*Course Project*

**Code of Honor.** All external resources used in the project, including research papers, open-source repositories, datasets, and any content or code generated using AI tools, e.g., ChatGPT, GitHub Copilot, Claude, Gemini, must be *clearly cited* in the final submission. The final report must also include *a clear breakdown of individual group member contributions.* Any lack of transparency in the use of external resources or in reporting group contributions will be considered academic dishonesty and will significantly impact the final evaluation.

| | |
|---|---|
| **Topic** | RL for Robotic Locomotion |
| **Category** | Robotics and Automation |

Objective Design and implement RL agents to train a simulated robot to perform locomotion tasks such as hopping or walking. The project should compare algorithms like PPO and SAC in terms of stability, sample efficiency, and robustness. The environment can be selected from lightweight simulators such as PyBullet (Hopper, Ant, or Walker).

Motivation Locomotion is a central challenge in robotics, requiring agents to coordinate multiple joints to achieve stable movement. RL has shown strong results in continuous control for locomotion [2], with algorithms like PPO [3] and SAC [1] widely used in modern research. This project introduces students to locomotion control problems in simulation, highlighting how RL can be used to train policies for dynamic robotic tasks. Students will also explore how environment modifications affect the robustness of locomotion policies.

Requirements The final submission should address the following requirements while the details can be freely decided by the group members.

1. Implementation: in this respect, you should
   - select a PyBullet locomotion environment (e.g., Hopper, Ant, Walker),
   - implement an RL agent (e.g., PPO [3], SAC [1]) for continuous control, and
   - implement a simple baseline (e.g., random policy or scripted controller) for comparison.
2. Environment modification: you can use a pre-implemented environment. To give the implementation some level of novelty, you **must** modify the environment with **at least one** of the following modifications:
   - adding Gaussian noise to joint positions or velocities,
   - modifying friction or mass parameters,
   - introducing delays in action execution,
   - adding distractor features to the state representation.
3. Evaluation: the final project should report key evaluation of the implemented algorithms in the modified environment. In this respect, the results should

- compare performance across different algorithms and baselines,
- evaluate stability, convergence speed, and robustness under environment modifications, and
- report locomotion metrics such as average distance traveled, reward per episode, and fall frequency.

4. The results should be elaborated through
   - ablation experiments, e.g., without entropy regularization or with different reward scaling, and
   - providing discussions on trade-offs between algorithm complexity, robustness, and performance.

MILESTONES The following milestones are to be accomplished through semester.

1. Literature Review and Setup
   - Review RL methods for locomotion control.
   - Choose environment and finalize environment modifications.
2. Implementation
   - Implement baseline controller.
   - Implement RL agents (PPO or SAC) and validate on unmodified environment.
   - Train agents on modified environments.
3. Evaluation and Analysis
   - Collect and plot locomotion performance metrics (distance traveled, reward per episode, fall frequency).
   - Compare across algorithms and modifications.
   - Perform ablation studies.
4. Final Report and Presentation

SUBMISSION GUIDELINES The main body of work is submitted through Git. In addition, each group submits a final paper and gives a presentation. In this respect, please follow these steps.

- Each group must maintain a Git repository, e.g., GitHub or GitLab, for the project. By the time of final submission, the repository should have
  - Well-documented codebase
  - Clear `README.md` with setup and usage instructions
  - A `requirements.txt` file listing all required packages or an `environment.yaml` file with a reproducible environment setup
  - Demo script or notebook showing sample input-output
  - *If applicable,* a `/doc` folder with extended documentation
- A final report (maximum *5 pages*) must be submitted in a PDF format. The report should be written in the provided formal style, including an abstract, introduction, method, experiments, results, and conclusion.
  **Important:** Submissions that do not use template are considered *incomplete.*
- A 5-minute presentation (maximum *5 slides including the title slide*) is given on the internal seminar on Week 14, i.e., *Dec 1 to Dec 5,* by the group. For presentation, any template can be used.

Final Notes While planning for the milestones please consider the following points.

1. Creativity in modifying the environment and designing reward functions is encouraged.
2. Training should remain feasible by limiting rollout length, reducing network sizes, and using shorter training horizons.
3. Students are encouraged to explore reward shaping or curriculum learning for locomotion stability.
4. Environment modifications should be realistic but lightweight to ensure experiments remain tractable.
5. Students are expected to balance training feasibility with achieving meaningful results.
6. Teams are expected to manage their computing needs and are advised to perform early tests to estimate runtime and training feasibility. As graduate students, team members can use facilities provided by the university, e.g., ECE Facility. Teams are expected to inform themselves about the limitations of the available computing resources and design accordingly.

## References


[1] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870. Pmlr, 2018.

[2] Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.