Reinforcement Learning and Optimal Control for Autonomous Systems I
Project 2 - Reinforcement learning

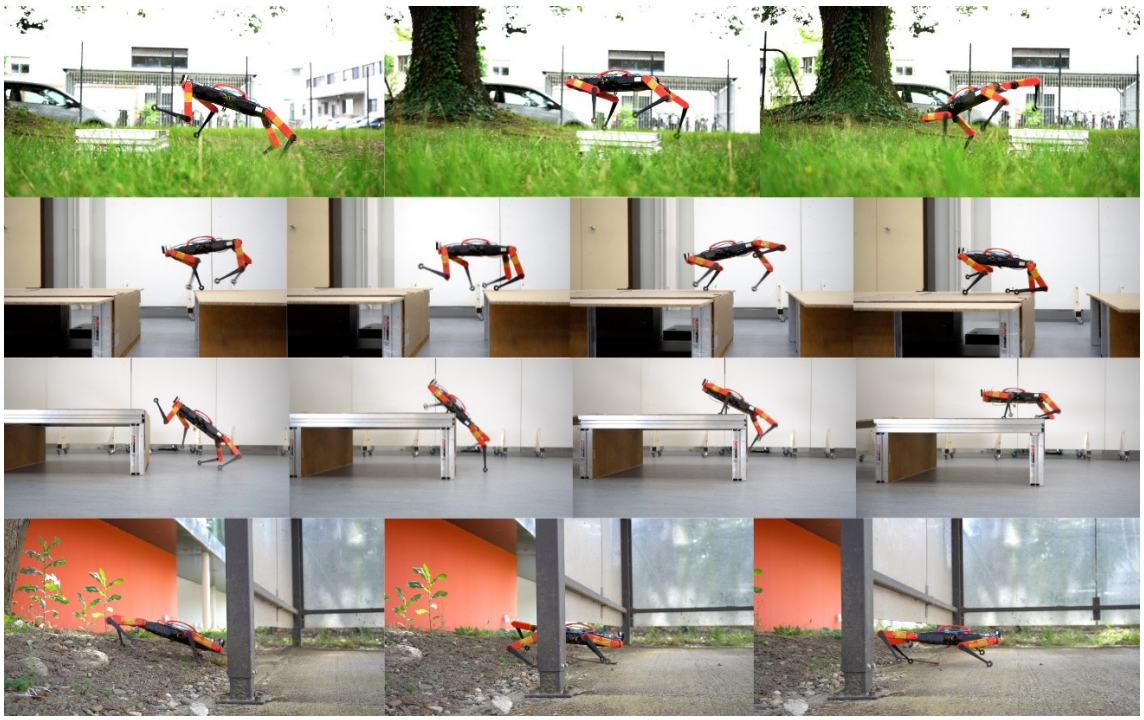

Figure 1: Examples of agile quadruped behaviors from SoloParkour [2], used here as high-level inspiration for gait quality and robustness.

Project Overview

In this project, you will train a Unitree Go2 walking policy in Isaac Lab, using PPO, starting from a deliberately weak baseline that only rewards linear and yaw velocity tracking, and your goal is to improve gait quality, stability, and command tracking through principled reward shaping, regularization, and robustness strategies (e.g., terrain randomness and disturbances). The policy should exhibit walking or trotting gaits with clean footfall timing, maintain a stable base (i.e. main body) approximately parallel to the ground at a reasonable height, act smoothly with regularized torques, and accurately follow three commands: forward/lateral velocities and yaw rate.

In order to get started, look at the project webpage, then follow the instruction of the Github repository to set the basic training environment on the HPC. The project tutorial is meant to help you start with improving the policy.

1. **Project webpage:** https://machines-in-motion.github.io/RL_class_go2_project/

2. **Github Repository:** https://github.com/machines-in-motion/rob6323_go2_project
3. **Project Tutorial:** https://github.com/machines-in-motion/rob6323_go2_project/blob/master/tutorial/tutorial.md

Deliverables

1. **Repository (fork of the provided baseline):** a cleanly organized codebase with your modifications to training configuration, reward terms, randomization, termination conditions, and any controller or observation/action-space adjustments clearly implemented. Include concise inline comments and a top-level `README.md` listing each major change, why it was made, and how to reproduce your results.
2. **Video:** a short demo (e.g., 20–60 seconds) of your best policy walking in Isaac Lab.
3. **Report (PDF, 2–4 pages):** a brief, structured write-up that explains the baseline limitations you observed, the modifications you introduced or tested (even when no/a bad effect was observed), why you introduced them, and their observed effects.

Grading (100 points total)

A. Policy Quality (60 pts)

- **Reproducing the tutorial part 1-4 (20pt):** implementing the basic gait improvements present in the provided tutorial.
- **Walking or trotting gait (20 pts):** clear, periodic footfall pattern with reasonable duty factor and symmetry for a walking or trotting gait (not pacing/pronounced hopping).
- **Base stability, attitude and height (5 pts):** low pitch/roll oscillations, base approximately parallel to the ground with a reasonable nominal height; avoids dragging the body or frequent knee/hip collisions.
- **Action regularization and smoothness (5 pts):** limited torque magnitudes (penalties on $\|\tau\|$, use very small reward scale such as -0.0001 or smaller) yielding visually smooth motions.
- **Command following (10 pts):** tracks $(v_x, v_y, \dot{\psi})$ commands with low steady-state error and no catastrophic failures when commands change slowly. A good linear velocity command tracking would bring `track_lin_vel_xy_exp` around 48 at the end of training in tensorboard, if you haven't modified the reward scale, and `track_ang_vel_z_exp` to 24. Also in the generated video, you can check that the green and blue arrow align correctly. One is the command, the other is the current direction of the robot.

B. Repository Quality (20 pts)

- **Code readability and structure (10 pts):** clear file organization, logical configs, and informative comments on key sections (reward, resets, observations/actions, etc).
- **Documentation of changes (10 pts):** top-level `README.md` summarizing each major addition, rationale, and reproduction steps (including exact command lines and seeds).

C. Report Quality (20 pts) The report should introduce the problem, explain design choices, methodology and metrics, analyze the results and provide a short discussion of the outcomes.

Bonus Tasks (up to +20 pts of extra credit)

1. **Actuator friction model with randomization: (5pts)** To transfer learned policies to real robot, it is necessary to create a more realistic simulation, especially add actuator friction dynamics. In this task, you need to add a simple actuator friction/viscous model in your environment code and randomize parameters per episode, following the spirit of [1]. This friction model combines static and viscous friction and should be subtracted from the torques computed by your low-level PD controller before they are sent to the robot.

   The friction torque is defined as:

$$\tau_{friction} = \tau_{stiction} + \tau_{viscous} \tag{1}$$

$$\tau_{stiction} = F_s \cdot \tanh\left(\frac{\dot{q}}{0.1}\right) \tag{2}$$

$$\tau_{viscous} = \mu_v \cdot \dot{q} \tag{3}$$

$$\tau_{PD} \leftarrow \tau_{PD} - \tau_{friction} \tag{4}$$

   where τ_{PD} is the output torque from the PD controller, $\dot{q}$ is the joint velocity, F_s is the stiction coefficient, and μ_v is the viscous coefficient. This forces the policy to overcome internal joint resistance, significantly narrowing the sim-to-real gap. This robustness is crucial for real-world deployment. **Note:** You should expect your total reward to decrease, as this makes the task harder. Add the following randomization logic to your episode reset (e.g., in the `_reset_idx` method): during reset for environment ID $\mathcal{E}$:

$$\mu_v^{\mathcal{E}} \sim \mathcal{U}(0.0, 0.3), \quad F_s^{\mathcal{E}} \sim \mathcal{U}(0.0, 2.5) \tag{5}$$

   Opportunities for real robot experiments The top 5 teams who implement the actuator friction model will have the opportunity to try their policies on the real Go2 robot (if the gaits are of sufficient quality to avoid damaging the robot). This will happen after the holiday break.

2. **Showcase a new skill in simulation: (15 pts)** You can either train the locomotion gait on uneven terrain (see IsaacLab example), create a bipedal walking task (see `https://arxiv.org/pdf/2509.00215v2` and `https://arxiv.org/abs/2311.05818` for inspiration), or create a controlled backflip with recovery. To get the maximum bonus point you only need to do 1 new task. Provide a reproducible training recipe and video.

Submission Checklist

- GitHub link to your forked repository with a reproducible training script/config and a `README.md`.
- A short MP4 video demonstrating nominal walking and command following.
- A PDF report (2–4 pages) with metrics, plots, and discussion.

References

[1] Joseph Amigo, Rooholla Khorrambakht, Elliot Chane-Sane, Ludovic Righetti, and Nicolas Mansard. First order model-based rl through decoupled backpropagation. In *Conference on Robot Learning (CoRL)*, 2025.

[2] Elliot Chane-Sane, Joseph Amigo, Thomas Flayols, Ludovic Righetti, and Nicolas Mansard. Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience. In *Conference on Robot Learning (CoRL)*, 2024.