzzy World的世界逻辑

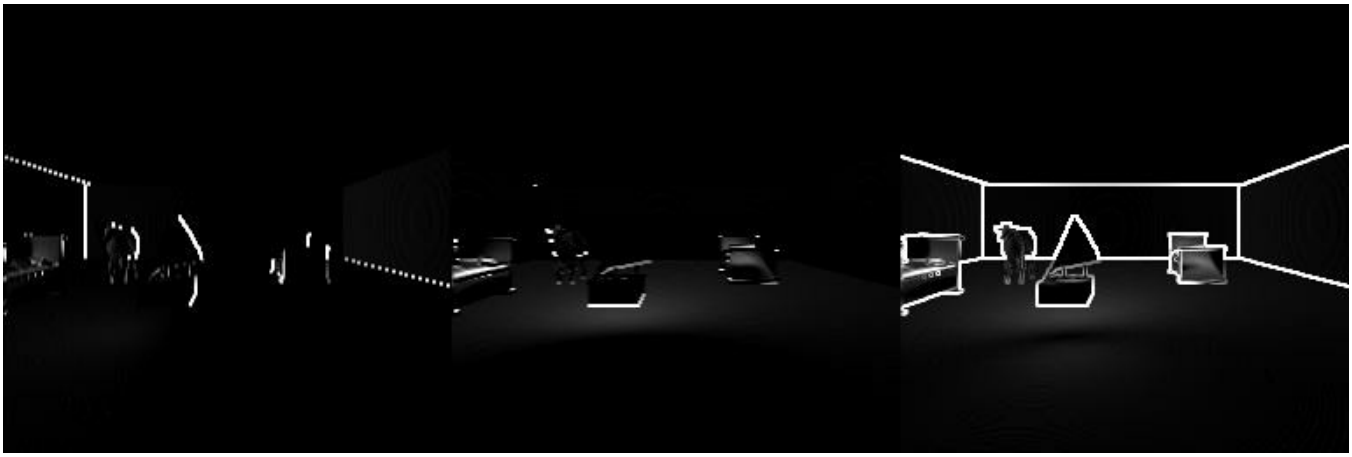
在Fuzzy World中,Agent通过第一人称视角观察世界和交互操作后世界的变化，自然语言被用于引导和辅助Agent来认知世界完成Task;所以这个世界至少包含以下内容：

- **object** O:环境里的物体实体，可以是自定义导入的.obj格式的模型。
- **state** S^t:物体在时刻t的状态S^t：物体是否运动，其温度或者位置等等。
- **rule set** $\{R_i\}$:规则集合，比如“开关2打开则物体5的温度升高”这样的一阶逻辑，一个场景中的规则集合$\{R_i\}$是给定的有限的，可以描述为物体及状态的转移映射即:

$$R_i : S^t(O_m) \rightarrow S^{t+1}(O_n)$$

以上是环境自身客观存在的性质，Agent可以通过**语言提示**和**主动尝试**探索物体性质和这些规则，来完成任务。

Agent自身包含的信息如下：



- **visual** V^t: Agent在时刻t的机器视觉信息。
- **sentence** L^t: Agent在时刻t的接受的提示语句。
- **strategy** π_θ: Agent的决策函数。
- **action** L^t: Agent在时刻t采取的行动：移动，改变环境中的某个物体的某个属性，不作为等等。
- **value** v^t: Agent在时刻t评估当前决策的价值。
- **graph** G: Agent维护的一个语义网络，包含了世界物体的逻辑规则等高级语义内容。

还记得强化学习方法Q-learning里那个Value Function吗：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \lambda \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

Q-learning维护了一个Q矩阵，而graph G的作用就包含了Q矩阵的功能(用于决策，但不仅限于此)。

自动环境生成

一个关键的疑虑就是:**难道使用这个工具需要手工去编写大量的场景及其物体分布、逻辑规则?**不需要,因为工具已经内置了自动地图生成功能,可以生成海量的交互环境供Agent训练.

具体来说我规定了一种名为".fw"的文件来文本化地描述一个**FuzzyWorld的场景物体及世界逻辑**，一个".fw"的文件格式如下：

```
WORLD_NAME:sam1;
WORLD_SIZE:44;
OBJ_NUM:5;
OBJ:ID:"0";MODEL:"./obj/box/jeep.obj";MESH:"jeep";POS:-1,-0.5;TEMP:"TEMP_NO
OBJ:ID:"1";MODEL:"./obj/box/music.obj";MESH:"music";POS:0,0;TEMP:"TEMP_NORM
OBJ:ID:"2";MODEL:"./obj/box/bed.obj";MESH:"bed";POS:-0.5,0;TEMP:"TEMP_NORMA
OBJ:ID:"3";MODEL:"./obj/box/cow.obj";MESH:"cow";POS:-0.5,-1;TEMP:"TEMP_NORM
OBJ:ID:"4";MODEL:"./obj/box/bed.obj";MESH:"bed";POS:0.5,-0.5;TEMP:"TEMP_NOR
RULE_NUM:4;
RULE:"3":"STOP"=>"4":"TEMP_HIGH";
RULE:"2":"MOVE"=>"2":"STOP";
RULE:"0":"TEMP_NORMAL"=>"3":"TEMP_HIGH";
RULE:"0":"MOVE"=>"2":"STOP";
```

所以通过这种文本格式，程序可以由文本描述来创建一个场景，亦可以由程序快速生成多个文本描述供创建大量的场景和逻辑规则来自动地构建海量训练集。

如何构建数据集训练自己的模型?

训练可以分为在线训练和离线训练两种，在线训练就是一遍跑3D场景一遍训练模型，考虑到这种方式太烧显卡(至少我的笔记本扛不动...)，所以重点说说离线训练(当然工具支持将其拓展为在线训练，但是没必要一边看Agent行动一边训练)。

离线训练就是先跑一遍**正确的任务完成示范**，收集完成步骤的每一帧图像，每一步策略描述和对应的场景信息(实体、位置、状态、规则)，存储这些数据，然后再去训练模型，在程序里我写了一份程序演示如何收集数据(程序见https://github.com/Luomin1993/fuzzy-world/blob/master/make_train_data.py)，数据包括:

- **rule set** $\{R_i\}$;
- **visual screenshot** V^t;
- **sentence set** L^t;
- **correct action steps** $\{a_t\}$;

总结

总体来说就是，我们开发了一个小工具，它可以实现：

- Agent在3D虚拟世界利用**语言**和**视觉**完成特定任务。
- Agent可以利用环境的逻辑和环境交互。
- 可以先收集数据，然后离线地用不同的方法训练模型。

可以用来研究什么问题：

- 基于自然语言命令的逻辑推理：特别是长程的，比如命令是**使汽车的温度降低**，而当前环境蕴含逻辑：**打开(开关A)** → **会转动(椅子B)** → **会升温(电阻)** → **会降温(汽车)**这样的长程逻辑。
- Agent之间的交互：通过和环境交互训练，Agent可以维护一个**语义graph** G，既可以转化为自然语言传达给人类，也可以传递给另一个Agent;
- 视觉对语义理解的重要性：通过对比有视觉的语义模型和无视觉的纯语义模型的差异得出视觉对语义理解的重要性。