

1、Clone与环境配置

第零步：ssh连接并设置学术加速

[EB cloud 学术加速页面](#)

```
1 # 开启加速服务
2 source /public/bin/network_accelerate
3
4 # 关闭加速服务
5 source /public/bin/network_accelerate_stop
```

第一步：创建与激活环境

这里使用了 `uv`（一个极速 Python 包管理器）来创建环境。

```
1 # 1. 创建名为 env_isaacLab 的虚拟环境，指定 Python 版本为 3.11
2 # --seed 参数用于初始化 pip 等基础包
3 uv venv --python 3.11 --seed env_isaacLab
4
5 # 2. 激活刚刚创建的虚拟环境
6 source env_isaacLab/bin/activate
```

第二步：安装核心依赖 (PyTorch & Isaac Sim)

安装特定版本的 PyTorch 和 NVIDIA Isaac Sim 的 Python 客户端。

```
1 # 3. 安装指定版本的 PyTorch 和 Torchvision
2 # --index-url 指定从 PyTorch 官方源下载 CUDA 12.8 对应的版本
3 pip install -U torch==2.7.0 torchvision==0.22.0 --index-url
  https://download.pytorch.org/whl/cu128
4
5 # 4. 从 NVIDIA 的 PyPI 源安装 Isaac Sim 核心库
6 # 指定版本为 5.1.0，并包含所有组件和缓存扩展
7 pip install "isaacsim[all,extscache]==5.1.0" --extra-index-url
  https://pypi.nvidia.com
```

第三步：克隆仓库并安装 Isaac Lab

下载源码并执行项目自带的安装脚本。

```
1 # 5. 从 GitHub 克隆 Isaac Lab 的源代码仓库
```



```
2  git clone https://github.com/isaac-sim/IsaacLab.git
3
4  # 6. 进入克隆下来的目录
5  cd IsaacLab
6
7  # 7. 运行 Isaac Lab 的安装脚本
8  # --install (或 -i) 参数表示执行安装流程
9  ./isaacclab.sh --install
```

第四步：运行倒立摆测试

```
1  python scripts/reinforcement_learning/skrl/train.py /
2  --task=Isaac-Cartpole-v0
```

```
1  python scripts/reinforcement_learning/skrl/play.py /
2  --task=Isaac-Cartpole-v0 --headless
```

2、运行isaacclab

```
1  # 运行docker容器
2  ./docker/container.py start
3  ./docker/container.py enter
```

```
1  # 停止容器
2  python docker/container.py stop
```

3、创建新项目

```
1  ./isaacclab.sh --new
```

- 设置项目的保存路径
- 设置项目名称（会自动生成对应的 Python 包名）

选项	说明
External	独立于 Isaac Lab 仓库的项目，作为外部扩展运行。可以推送到自己的 GitHub 仓库，便于独立维护和更新

选项	说明
Internal	作为 Isaac Lab 仓库的一部分。仅用于向 Isaac Lab 贡献新任务

选项	说明
Direct	所有实现细节都在环境类中，抽象更少，开发路径最短，适合快速原型开发
Manager-Based	使用模块化定义（ActionManager、ObservationManager、RewardManager 等），更适合需要高度模块化和可扩展性的项目

框架	说明
RSL-RL	基于 PPO 算法，支持 GPU 并行训练
SKRL	支持多种算法（PPO, A2C, SAC, TD3 等），使用 PyTorch
Stable-Baselines3	流行的 RL 库，支持 PPO, A2C, SAC 等
RL-Games	支持分布式训练，适合大规模并行

```
[INFO] Running template generator...

? Task type: External
? Project path: /workspace/
? Project name: test

RL environment features support according to Isaac Lab workflows
```

Environment feature	Direct	Manager-based
Single-agent	yes	yes
Multi-agent	yes	no
Fundamental/composite spaces (apart from 'Box')	yes	no

```
? Isaac Lab workflow: Direct | multi-agent

Supported RL libraries
```

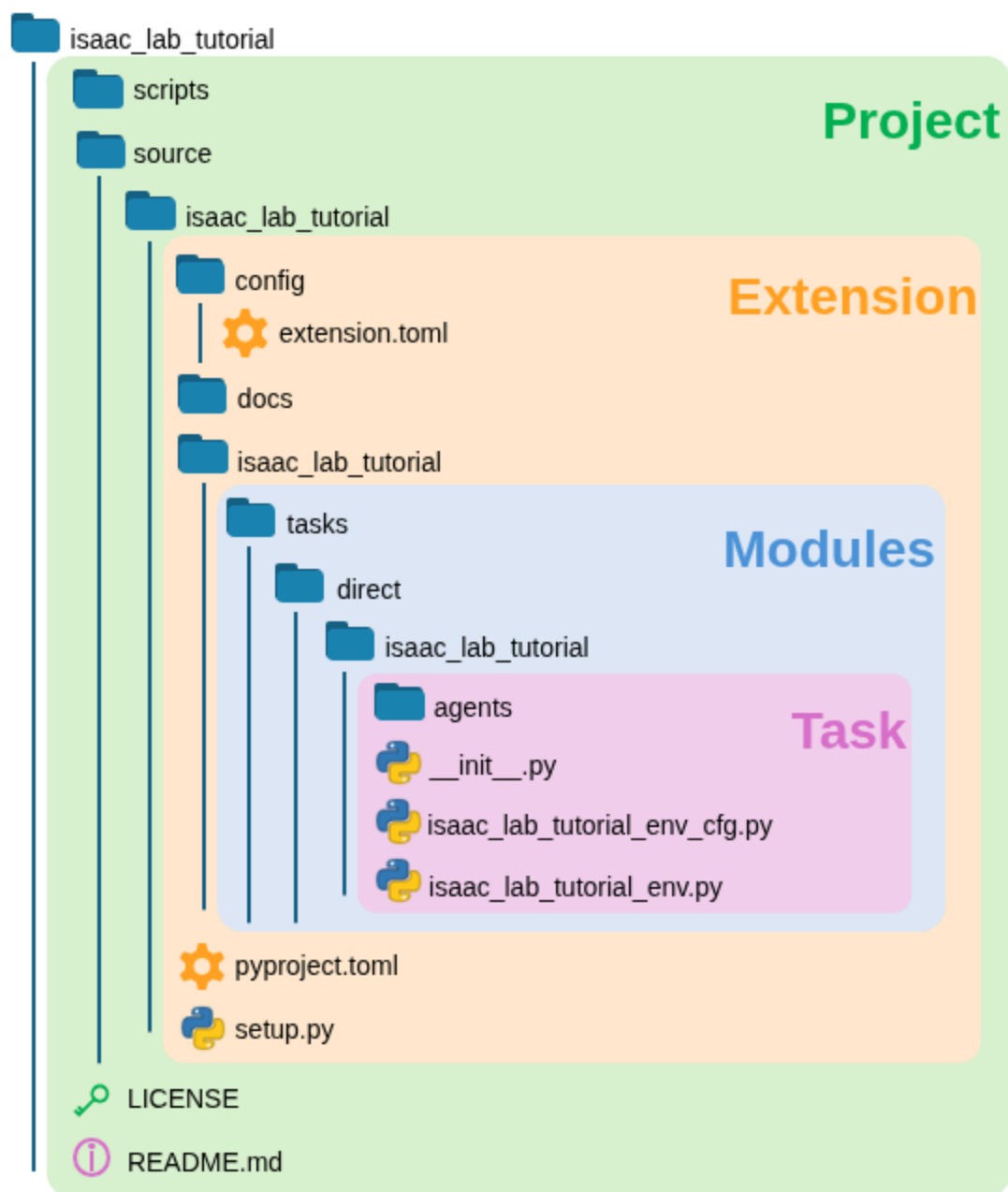
RL/training feature	rl_games	rsl_rl	skrl	sb3
ML frameworks	PyTorch	PyTorch	PyTorch, JAX	PyTorch
Relative performance	~1X	~1X	~1X	~0.03X
Algorithms	PPO	PPO	AMP, IPPPO, MAPPO, PPO	PPO
Multi-agent support	no	no	yes	no
Distributed training	yes	no	yes	no
Vectorized training	yes	yes	yes	no
Fundamental/composite spaces	no	no	yes	no

```
? RL library: 
> o skrl
-----
o all
```

创建后安装项目：

```
1 python -m pip install -e source/my_robot_task
```


新项目的文件结构：



init文件：

```
1 gym.register(  
2     id="Template-Test1-Marl-Direct-v0",  
3     entry_point=f"{__name__}.test1_marl_env:Test1MarlEnv",  
4     disable_env_checker=True,  
5     kwargs={
```



```

6         "env_cfg_entry_point": f"
    {__name__}.test1_marl_env_cfg:Test1MarlEnvCfg",
7         "skrl_mappo_cfg_entry_point": f"
    {agents.__name__}:skrl_mappo_cfg.yaml",
8     },
9 )

```

id = "XXXX" 的 XXXX为task ID

```

1 python scripts/reinforcement_learning/skrl/train.py /
2 --task=XXXX

```

```

1 python scripts/reinforcement_learning/skrl/play.py /
2 --task=XXXX --headless

```

CLI 重要参数解释：

训练参数 (train.py)

```

1 python scripts/reinforcement_learning/skrl/train.py \
2     --task=Isaac-XXX-v0          # 任务名称（必须）
3     --num_envs=64                # 并行环境数量
4     --headless                   # 无头模式（不渲染GUI）
5     --video                      # 录制视频
6     --enable_cameras             # 启用离屏渲染
7     --run_name my_experiment      # 运行名称（用于日志目录）
8     --agent rsl_rl_cfg_entry_point # 指定agent配置
9     --device cpu                 # 设备：cpu / gpu

```

播放参数 (play.py)

```

1 python scripts/reinforcement_learning/skrl/play.py \
2     --task=Isaac-XXX-v0          # 任务名称（必须）
3     --num_envs=32               # 环境数量
4     --headless                   # 无头模式
5     --use_last_checkpoint        # 使用最新checkpoint
6     --checkpoint /path/to/model.pt # 指定checkpoint路径
7     --device cpu                 # 设备

```

参数说明

参数	作用
--task	指定要训练/播放的环境ID
--num_envs	并行环境数，越多训练越快但显存占用越大
--headless	服务器训练必须，禁用GUI渲染
--video	录制训练过程视频保存到logs目录
--run_name	区分不同实验的运行
--agent	选择不同的agent配置
--use_last_checkpoint	播放时使用最新保存的模型
--device	指定用CPU还是GPU

XXXX_marl_env_cfg.py

```

1  # Copyright (c) 2022-2025, The Isaac Lab Project Developers
2  # ... (版权声明省略)
3
4  from isaacsim.assets.robots.cart_double_pendulum import
   CART_DOUBLE_PENDULUM_CFG
5  from isaacsim.assets import ArticulationCfg
6  from isaacsim.envs import DirectMARLEnvCfg
7  from isaacsim.scene import InteractiveSceneCfg
8  from isaacsim.sim import SimulationCfg
9  from isaacsim.utils import configclass
10
11 @configclass
12 class Test1MarlEnvCfg(DirectMARLEnvCfg):
13     """
14     多智能体倒立摆环境配置类
15     继承自 DirectMARLEnvCfg，表示使用直接工作流（无 Manager）的多智能体环境
16     """
17
18     # -----
19     # 1. 基础环境设置
20     # -----
21     decimation = 2          # 抽帧数。模拟器每跑 2 步物理计算，智能体才决策 1
   次。
22
23     episode_length_s = 5.0 # 假设物理频率 120Hz，控制频率就是 60Hz。
   每一回合(Episode)的最大时长（秒）。
24

```



```

25     # -----
26     # 2. 多智能体(MARL)定义
27     # -----
28     # 定义在这个环境中有哪几个智能体。
29     # 这里把"滑块(cart)"和"摆杆(pendulum)"拆成了两个独立的智能体。
30     possible_agents = ["cart", "pendulum"]
31
32     # 动作空间维度:
33     # cart: 1个维度 (左右推力)
34     # pendulum: 1个维度 (关节力矩)
35     action_spaces = {"cart": 1, "pendulum": 1}
36
37     # 观测空间维度:
38     # cart: 看到 4 个数 (自身位置, 自身速度, 杆角度, 杆角速度)
39     # pendulum: 看到 3 个数 (相对角度等, 具体看 env.py)
40     observation_spaces = {"cart": 4, "pendulum": 3}
41
42     # 全局状态空间维度。设置为 -1 通常表示不使用或者自动推断。
43     state_space = -1
44
45     # -----
46     # 3. 物理模拟参数
47     # -----
48     # dt=1/120: 物理引擎的步长, 即每秒计算 120 次物理碰撞和受力。
49     # render_interval: 渲染间隔, 与控制频率保持一致。
50     sim: SimulationCfg = SimulationCfg(dt=1 / 120,
render_interval=decimation)
51
52     # -----
53     # 4. 机器人资产 (Robot Asset)
54     # -----
55     # 加载预定义的 Cart-Double-Pendulum 资产配置
56     # replace(...) 用于修改该机器人在 USD 舞台中的路径, 支持正则表达式
57     # "/World/envs/env_.*Robot" 意味着每个环境里都有一个 Robot
58     robot_cfg: ArticulationCfg =
CART_DOUBLE_PENDULUM_CFG.replace(prim_path="/World/envs/env_.*Robot")
59
60     # -----
61     # 5. 场景设置
62     # -----
63     # num_envs=4096: 并行训练的环境数量 (GPU 强的话可以开很大)。
64     # env_spacing=4.0: 每个环境之间隔开 4 米, 防止机器人互相穿模干扰。
65     scene: InteractiveSceneCfg = InteractiveSceneCfg(num_envs=4096,
env_spacing=4.0, replicate_physics=True)
66
67     # -----

```



```

68     # 6. 自定义参数 (用于 env.py 中调用)
69     # -----
70     # --- 关节名称 (需要在 USD 文件里对应) ---
71     cart_dof_name = "slider_to_cart"          # 滑块关节
72     pole_dof_name = "cart_to_pole"           # 第一级杆关节
73     pendulum_dof_name = "pole_to_pendulum"   # 第二级杆关节
74
75     # --- 动作缩放 (Action Scaling) ---
76     # 神经网络输出通常是 [-1, 1], 需要乘上这个数变成真实的力(牛顿)或力矩(牛米)
77     cart_action_scale = 100.0 # [N]
78     pendulum_action_scale = 50.0 # [Nm]
79
80     # --- 奖励权重 (Reward Scales) ---
81     # 强化学习的核心: 定义什么是“好”, 什么是“坏”
82     rew_scale_alive = 1.0          # 活着(没倒)就给分
83     rew_scale_terminated = -2.0    # 死了(倒了)倒扣分
84     rew_scale_cart_pos = 0         # 滑块位置惩罚(这里设为0, 表示不限制滑块位置)
85     rew_scale_cart_vel = -0.01    # 滑块速度惩罚(希望它动作平稳, 不要剧烈抖动)
86     rew_scale_pole_pos = -1.0     # 杆子角度惩罚(希望杆子垂直, 角度越小越好)
87     rew_scale_pole_vel = -0.01    # 杆子角速度惩罚
88     rew_scale_pendulum_pos = -1.0
89     rew_scale_pendulum_vel = -0.01
90
91     # --- 重置条件 (Reset) ---
92     # 每次重置时, 给关节角度加一点随机噪声, 增加鲁棒性
93     initial_pendulum_angle_range = [-0.25, 0.25] # [rad]
94     initial_pole_angle_range = [-0.25, 0.25]    # [rad]
95     max_cart_pos = 3.0 # 如果滑块跑出 3米 远, 就视为失败重置

```

XXXX_marl_env.py

```

1  # ... (Imports 省略)
2
3  class Test1MarlEnv(DirectMARLEnv):
4      # 指明配置类是上面定义的 Test1MarlEnvCfg
5      cfg: Test1MarlEnvCfg
6
7      def __init__(self, cfg: Test1MarlEnvCfg, render_mode: str | None =
None, **kwargs):
8          super().__init__(cfg, render_mode, **kwargs)
9
10         # -----
11         # 初始化: 获取关节索引

```



```

12         # -----
13         # 为了高效计算，我们不能每一帧都用字符串去查关节。
14         # 这里一次性查好关节对应的 index (整数)，存在 self 里。
15         self._cart_dof_idx, _ =
self.robot.find_joints(self.cfg.cart_dof_name)
16         self._pole_dof_idx, _ =
self.robot.find_joints(self.cfg.pole_dof_name)
17         self._pendulum_dof_idx, _ =
self.robot.find_joints(self.cfg.pendulum_dof_name)
18
19         # 创建快捷引用，方便后续直接访问关节位置和速度
20         self.joint_pos = self.robot.data.joint_pos
21         self.joint_vel = self.robot.data.joint_vel
22
23     def _setup_scene(self):
24         """场景构造函数：在仿真开始前运行一次"""
25         self.robot = Articulation(self.cfg.robot_cfg)
26         # 1. 生成地面
27         spawn_ground_plane(prim_path="/World/ground",
cfg=GroundPlaneCfg())
28         # 2. 克隆环境 (把上面的配置复制 4096 份)
29         self.scene.clone_environments(copy_from_source=False)
30         # 3. 碰撞过滤 (CPU 模式下优化性能，GPU 模式通常不需要)
31         if self.device == "cpu":
32             self.scene.filter_collisions(global_prim_paths=[])
33         # 4. 把机器人注册到场景管理器中
34         self.scene.articulations["robot"] = self.robot
35         # 5. 添加光照
36         light_cfg = sim_utils.DomeLightCfg(intensity=2000.0, color=(0.75,
0.75, 0.75))
37         light_cfg.func("/World/Light", light_cfg)
38
39     def _pre_physics_step(self, actions: dict[str, torch.Tensor]) -> None:
40         """物理步前钩子：保存神经网络传来的动作"""
41         self.actions = actions
42
43     def _apply_action(self) -> None:
44         """应用动作：将神经网络输出转换为物理力矩"""
45         # 给 Cart (滑块) 施加力
46         # actions["cart"] 是网络输出，乘以 cart_action_scale 变成真实的牛顿
47         self.robot.set_joint_effort_target(
48             self.actions["cart"] * self.cfg.cart_action_scale,
joint_ids=self._cart_dof_idx
49         )
50         # 给 Pendulum (摆杆) 施加力矩
51         self.robot.set_joint_effort_target(

```



```

52         self.actions["pendulum"] * self.cfg.pendulum_action_scale,
joint_ids=self._pendulum_dof_idx
53     )
54
55     def _get_observations(self) -> dict[str, torch.Tensor]:
56         """获取观测：定义智能体能看到什么数据"""
57         # 将关节角度归一化到  $[-\pi, \pi]$  之间
58         pole_joint_pos = normalize_angle(self.joint_pos[:,
self._pole_dof_idx[0]].unsqueeze(dim=1))
59         pendulum_joint_pos = normalize_angle(self.joint_pos[:,
self._pendulum_dof_idx[0]].unsqueeze(dim=1))
60
61         observations = {
62             "cart": torch.cat(
63                 (
64                     # Cart 看到：自己的位置 + 自己的速度 + 杆的角度 + 杆的角速度
65                     self.joint_pos[:,
self._cart_dof_idx[0]].unsqueeze(dim=1),
66                     self.joint_vel[:,
self._cart_dof_idx[0]].unsqueeze(dim=1),
67                     pole_joint_pos,
68                     self.joint_vel[:,
self._pole_dof_idx[0]].unsqueeze(dim=1),
69                 ),
70                 dim=-1,
71             ),
72             "pendulum": torch.cat(
73                 (
74                     # Pendulum 看到：两杆角度之和(可能是绝对角度) + 自己的相对角度
+ 自己的角速度
75                     pole_joint_pos + pendulum_joint_pos,
76                     pendulum_joint_pos,
77                     self.joint_vel[:,
self._pendulum_dof_idx[0]].unsqueeze(dim=1),
78                 ),
79                 dim=-1,
80             ),
81         }
82         return observations
83
84     def _get_rewards(self) -> dict[str, torch.Tensor]:
85         """计算奖励：调用下面的 JIT 函数计算"""
86         # 收集所有需要的数据传给 compute_rewards 函数
87         total_reward = compute_rewards(
88             self.cfg.rew_scale_alive,
89             self.cfg.rew_scale_terminated,

```



```

90         self.cfg.rew_scale_cart_pos,
91         self.cfg.rew_scale_cart_vel,
92         self.cfg.rew_scale_pole_pos,
93         self.cfg.rew_scale_pole_vel,
94         self.cfg.rew_scale_pendulum_pos,
95         self.cfg.rew_scale_pendulum_vel,
96         self.joint_pos[:, self._cart_dof_idx[0]], # 滑块位置
97         self.joint_vel[:, self._cart_dof_idx[0]], # 滑块速度
98         normalize_angle(self.joint_pos[:, self._pole_dof_idx[0]]), #
杆角度
99         self.joint_vel[:, self._pole_dof_idx[0]], # 杆速度
100        normalize_angle(self.joint_pos[:, self._pendulum_dof_idx[0]]),
# 摆角度
101        self.joint_vel[:, self._pendulum_dof_idx[0]], # 摆速度
102        math.prod(self.terminated_dict.values()), # 是否结束
103    )
104    return total_reward
105
106    def _get_dones(self) -> tuple[dict[str, torch.Tensor], dict[str,
torch.Tensor]]:
107        """判断回合是否结束 (Done)"""
108        # 更新一下最新的物理状态
109        self.joint_pos = self.robot.data.joint_pos
110        self.joint_vel = self.robot.data.joint_vel
111
112        # 1. 超时判断: 是否超过了最大时长 (5秒)
113        time_out = self.episode_length_buf >= self.max_episode_length - 1
114
115        # 2. 出界判断 (Fail):
116        # - 滑块跑出 +/- 3米
117        out_of_bounds = torch.any(torch.abs(self.joint_pos[:,
self._cart_dof_idx]) > self.cfg.max_cart_pos, dim=1)
118        # - 或者 杆子倒得太厉害 (超过 90度, 即 pi/2)
119        out_of_bounds = out_of_bounds |
torch.any(torch.abs(self.joint_pos[:, self._pole_dof_idx]) > math.pi / 2,
dim=1)
120
121        # 构建返回值字典
122        terminated = {agent: out_of_bounds for agent in
self.cfg.possible_agents} # 失败结束
123        time_outs = {agent: time_out for agent in
self.cfg.possible_agents} # 时间到结束
124        return terminated, time_outs
125
126    def _reset_idx(self, env_ids: Sequence[int] | None):
127        """重置环境: 当环境 Done 了之后如何复位"""

```



```

128         if env_ids is None:
129             env_ids = self.robot._ALL_INDICES
130         super()._reset_idx(env_ids)
131
132         # 1. 获取默认姿态
133         joint_pos = self.robot.data.default_joint_pos[env_ids]
134
135         # 2. 添加随机噪声 (Domain Randomization)
136         # 在初始角度上加一点 [-0.25pi, 0.25pi] 的随机值, 防止过拟合
137         joint_pos[:, self._pole_dof_idx] += sample_uniform(
138             self.cfg.initial_pole_angle_range[0] * math.pi,
139             self.cfg.initial_pole_angle_range[1] * math.pi,
140             joint_pos[:, self._pole_dof_idx].shape,
141             joint_pos.device,
142         )
143         joint_pos[:, self._pendulum_dof_idx] += sample_uniform(
144             self.cfg.initial_pendulum_angle_range[0] * math.pi,
145             self.cfg.initial_pendulum_angle_range[1] * math.pi,
146             joint_pos[:, self._pendulum_dof_idx].shape,
147             joint_pos.device,
148         )
149
150         joint_vel = self.robot.data.default_joint_vel[env_ids]
151         default_root_state = self.robot.data.default_root_state[env_ids]
152         default_root_state[:, :3] += self.scene.env_origins[env_ids]
153
154         # 3. 将这些修改后的状态写入到仿真器内存中
155         self.joint_pos[env_ids] = joint_pos
156         self.joint_vel[env_ids] = joint_vel
157
158         self.robot.write_root_pose_to_sim(default_root_state[:, :7],
159 env_ids)
160         self.robot.write_root_velocity_to_sim(default_root_state[:, 7:],
161 env_ids)
162         self.robot.write_joint_state_to_sim(joint_pos, joint_vel, None,
163 env_ids)
164
165         # -----
166         # 辅助函数 (使用 JIT 加速)
167         # -----
168         # @torch.jit.script 装饰器意味着这部分代码会被编译成 C++ 级运行, 极大提升 GPU 上的
169         # 执行速度
170
171         @torch.jit.script
172         def normalize_angle(angle):

```



```

170     """辅助函数：将角度标准化到 [-pi, pi] 之间"""
171     return (angle + math.pi) % (2 * math.pi) - math.pi
172
173
174 @torch.jit.script
175 def compute_rewards(
176     # ... 参数列表 ...
177     rew_scale_alive: float,
178     # ...
179     reset_terminated: torch.Tensor,
180 ):
181     """
182     具体的奖励计算逻辑
183     """
184     # 存活奖励：只要没失败(terminated)，每一步都给分。鼓励活得久。
185     rew_alive = rew_scale_alive * (1.0 - reset_terminated.float())
186
187     # 死亡惩罚：如果失败了，扣一大笔分。
188     rew_termination = rew_scale_terminated * reset_terminated.float()
189
190     # 姿态惩罚：使用 square（平方）惩罚，意味着离目标越远惩罚越重
191     # 希望 pole_pos（杆角度）接近 0
192     rew_pole_pos = rew_scale_pole_pos *
193     torch.sum(torch.square(pole_pos).unsqueeze(dim=1), dim=-1)
194
195     # 希望整个连杆结构也是直的
196     rew_pendulum_pos = rew_scale_pendulum_pos * torch.sum(
197         torch.square(pole_pos + pendulum_pos).unsqueeze(dim=1), dim=-1
198     )
199
200     # 能量/动作惩罚：希望速度小一点，不要剧烈晃动
201     rew_cart_vel = rew_scale_cart_vel *
202     torch.sum(torch.abs(cart_vel).unsqueeze(dim=1), dim=-1)
203     rew_pole_vel = rew_scale_pole_vel *
204     torch.sum(torch.abs(pole_vel).unsqueeze(dim=1), dim=-1)
205     rew_pendulum_vel = rew_scale_pendulum_vel *
206     torch.sum(torch.abs(pendulum_vel).unsqueeze(dim=1), dim=-1)
207
208     # 汇总奖励字典
209     total_reward = {
210         "cart": rew_alive + rew_termination + rew_pole_pos + rew_cart_vel
211         + rew_pole_vel,
212         "pendulum": rew_alive + rew_termination + rew_pendulum_pos +
213         rew_pendulum_vel,
214     }
215     return total_reward

```


Create

Owned by me

Recently opened

Created by me







































Shared with me

Labels

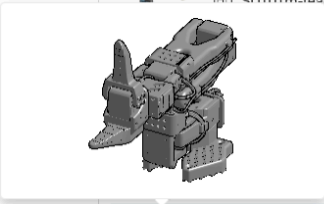
Public

Trash

Search results in Public

Name	Modified	Owned by	Copies	Links
 Par...  M... SO101-0; SFL123...  In	1:03 AM Mar 1 2025	mstoerzel	0	1
 so101  Main	12:55 AM May 16 2025	Brett Peters	0	1
 so101  Main	4:56 AM Sep 30 2025	Isaiah Bjorklund	0	0
 so101  Main	12:30 AM Dec 8 2025	Jafar Uruc	0	1
 SO101  Main  In progress	3:31 AM May 21 2025	Awantika Bastola	2	0
 SO101  Main  In progress	12:18 PM May 27 2025	mason knittle	0	0
 SO101 Assembly...  Main	12:24 PM Jan 16	Dillinski	0	0
 SO101 Assembly...  M...  In prc	12:24 PM Jan 16	Dillinski	0	0
 SO101 Assembl...  M...  In prog	12:24 PM Jan 16	Dillinski	0	0
 SO101m-leader  Main	1:08 AM Dec 28 2025	kgdo	0	0
 mbly...  Main	8:47 PM Dec 22 2025	Fred Eple	0	0
 mbly...  M...  In prc	8:47 PM Dec 22 2025	Fred Eple	0	0
 mbly...  M...  In prog	8:47 PM Dec 22 2025	Fred Eple	0	0
 mbly...  Main	8:47 PM Dec 22 2025	Fred Eple	0	0
 SO101 Assem...  M...  In progre	2:05 PM Nov 20 2025	Seunghwan Lee	0	0

Subscription: Student



<https://github.com/Rhoban/onshape-to-robot.git>

Rhoban / onshape-to-robot

<> Code

Issues 6

Pull requests 9

Actions

Projects

Security

Insights

onshape-to-robot

Public

Watch

16

master

6 Branches

97 Tags

Go to file

t

Add file


<> Code

Gregwar v1.7.9

b16144d · 2 months ago


708 Commits


.github/workflows	Wheels workflow	8 months ago
docs	Typo	4 months ago
onshape_to_robot	Adding "m" unit (see #177)	2 months ago
.gitignore	Ignoring uv.lock & .env files	2 months ago
.readthedocs.yaml	Referencing doc requirements	3 years ago
LICENSE	MIT	7 years ago
MANIFEST.in	Include README-pypi.md in manifest	8 months ago
Makefile	Changing architecture, packaging PyPI	6 years ago
README-pypi.md	Writing detail (OnShape -> Onshape)	11 months ago
README.md	Writing detail (OnShape -> Onshape)	11 months ago
pyproject.toml	v1.7.9	2 months ago
requirements.txt	fix: Add optional dependency to setup	11 months ago



main

SO-ARM100 / Simulation / SO101



 **Gregory119** Update actuator model params (#141)

Name	Last commit message
..	
assets	Add SO101 mujoco and urdf (#80)
README.md	Update URDF (#83)
joints_properties.xml	Add old and new simulation files for so101 (#86)
scene.xml	Add old and new simulation files for so101 (#86)
so101_new_calib.urdf	fix(urdf): fixing multiple issues related to the last URDF update (#117)
so101_new_calib.xml	Update actuator model params (#141)
so101_old_calib.urdf	Updated the new so101 calibration urdf with fixed direction and frame (...)
so101_old_calib.xml	Update Mujoco XML files with colors (#115)