Unitree_rl部署方案的安装与学习记录 副本

Unitree_rl是基于unitree_guide改进的四足机器人强化学习部署方案。unitree_guide是一个ROS开源项目，用于控制 Unitree的四足机器人，而unitree_rl允许用户使用rl训练模型作为机器人的控制器，可将通过isaac gym训练出的模型用于gazebo仿真和实机部署，用于验证强化学习方法的结果。

github页面：

https://github.com/dstx123/unitree_rl/blob/master/README_CN.md

实机演示视频：

https://www.bilibili.com/video/BV1Gt421M7NV/?spm_id_from=333.999.0.0&vd_source=5f2437f40dc85b8d866187b401f714d2

安装和报错解决方法

Unitree_rl是使用C/C++编写的，所以需要提前下载C++版本的libtorch，安装CUDA11.3和CUDNN，并且要在 `unitree_guide/unitree_guide/CMakeLists.txt` 手动设置libtorch的路径：

```
1 set(Torch_DIR /yourpath/libtorch/share/cmake/Torch)
```

之后在src所在的目录运行

```
1 catkin_make
```

跟着github页面指示操作即可。不出意外的话马上要出现意外了。

这时进行catkin_make大概率会报错：**error: redefinition of ‘struct at::_ops::lu_solve’**

这是由于unitree_rl编写的QP求解函数和libtorch同样define了一个叫lu_solve的函数，所以编译器就会报错提示出现了重定义。我在该项目的issue中提到了这个问题：error: redefinition of ‘struct at::_ops::lu_solve’ · Issue #1 · dstx123/unitree_rl

作者给出的解决方法是修改libtorch里的lu_solve，我尝试了一下，发现要改的地方太多，这一改牵一发而动全身。相比之下直接在unitree_rl下的Array.hh里修改比较方便，把这个头文件的lu_solve全部替换成另一个词就好了。我查了一下整个项目的文件，lu_solve出现的地方大概就5处，改了之后

能成功编译也能顺利运行，但对于QP求解有没有影响呢，这个我还没做对比实验，所以也不确定这么做是否合理，有精力的小伙伴可以试试只修改libtorch里的lu_solve。

仿真

编译完先source一下，然后启动launch文件

```
source devel/setup.bash
roslaunch unitree_guide gazeboSim.launch
```

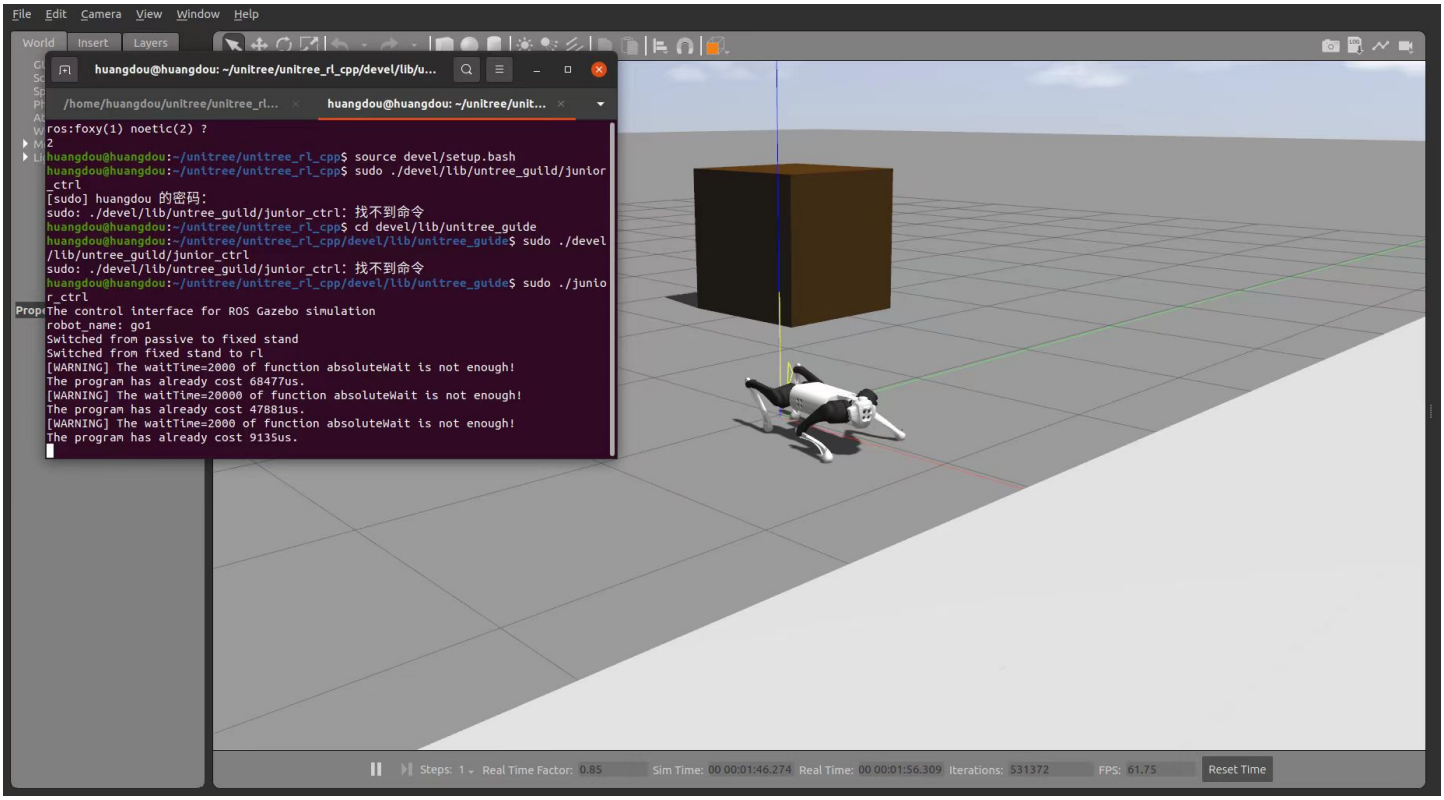
编译完成后会在devel/lib/unitree_guide/下生成一个控制器，另开一个终端source并执行控制程序

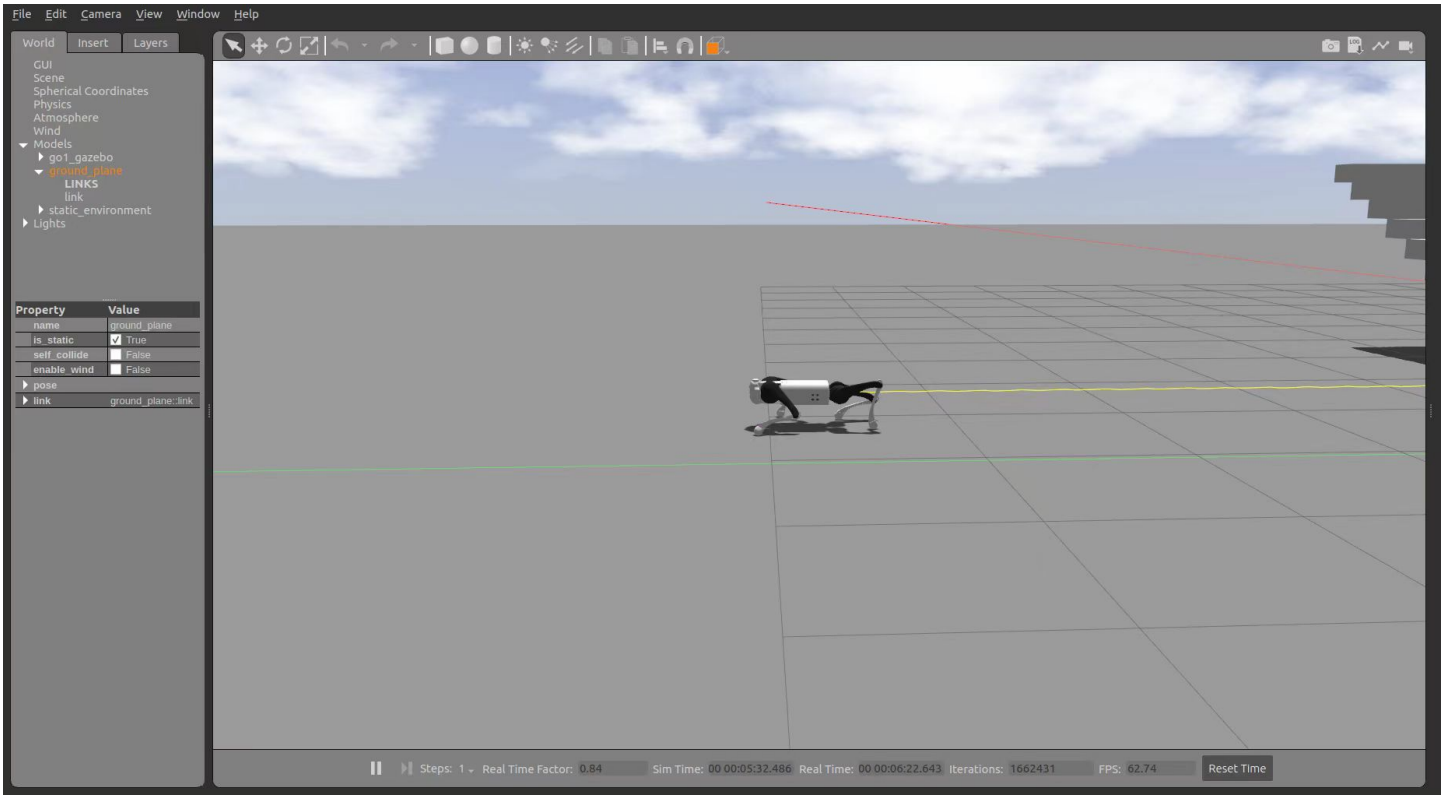
```
source devel/setup.bash
./devel/lib/unitree_guide/junior_ctrl
```

按下 `2` 键，机器人将站起来。然后按 `4` 键，机器人进入强化学习状态机，待机器人站稳后通过键盘控制运动。

在终端中可以通过'w' 's' 'a' 'd'键控制机器人x和y轴速度，通过'j' 'l'键控制机器人旋转。按‘空格’使机器人站定

仿真页面长这样：





这个机器人是用作者已经训练好的模型控制的，走路很奇怪，是用一种半趴的形式走的，和unitree_guide里的站起来走路不太一样，不过unitree_rl的演示视频倒也是这么走的……

仿真里机器人上不了前面的阶梯，不知道是阶梯太高、还是作者给的模型不好、亦或者是QP求解被修改导致的。

使用自定义模型仿真

unitree_rl允许用户加载自己在isaac gym中训练得到的模型进行仿真，将pth模型文件转换为jit文件，放在model文件夹下，并在文件 `unitree_guide/unitree_guide/CMakeLists.txt` 中修改模型路径：

```
set(BODY_MODEL_PATH "${CMAKE_CURRENT_SOURCE_DIR}/model/body.jit")
set(ADAPT_MODEL_PATH "${CMAKE_CURRENT_SOURCE_DIR}/model/adapt.jit")
add_definitions(-DBODY_MODEL_PATH="${BODY_MODEL_PATH}")
add_definitions(-DADAPT_MODEL_PATH="${ADAPT_MODEL_PATH}")
```

这里具有迷惑性，好像要准备两个model，但是我突然想起到通过isaac gym训练得到的模型不是只有一个pth文件吗？那这两个model分别代表什么呢？于是我问了作者如何得到adapt model和body model：

How to get Adapt_model and Body_model? · Issue #3 · dstx123/unitree_rl

作者解释实际上这里并不是一定要放body和adapt两个模型文件。

rl模型在 `void State_RL::_loadPolicy()` 和 `void State_RL::_action_compute()` 两个函数内被引用：

```
void State_RL::_loadPolicy()  // 加载JIT模型
{
    this->_body_module = torch::jit::load(BODY_MODEL_PATH);
    this->_adapt_module = torch::jit::load(ADAPT_MODEL_PATH);
}
void State_RL::_action_compute(){
    torch::Tensor latent = _adapt_module.forward({this->_obs_buffer_tensor}).toTensor();
    this->_action = _body_module.forward({torch::cat({_observation,latent},-1)}).toTensor();
    this->_action = torch::clamp(this->_action, -clip_actions, clip_actions);
     .......
     }
```

可以看到这段代码的作用是将observation和latent作为输入，经过body model后得到action（也就是机器人关节角度），latent可以通过adapt model得到。但是adapt model是什么、怎么得到？latent张量又代表了什么含义？body model又怎么得到？作者并没有说明，只是说这两个模型是个例子，只要RL模型的最终输出是机器人关节角度就行了。

个人猜测这两个模型是作者未发表文章的内容，所以暂时不用关注了。