Learning to Walk: Emergent Behaviors in Multi-Agent Humanoid Simulation

*A brief report on physical simulation & RL experiments based on **Nvidia Isaaclab***

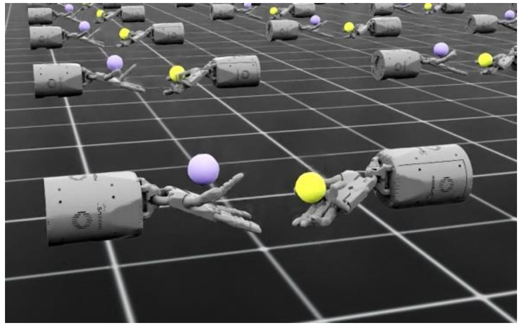
Zhao Zilin, zhao_zilin@outlook.com
College of Software, Jilin University

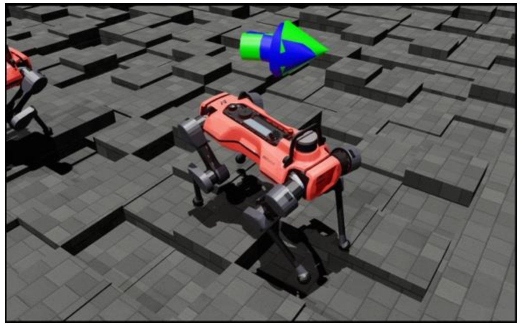
CONTENT

Install Isaaclab

3 common misconceptions and recommended configuration methods

I. Install Isaaclab





NVIDIA Isaac Lab

Actuators	Controllers	Environments	Tasking Framework	Spawners
Sensors	Imitation Learning	Reinforcement Learning	Randomization	Terrain Generation

NVIDIA Isaac Sim

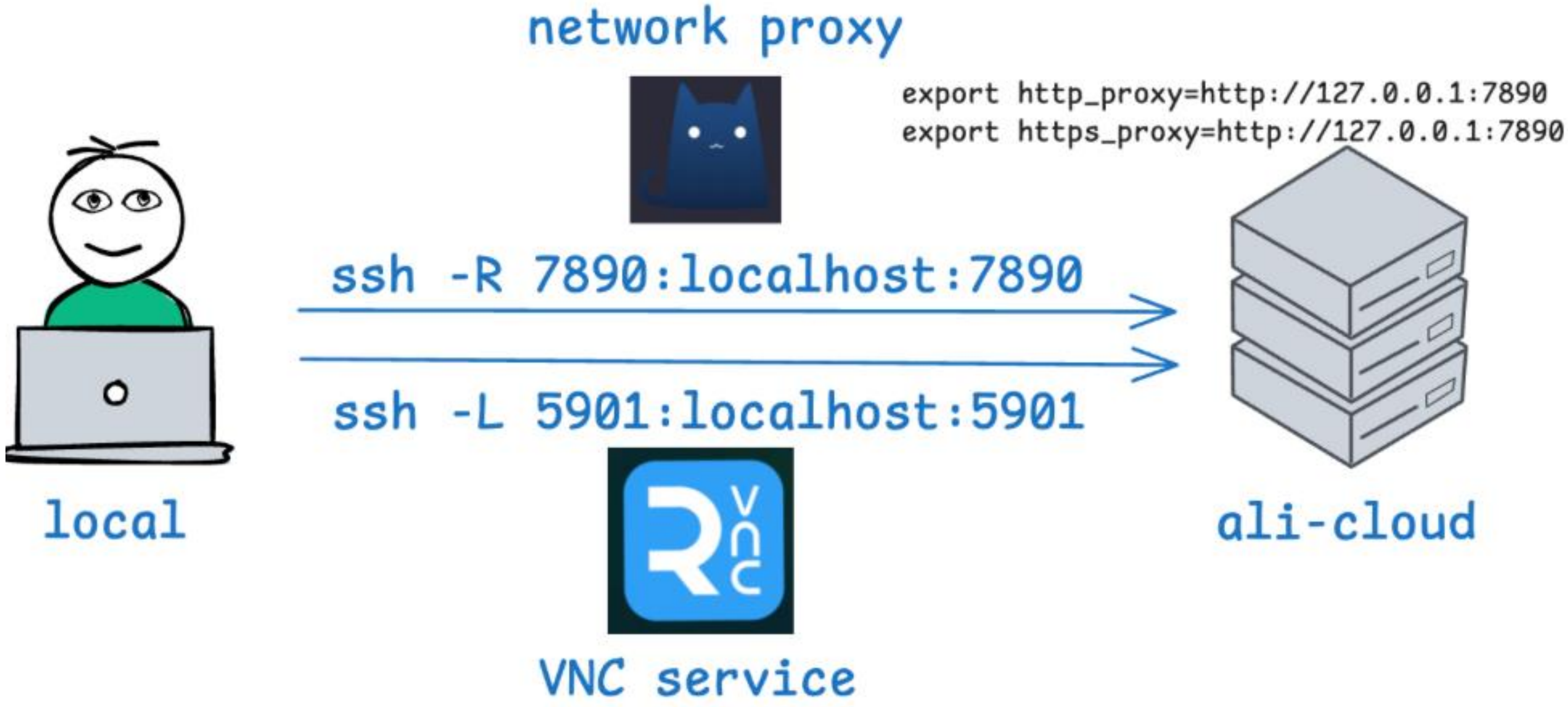
OmniGraph	Robot Assembler	ROS	Synthetic Data Generation	URDF/MJCF Importer

NVIDIA Omniverse

Physics	Rendering	UI	USD	Viewport

1. Building on online GPU platforms like AutoDL
2. Using GPUs without RT core, like A100, H100
3. Build locally without Nvidia Drivers

吉林大学


network proxy
export http_proxy=http://127.0.0.1:7890
export https_proxy=http://127.0.0.1:7890
ssh -R 7890:localhost:7890
ssh -L 5901:localhost:5901
local
ali-cloud
VNC service

Scene Construction

Simple scene content design, light source, robot basic control

II. Scene Contr.: requests

Basic Requests:

1. Scene construction:

-Create a 10m x 10m indoor flat scene using Isaac Sim (optional: add obstacles/slopes)

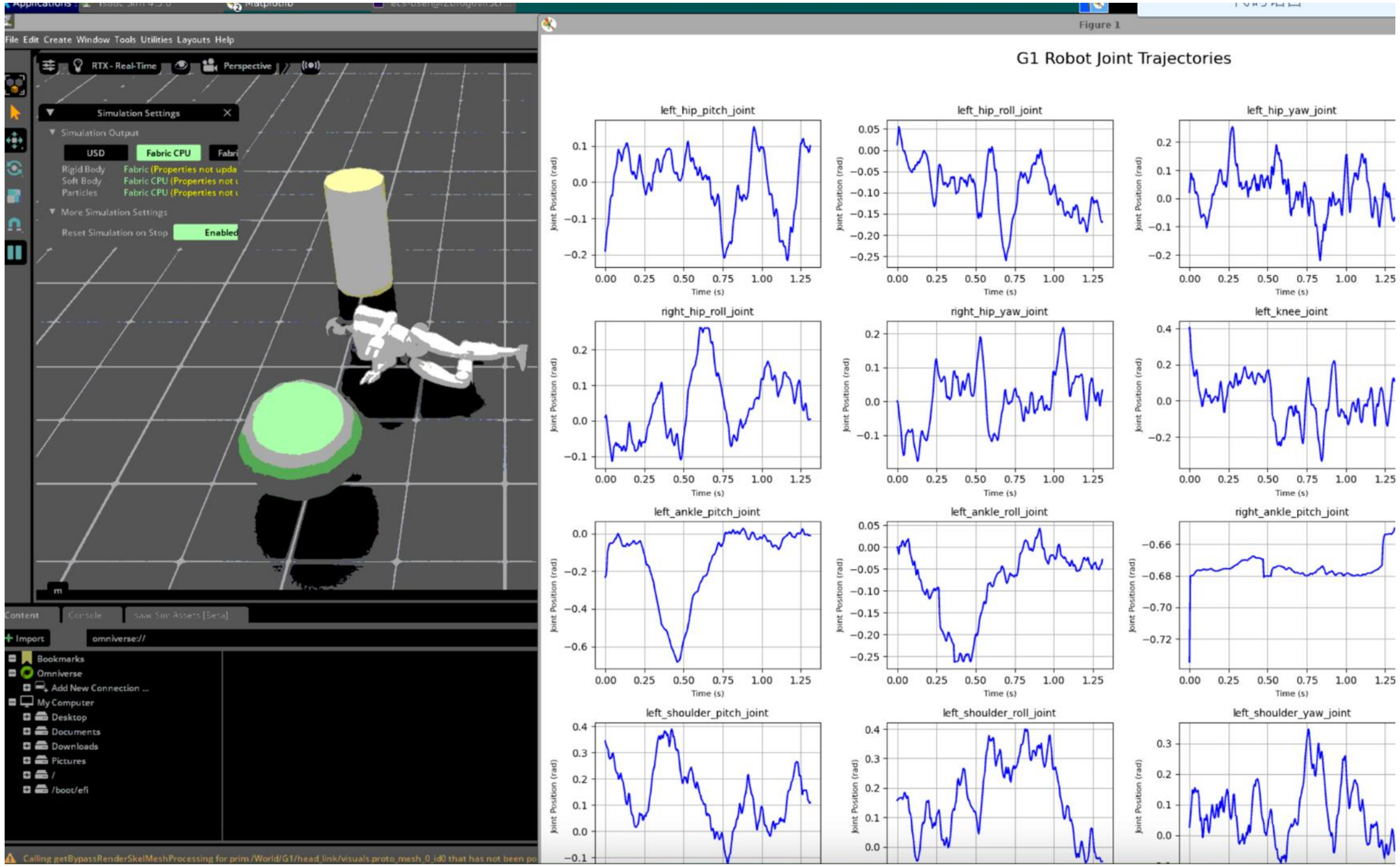
-Add 1 humanoid robot (such as Unitree H1/Ge2, etc.)

-Deploy basic lighting and physical materials

2. Motion control test

-Control robots to complete basic actions through Isaac Lab's Python API

-Output the 5-second joint angle change curve of the robot completing the standing action



```python
for i, name in enumerate(joint_names):
    set = False
    # 跳过右腿关节 (pitch方向) 保持踢腿动作
    if any(key in name for key in ["right_hip_pitch_joint", "right_knee_joint", "right_ankle_pitch_joint"]):
        continue  # 不修改踢腿姿态
    # 匹配上肢关节
    for key, (low, high) in upper_body_joint_ranges.items():
        if key in name:
            joint_pos[:, i] = torch.rand(1, device=sim.device) * (high - low) + low
            set = True
            break
    # 匹配下肢关节
    if not set:
        for key, (low, high) in lower_body_joint_ranges.items():
            if key in name:
                joint_pos[:, i] = torch.rand(1, device=sim.device) * (high - low) + low
                set = True
                break
    # 如果没有匹配到任何设置，则使用默认小幅扰动
    if not set:
        joint_pos[:, i] = torch.rand(1, device=sim.device) * 0.2 - 0.1  # [-0.1, 0.1]
```

II. Scene Contr.: extendent

Animation Stage Recorder (Beta)

Overview

The Stage recorder enables the user to capture the motion and USD property changes within a USD stage. With options to write directly to a layer or to write to disk as a timeSampled USD file. The stage recorder will record at the framerate specified in your timeline with the exception of when “live Mode” is enabled - In live mode you can specify a max target framerate in the recorder. The recorder uses the same hierarchy in the output USD as existed at the time of recording, So the layer structure and asset names must match when applying the animation as a layer.

User Manual

To Load the Stage Recorder - enable the extension omni.kit.stageRecorder To access the Stage Recorder window - Click the “record” Icon in the timeline window or go to Window/Animation/StageRecorder

Surprisingly, the debugging difficulty of recording videos is very high

- The memories of large models are all old APIs
- The official documentation is currently being updated, and some tools have bugs
    - The stage recorder[1] function is in beta version, and the recording file is missing the model
    - ReplicanBasicWriter[2] is helpful

[1] https://isaac-sim.github.io/IsaacLab/main/source/how-to/record_animation.html
[2] https://isaac-sim.github.io/IsaacLab/main/source/how-to/save_camera_output.html

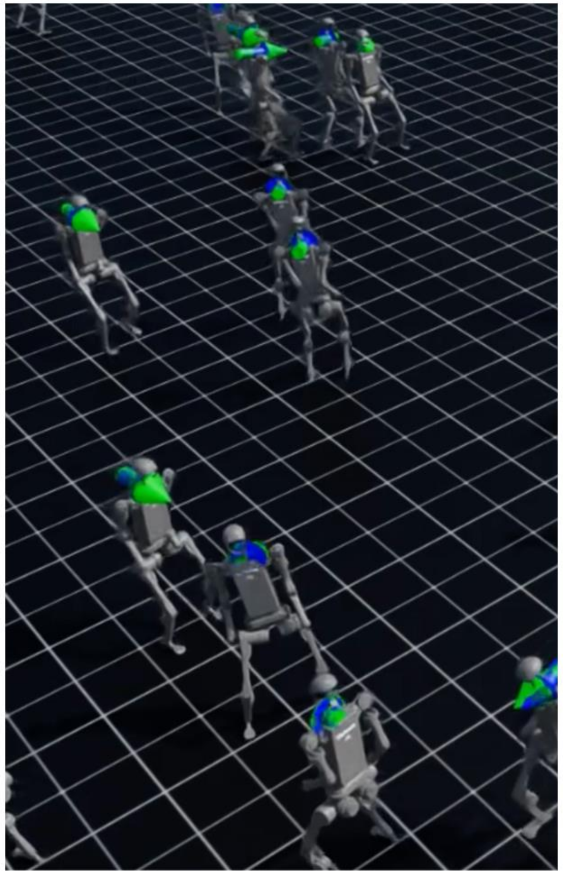
Parallel simulation

Isaaclab and Unitree provide official RL demos respectively

Objective: To reproduce the walking strategy of H1 robot based on open-source code from Unitree

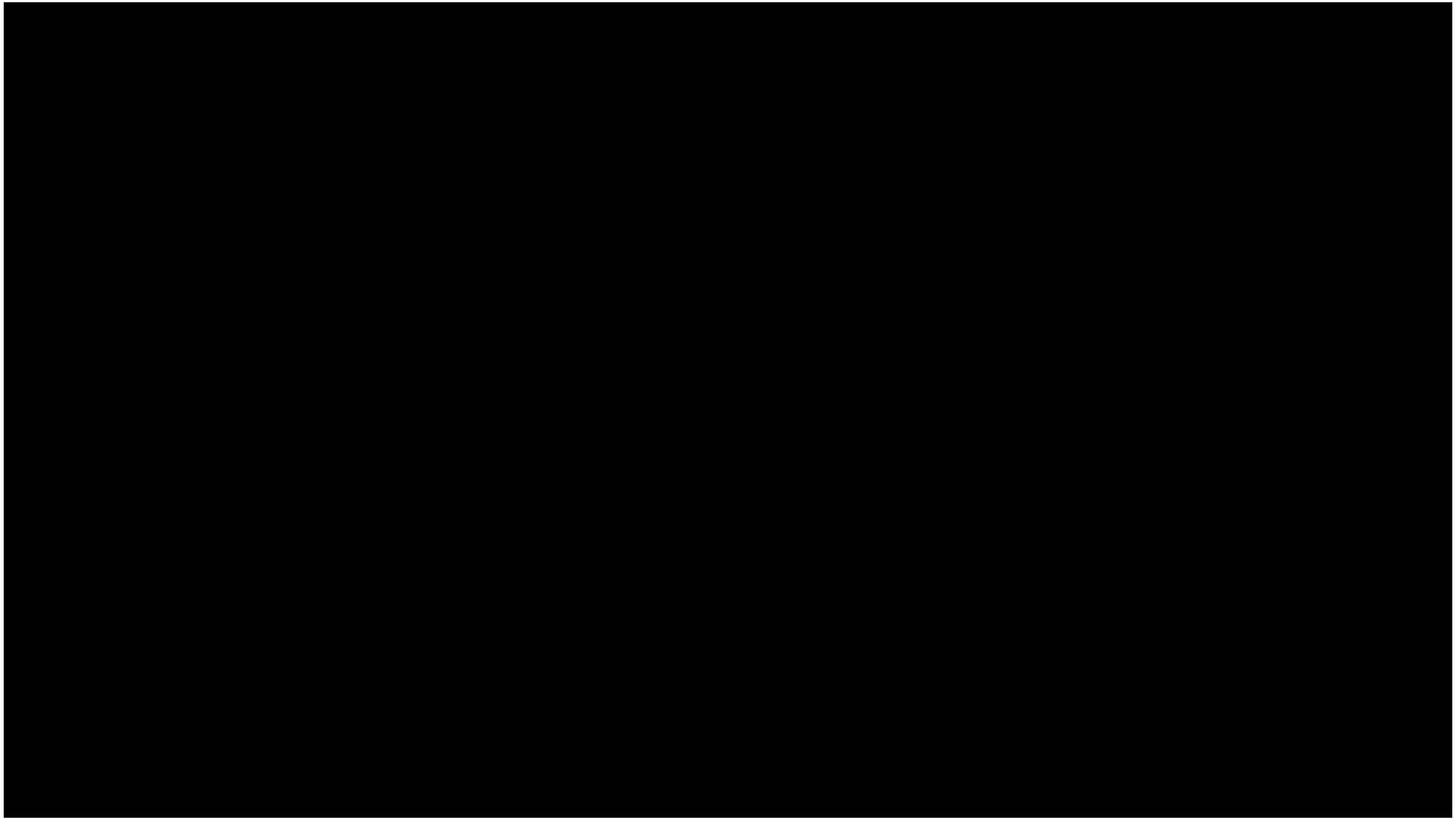
1. Code deployment and training
-Use the reinforcement learning example provided by Unitree official website
-Design of reward function analysis: speed tracking reward; Punishment for gait symmetry, etc

Isaac-Velocity-Flat-H1-v0



Isaac-Velocity-Rough-H1-v0

III. Parallel simu.



Unitree Official repo:

```
...
  File "/home/ecs-user/miniconda3/envs/is/lib/python3.10/site-packages/torch/distributions/normal.py", line 73, in sample
    return torch.normal(self.loc.expand(shape), self.scale.expand(shape))
RuntimeError: normal expects all elements of std >= 0.0
```

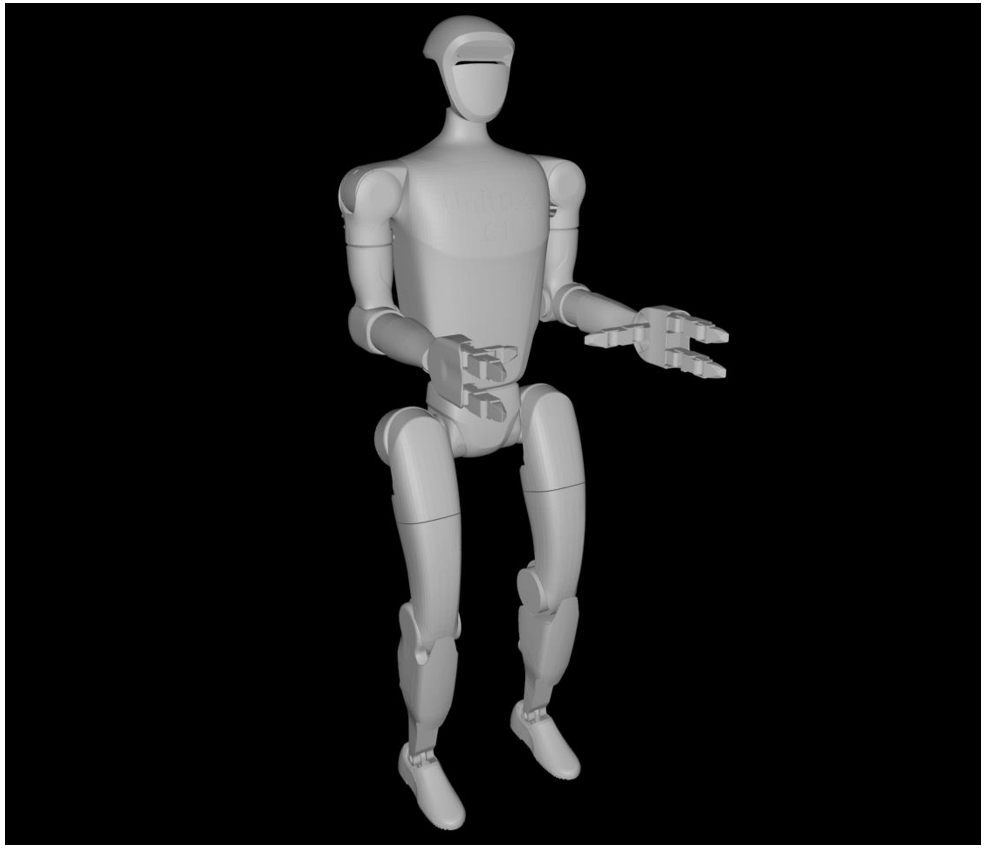
Isaaclab Official repo:

```
...
Traceback (most recent call last):
  File "/home/ecs-user/IsaacLab/scripts/demos/h1_locomotion.py", line 50, in <module>
    from omni.kit.viewport.utility import get_viewport_from_window_name
ModuleNotFoundError: No module named 'omni.kit.viewport'
```


[1] https://github.com/unitreerobotics/unitree_rl_lab/issues/7
[2] https://github.com/isaac-sim/IsaacLab/issues/2858

Strategy Migration

Cross platform gait migration



Goal: Transfer training strategies to other humanoid robots

Model adaptation

- Import a third-party robot URDF file (aligning joint naming conventions, such as. * d_hip_pitch)
- Adjust dynamic parameters, such as joint torque limit (third-party robot motor torque may be lower than Unitree H1)/leg mass distribution

Transfer the strategy of H1 to G1

1. G1 obtains observations from the environment
2. Observations from G1 to H1
3. Call the H1 strategy model to obtain H1 actions
4. Actions from H1 to G1
5. Iterative simulation

Code probe 1: Runtime Analysis

```python
def find_robot(env): #在嵌套环境中递归的寻找名为robot的对象
    max_depth = 10
    for _ in range(max_depth):
        if hasattr(env, "scene"):
            if hasattr(env.scene, "robot"):
                return env.scene.robot
            if hasattr(env.scene, "articulations") and "robot" in env.scene.articulations:
                return env.scene.articulations["robot"]
        if hasattr(env, "envs"):
            env = env.envs[0]
        elif hasattr(env, "env"):
            env = env.env
        elif hasattr(env, "unwrapped"):
            env = env.unwrapped
        else:
            break
    raise AttributeError("Could not find robot in environment.")
```

```python
h1_joint_names =
["left_hip_yaw",
"right_hip_yaw", "torso",
"left_hip_roll",
"right_hip_roll",
"left_shoulder_pitch",
"right_shoulder_pitch",
"left_hip_pitch",
"right_hip_pitch",
"left_shoulder_roll",
"right_shoulder_roll",
"left_knee", "right_knee",
"left_shoulder_yaw",
"right_shoulder_yaw",
"left_ankle",
"right_ankle",
"left_elbow",
"right_elbow"]
```

Code probe 2: Network packet capt.

```
(is) ecs-user@iZuf6g6vii5ci3ikxssy9lZ:~/IsaacLab$ source ~/proxy-on.sh 9999 # 设置一个不存在的代理
127.0.0.1:9999
(is) ecs-user@iZuf6g6vii5ci3ikxssy9lZ:~/IsaacLab$ curl baidu.com # 确认一下网络是否真的失效了
curl: (7) Failed to connect to 127.0.0.1 port 9999 after 0 ms: Connection refused
(is) ecs-user@iZuf6g6vii5ci3ikxssy9lZ:~/IsaacLab$ python
scripts/reinforcement_learning/rsl_rl/train.py --task=Isaac-Velocity-Flat-G1-v0 --headless
...
File "/home/ecs-user/IsaacLab/source/isaaclab/isaaclab/sim/spawners/from_files/from_files.py",
line 66, in spawn_from_usd
    return _spawn_from_usd_file(prim_path, cfg.usd_path, cfg, translation, orientation)
  File "/home/ecs-user/IsaacLab/source/isaaclab/isaaclab/sim/spawners/from_files/from_files.py",
line 237, in _spawn_from_usd_file
    raise FileNotFoundError(f"USD file not found at path: '{usd_path}'.")
FileNotFoundError: USD file not found at path: 'http://omniverse-content-production.s3-us-west-
2.amazonaws.com/Assets/Isaac/4.5/Isaac/IsaacLab/Robots/Unitree/G1/g1_minimal.usd'.
```

VI. Strategy Migir.: actions alignment

吉林大学

```
--- Exploring: /home/ecs-user/working-labs/H1transferG1/assets/g1_minimal.usd ---
Prim: /physicsScene  Type: PhysicsScene
Prim: /g1  Type: Xform
Prim: /g1/pelvis  Type: Xform
Prim: /g1/pelvis/visuals  Type:
Prim: /g1/pelvis/left_hip_pitch_joint  Type: PhysicsRevoluteJoint
Prim: /g1/pelvis/pelvis_contour_joint  Type: PhysicsFixedJoint
Prim: /g1/pelvis/right_hip_pitch_joint  Type: PhysicsRevoluteJoint
...
```

```python
def explore_usd_file(path):
    stage = Usd.Stage.Open(path)
    if not stage:
        print(f"Failed to open {path}")
        return
    print(f"\n--- Exploring: {path} ---")
    # 遍历所有 Prim (物体节点)
    for prim in stage.Traverse():
        print(f"Prim: {prim.GetPath()}  Type: {prim.GetTypeName()}")
        # 示例：如果是 Mesh，打印点数
        if prim.GetTypeName() == "Mesh":
            mesh = UsdGeom.Mesh(prim)
            points_attr = mesh.GetPointsAttr()
            points = points_attr.Get()
            print(f"  Mesh with {len(points)} points")
```

Code probe 3: Torch model analysis

```
Actor MLP: Sequential(
  (0): Linear(in_features=69, out_features=128, bias=True)
  (1): ELU(alpha=1.0)
  (2): Linear(in_features=128, out_features=128, bias=True)
  (3): ELU(alpha=1.0)
  (4): Linear(in_features=128, out_features=128, bias=True)
  (5): ELU(alpha=1.0)
  (6): Linear(in_features=128, out_features=19, bias=True)
)
```

VI. Strategy Migir.: actions alignment

吉林大学

```python
H1_TO_G1_JOINT_MAP = {
    # 下肢（左腿）
    "left_hip_yaw": "left_hip_yaw_joint",
    "left_hip_roll": "left_hip_roll_joint",
    "left_hip_pitch": "left_hip_pitch_joint",
    "left_knee": "left_knee_joint",
    "left_ankle": "left_ankle_pitch_joint",
    # G1多出的踝关节roll
    "left_ankle_roll_joint": 0.0,
    # 下肢（右腿）
    "right_hip_yaw": "right_hip_yaw_joint",
    "right_hip_roll": "right_hip_roll_joint",
    "right_hip_pitch": "right_hip_pitch_joint",
    "right_knee": "right_knee_joint",
    "right_ankle": "right_ankle_pitch_joint",
    # G1多出的踝关节roll
    "right_ankle_roll_joint": 0.0,
    # 躯干
    "torso": "torso_joint",
```

1. G1 obtains observations from the environment
2. Observations from G1 to H1
3. Call the H1 strategy model to obtain H1 actions
4. Actions from H1 to G1
5. Iterative simulation

```python
def map_h1_action_to_g1(h1_action, h1_joint_names=h1_joint_names, g1_joint_names=g1_joint_names):
    # H1关节名到action值的映射
    h1_action_dict = dict(zip(h1_joint_names, h1_action))
    g1_action = []
    for g1_joint in g1_joint_names:
        # 反查映射表，找到H1关节名
```

Check the source code to obtain preliminary structure

source/isaaclab_tasks/isaaclab_tasks/manager_based/locomotion/velocity/velocity_env_cfg.py

Observation =

- `base_lin_vel`, `base_ang_vel`
- `projected_gravity`
- `velocity_commands`
- `joint_pos`, `joint_vel`
- `actions`
- `height_scan`

Code probe: observation extract

```python
# 沿用上一节提到的--extrat参数
# ====== 新增：提取 observation 示例和 shape ======
obs = env.reset()
import numpy as np
print("\nExtracted observation structure:")
if isinstance(obs, (tuple, list)) and len(obs) > 0 and isinstance(obs[0], dict) and 'policy' in obs[0]:
    policy_obs = obs[0]['policy']
    print(f"policy: shape={policy_obs.shape} dtype={getattr(policy_obs, 'dtype', type(policy_obs))}")
    # 打印第一个环境的 observation 向量
    arr = policy_obs[0].detach().cpu().numpy() if hasattr(policy_obs[0], 'detach') else np.array(policy_obs[0])
    print(f"  First row (len={len(arr)}): {arr}")
    print(f"  First 20 values: {arr[:20]}")
    if hasattr(policy_obs, 'detach'):
        policy_obs = policy_obs.detach().cpu().numpy()
```

```
Extracted observation structure:
policy: shape=torch.Size([4096, 69]) dtype=torch.float32
  First row (len=69): [-5.6066342e-02 -8.7517925e-02  4.9868785e-02 -9.6398674e-02
```

solve equations:

-H1 has 19 joints and an observation space of 69 dimensions
-G1 has 37 joints and an observation space of 123 dimensions

- `base_lin_vel`, `base_ang_vel`
- `projected_gravity`
- `velocity_commands`
- `joint_pos`, `joint_vel`
- `actions`
- `height_scan`

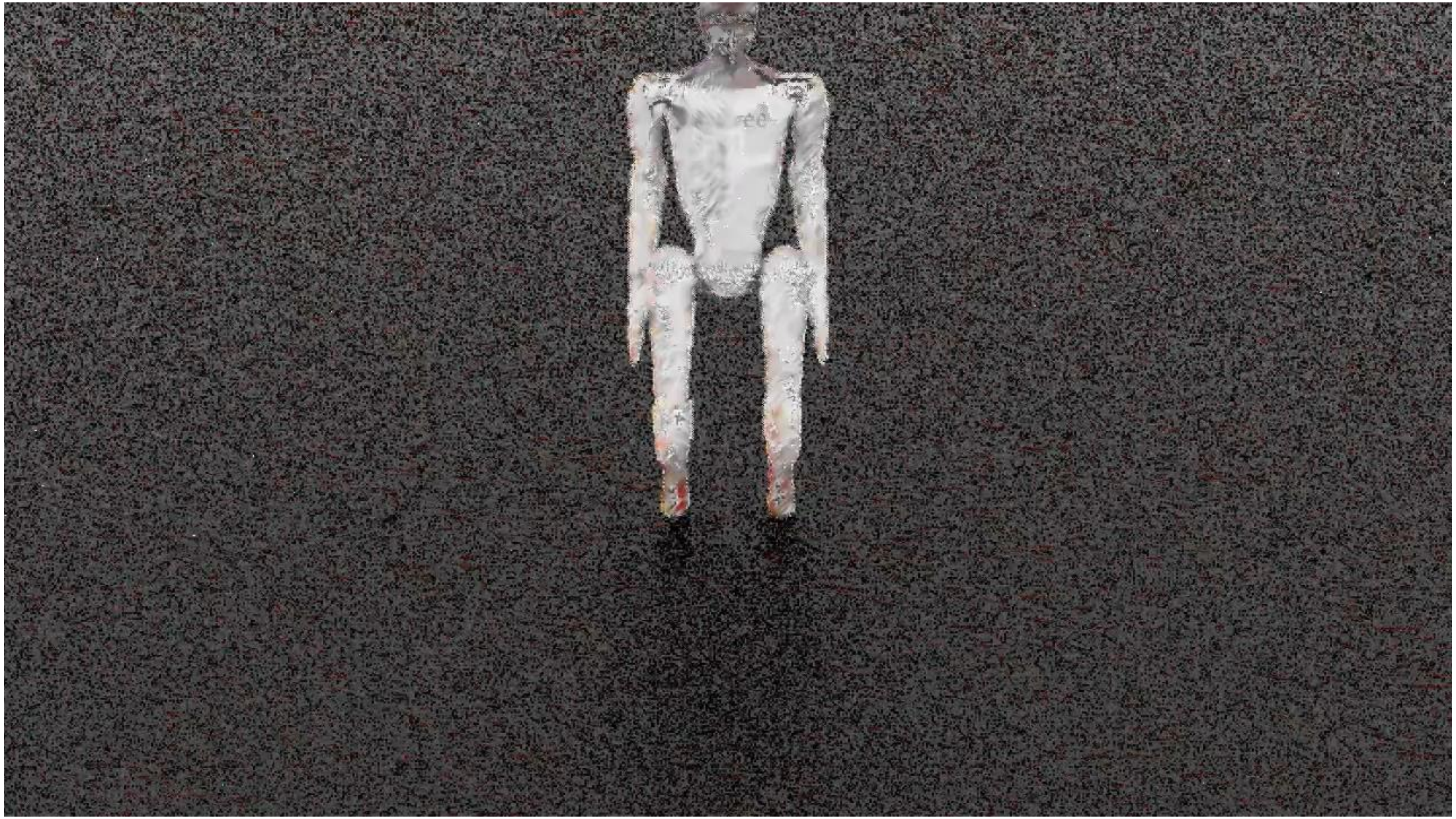
H1			G1		
[0:3]	base_lin_vel	(3)	[0:3]	base_lin_vel	(3)
[3:6]	base_ang_vel	(3)	[3:6]	base_ang_vel	(3)
[6:9]	projected_gravity	(3)	[6:9]	projected_gravity	(3)
[9:12]	velocity_commands	(3)	[9:12]	velocity_commands	(3)
[12:31]	joint_pos	(19)	[12:49]	joint_pos	(37)
[31:50]	joint_vel	(19)	[49:86]	joint_vel	(37)
[50:69]	actions	(19)	[86:123]	actions	(37)

吉林大学

Modify dynamic parameters and initial state

1. G1 obtains observations from the environment
2. Observations from G1 to H1
3. Call the H1 strategy model to obtain H1 actions
4. Actions from H1 to G1
5. Iterative simulation

```python
G1_CFG = ArticulationCfg(
...
    # 修改初始状态以匹配H1的配置
    init_state=ArticulationCfg.InitialStateCfg(
        pos=(0.0, 0.0, 0,74),  # 高度不用改
        joint_pos={
            # 下肢关节 - 映射自H1
            ".*_hip_yaw_joint": 0.0,  # 对应H1的 *_hip_yaw
            ".*_hip_roll_joint": 0.0,  # 对应H1的 *_hip_roll
            ".*_hip_pitch_joint": -0.28,  # 对应H1的 *_hip_pitch (-16度)
            ".*_knee_joint": 0.79,  # 对应H1的 *_knee (45度)
            ".*_ankle_pitch_joint": -0.52,  # 对应H1的 *_ankle (-30度)
            ".*_ankle_roll_joint": 0.0,  # G1特有, H1没有, 设为0
```



Please refer to the document for analysis and improvement

Brief Summary

吉林大学

阶段一：基础环境搭建

1. 场景构建：

- 使用 Isaac Sim 创建 10m×10m 的室内平面场景（可选：增加障碍物/斜坡）
- 添加 1 个人形机器人（例如宇树 H1/Go2 等）
- 部署基础光源和物理材质

扩展：设置相机并导出图片，再用ffmpeg录制

2. 运动控制测试

- 通过 Isaac Lab 的 Python API 控制机器人完成基础动作
- 输出机器人完成站立动作的 5 秒关节角度变化曲线图

阶段二：复现宇树机器人行走

目标：基于宇树开源代码实现 H1 机器人行走策略复现

1. 代码部署与训练

- 使用宇树官方提供的强化学习示例、
- 解析奖励函数设计：速度跟踪奖励；步态对称性惩罚等

扩展：提了两个issue

详见文档

阶段三：跨平台步态迁移

目标：将训练策略迁移至其他人形机器人

模型适配

亮点：代码探针，运行时分析，递归搜索，网络抓包

- 导入第三方机器人 URDF 文件（需对齐关节命名规范，如.*_hip_pitch）
- 调整动力学参数：例如关节力矩限制（第三方机器人电机扭矩可能低于宇树 H1）/腿部质量分布