Policy Exploration and Quadruped Control with Live PPO

Samuel Burns, Maria Guerrero Cordoba, Rida Faraz

Advised by Professor Daniel Seita of the SLURM Lab

Introduction

When it comes to automating everyday tasks, robots have significant potential. Yet, as new innovations rapidly emerge in research, the adoption of this technology moves at a much slower pace. This is due to the limitations of existing robotic manipulators. Even simple tasks such as folding a cloth or tying a rope become challenging when given a stationary robot. As a result, we aim to work with Unitree Robotic's Go1/Go2 to explore the potential for quadruped robots as everyday manipulators. In addition, there are challenges with translating simulation to the real world, so we set out to explore how we can best train a policy in MuJoCo's physics-based simulation to ensure that the robot performs well when translating to the real world.

Background

Before we set out to train the Go2, we looked into various learning policies that have been utilized for quadruped training. Our initial literature review involved exploring three main policies: diffusion policy, action chunking with transformers, and MuJoCo Playground's PPO.

Diffusion Policy

Diffusion Policy learns by denoising instead of directly predicting actions. The model starts from random noise and learns how to refine it. This is done through many small updates until the final action emerges. Noise is added to the action and the model is trained to remove the noise until it learns how to reconstruct expert actions from randomness. Diffusion Policy allows our robot to learn and generate actions by gradually refining random noise into structured movement. This enables more flexible behavior generation, making Diffusion Policy highly effective for tasks with uncertainty or multiple valid solutions, where there is not a single "correct" answer and many precise actions can succeed.

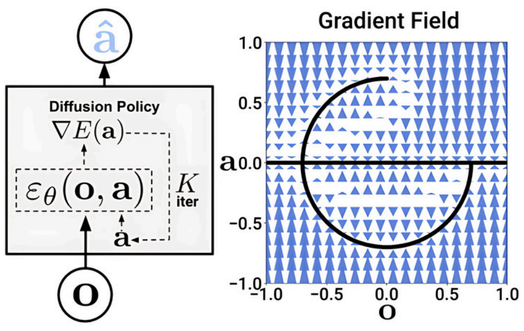


Fig. 1. Diffusion Policy Architecture

Action Chunking Transformers (ACT)

ACT is a policy architecture designed to generate a short sequence of actions (like "chunks") rather than deciding on a single action at each time step. It's useful for more structured behaviors such as standing up from a fall. The idea of chunking also helps reduce compounding errors. The key idea of ACT is to generate a small sequence of actions or actions that can be unrolled and executed over a short horizon, which can make behaviors smoother and more coherent. ACT can be trained with reinforcement learning or imitation to learn locomotion strategies like standing or walking. ACT could be used to generate a sequence that completes a task with fewer interruptions.

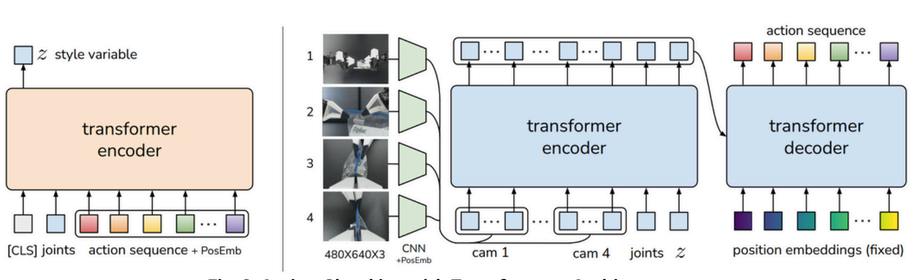


Fig. 2. Action Chunking with Transformers Architecture

Proximal Policy Optimization (PPO)

PPO is a reinforcement learning (RL) algorithm. Instead of learning from examples like a diffusion policy, PPO lets the robot interact with the environment and receive rewards based on its performance. Over time, the policy is updated to favor actions that lead to higher rewards, while keeping changes to the policy small and stable through a clipping mechanism. This balance between learning and stability helps prevent the robot from forgetting valuable behaviors. Here, PPO is used for locomotion, where the agent learns to balance by gradually improving its behavior through repeated interaction with the simulation.

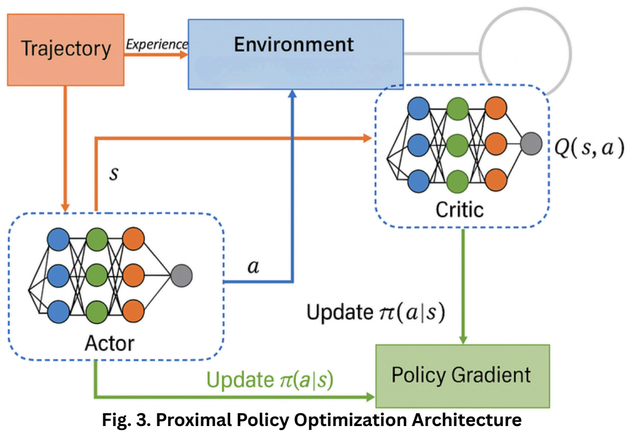
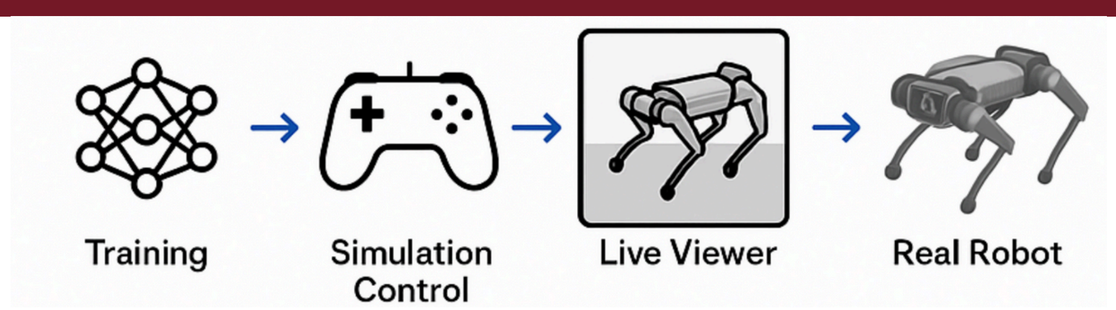


Fig. 3. Proximal Policy Optimization Architecture

Objective:



Live Joystick-Controlled Simulation

Goal & Motivation

- Enable live user-inputted control of the Unitree Go1 in simulation and physical hardware.
- Use Sim2Real transfer: begin with simulation, then later deploy the trained policy to the real robot.
- Challenge: MuJoCo Playground supports rollouts for training, but lacks an interactive viewer.
- *This project bridges that gap by:*
  - Implementing a live control loop using MuJoCo's passive viewer.
  - Enabling keyboard/controller input to drive a PPO-trained locomotion policy.
  - Establishing a pipeline compatible with real-world deployment.

Trained Policy

- Algorithm: PPO (Proximal Policy Optimization)
- Trained With: Brax (fast, JAX-based physics)
- Goal: Learn a policy mapping (observation, command) → actions
- Observation Includes:
  - Robot state (movement/orientation, joint angles, velocities, etc.)
  - Command: Desired [forward speed, lateral speed, yaw rate]

Command Injection

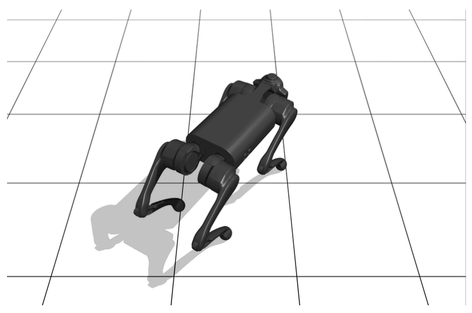
- Commands are stored in state.info["command"] every timestep.
- This command is then normalized, scaled, and injected into the observation vector
  - *Result: Policy sees a live goal and acts accordingly.*
- Keyboard/Controller inputs are converted into a velocity command vector:
  - $forward_{vel}$: forward/backward motion
  - $lateral_{vel}$: side-to-side motion
  - $turn_{rate}$: rotational (yaw) velocity
- These inputs are mapped to a normalized command vector:
  - [v_x, v_y, ω_z] representing desired motion in the robot's local frame (forward, sideways, turn).

Control Loop (Each Timestep)

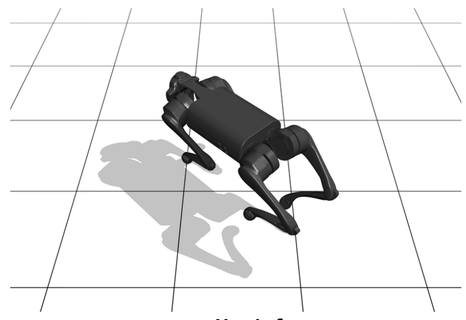
input → command → observation → policy → action → simulation

1. *Capture Input*
   - Keyboard or controller state is read and converted into a command.
2. *Update Environment Command*
   - Insert the command into state.info["command"].
3. *Recompute Observation*
   - Recalculate the observation using _get_obs() so the policy sees the new command.
4. *Infer Policy Action*
   - Feed the updated observation to the PPO policy (JAX, JIT-compiled for speed).
5. *Step Environment*
   - Advance simulation physics using the action (env.step()).
6. *Sync Viewer*
   - Apply control values to MuJoCo viewer and render the updated robot state.

Go1 in Action

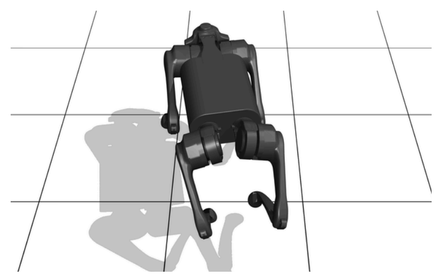


Yaw Right



Yaw Left

Walking Straight



45° Rightward Angeled Walking

Discussion

PPO is highly effective for training agents in reinforcement learning tasks, such as locomotion and control. With live implementation in MuJoCo, the next step is transitioning to real-world control via Sim2Real. Moving forward, we aim to explore ACT and Diffusion-based policies through implementation, which may be better suited for more specific and structured quadruped manipulation tasks, especially problems that involve precise coordination, longer-horizon planning, and multimodal tasks.

References

- Chi L., Zeng, A., Schaal, S., Tompson, J., Kalakrishnan, M., & Florence, P. (2023). Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. arXiv preprint arXiv:2303.04137.
- Zhao, T. Z., Kumar, V., Levine, S., & Finn, C. (2023). Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705.
- Zakka, K., Tabanpour, B., Liao, Q., Haiderbhai, M., Holt, S., Luo, J. Y., Allshire, A., Frey, E., Sreenath, K., Kahrs, L. A., Sferrazza, C., Tassa, Y., & Abbeel, P. (2024). MuJoCo Playground: A Scalable, Open-Source Framework for Robot Learning and Sim-to-Real Transfer. arXiv preprint arXiv:2502.08844.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Center for Undergraduate Research in Viterbi Engineering (CURVE)

SLURM Lab at USC

Sensing, Learning, and Understanding for Robotic Manipulation (SLURM)

USC Viterbi School of Engineering