**Unitree G1 Overview**

**Camera (Intel Realsense d435i)**

ros2 launch realsense2\_camera rs\_launch.py enable\_color:=true enable\_depth:=true

**Livox LiDAR Mid360**

~/livox\_ws$ source install/setup.bash

ros2 launch livox\_ros\_driver2 msg\_MID360\_launch.py

**ssh commands**

ssh -X unitree@192.168.124.100 --> for Wifi (.wificonnect.sh)

ssh -X unitree@192.168.123.164 --> for ethernet

**ROS Onboard**: ROS2 Foxy, ROS1 Noetic

**Odometry Data Source (Topic)**:

/odommodestate --> Position, velocity, IMU data (Orientation, acceleration, etc.)

**Transforms**:

pelvis

├── pelvis\_contour\_link     ──┐

├── left\_hip\_pitch\_link ───► left\_hip\_roll\_link ───► left\_hip\_yaw\_link

├── right\_hip\_pitch\_link ──► right\_hip\_roll\_link ──► right\_hip\_yaw\_link

├── waist\_yaw\_link ─────────► waist\_roll\_link ────► torso\_link

│                                                                                                             ├─► logo\_link

│                                                                                                             ├─► head\_link

│                                                                                                             ├─► waist\_support\_link

│                                                                                                             ├─► imu\_in\_torso

│                                                                                                             ├─► d435\_link

│                                                                                                             ├─► mid360\_link

│                                                                                                             ├─► left\_shoulder\_pitch\_link ──► left\_shoulder\_roll\_link

│                                                                                                              └─► right\_shoulder\_pitch\_link ─► right\_shoulder\_roll\_link

└── imu\_in\_pelvis

**SLAM Video (ORB SLAM3 on rosbag data of Unitree G1 by Me & Shalini):** https://drive.google.com/file/d/1Ou4SMNP6buWFjMuDYDfJDBT81N5YQNke/view?usp=sharing

**Unitree G1 online SDK:** https://support.unitree.com/home/en/G1\_developer/get\_sdk

**GitHub page:** https://github.com/unitreerobotics

**Unitree RL Repo**

**ISAAC LAB**

<https://github.com/unitreerobotics/unitree_rl_lab>

NOT A SENSOR BASED APPROACH\*

PROPRIOCEPTIVE FEEDBACK based on:

* Joint positions and velocities
* IMU data (angular velocity, gravity direction)
* Contact force sensors on feet

Primary RL Algorithm:

RSL-RL (Robotic Systems Lab – Reinforcement Learning) Framework by ETH Zurich 🡪 core is PPO (Proximal Policy Optimization) algorithm -> Actor Critic architecture

File: scripts/rsl\_rl/train.py

Supports: Go2, H1, G1-29dof

**Core Files**:

source/unitree\_rl\_lab/ - Main library package

scripts/rsl\_rl/train.py - Training script

scripts/rsl\_rl/play.py - Inference/visualization

scripts/list\_envs.py - Environment listing

source/unitree\_rl\_lab/unitree\_rl\_lab/assets/robots/unitree.py - Robot definitions

Requires USD files from unitree\_model repository

**Deployment:**

deploy/robots/g1\_29dof/ - C++ controller for G1

Sim2Sim (unitree\_mujoco) → Sim2Real

**Training Approach**

G1 learns locomotion in Isaac Lab with 4096 parallel environments, each episode lasting 20 seconds with 4x decimation (200Hz → 50Hz control).

State Space (What G1 Observes)

Policy observations (with 5-step history):

* base\_ang\_vel: Angular velocity (scaled 0.2x, with noise) 🡪 how fast robot torso is rotating
* projected\_gravity: Gravity direction in robot frame (with noise)
* velocity\_commands: Target velocities to achieve (x, y, yaw)
  + For training, values start with:
    - lin\_vel\_x: -0.1 to 0.1 m/s (forward/backward)
    - lin\_vel\_y: -0.1 to 0.1 m/s (left/right strafe)
    - ang\_vel\_z: -0.1 to 0.1 rad/s (turning)
* joint\_pos\_rel: Joint positions relative to default (with noise)
* joint\_vel\_rel: Joint velocities (scaled 0.05x, with noise)
* last\_action: Previous action for temporal consistency

Critic (PPO algorithm) gets additional privileged info: base\_lin\_vel: Linear velocity (not available to deployed policy)

**Action Space**

Joint Position Control: Policy outputs joint position targets scaled by 0.25, added to default poses. Uses realistic actuator models (N7520, N5020, W4010) with torque-speed curves.

**Reward Function Design**

Primary locomotion rewards:

* + track\_lin\_vel\_xy: Track commanded X/Y velocity (weight: 1.0)
  + track\_ang\_vel\_z: Track commanded yaw rate (weight: 0.5)
  + gait: Reward proper bipedal walking pattern, 0.8s period (weight: 0.5)
  + feet\_clearance: Encourage foot lifting during swing (weight: 1.0)

Stability & efficiency penalties:

* base\_height: Maintain 0.78m height (weight: -10.0)
* flat\_orientation: Stay upright (weight: -5.0)
* energy: Minimize power consumption (weight: -2e-5)
* joint\_deviation\_\*: Keep arms/waist near default poses (varying weights)
* feet\_slide: Prevent foot slipping (weight: -0.2)
* undesired\_contacts: Avoid non-foot body contacts (weight: -1.0)

Domain Randomization

* Mass randomization: Torso mass ±1-3kg
* Friction randomization: 0.3-1.0 range
* External disturbances: Periodic velocity pushes every 5s
* Sensor noise: Added to all observations

**ISAAC GYM**

https://github.com/unitreerobotics/unitree\_rl\_gym

Worklfow: Train → Play (Isaac Gym) → Sim2Sim (test in Mujoco) → Sim2Real (deploy to hardware)

legged\_gym/scripts/train.py - Main RL training script for Isaac Gym

legged\_gym/scripts/play.py - Visualization/testing of trained policies

Support for 4 robot types: go2, g1, h1

**Deploying Scripts:**

deploy/deploy\_mujoco/deploy\_mujoco.py - Sim2Sim testing in Mujoco

deploy/deploy\_real/deploy\_real.py - Sim2Real hardware deployment

deploy/deploy\_real/cpp\_g1/ - C++ implementation for G1 real-time deployment

**Configuration Files:**

deploy/deploy\_mujoco/configs/ - Mujoco deployment configs

deploy/deploy\_real/configs/ - Hardware deployment configs (g1.yaml, h1.yaml, h1\_2.yaml)

**Models:**

deploy/pre\_train/{robot}/motion.pt - Pre-trained models for each robot

logs/{experiment\_name}/exported/policies/ - Custom trained model exports

policy\_1.pt (MLP networks)

policy\_lstm\_1.pt (RNN networks)

**Paper I referenced:**

**Humanoid Locomotion and Manipulation: Current Progress**

**and Challenges in Control, Planning, and Learning**

**https://arxiv.org/abs/2501.02116**

**UC San Diego AMO work link**: https://amo-humanoid.github.io/