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	Controlling Robotic Arm with RL
Category	Robotics and Automation

Objective Design and implement an RL agent to control a robotic arm in a *reaching task*. The agent should learn to move the robot's end-effector to a target location in *continuous action space*. The project aims to compare RL algorithms such as PPO and SAC, and evaluate their robustness under environment modifications.

Motivation Robotic manipulation is a core problem in robotics, requiring precision, stability, and adaptability. RL provides a framework for training control policies directly from interaction with the environment, bypassing the need for explicit kinematic modeling [2]. This project introduces students to RL for robotic control, using lightweight simulated environments such as FetchReach (Gym Robotics) or Reacher (PyBullet). Students will gain experience with continuous control, sparse versus dense rewards, and robustness testing under noisy conditions.

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 simple robotic arm environment (e.g., FetchReach from Gym Robotics, or PyBullet Reacher),
   - implement an RL agent (e.g., PPO [3], SAC [1]) for continuous control, and
   - implement a simple heuristic baseline (e.g., proportional 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 end-effector observations,
   - introducing delays in action execution,
   - modifying reward sparsity (e.g., sparse vs dense),
   - adding distractor features to the state vector.

3. Evaluation: the final project should report key evaluation of the implemented algorithms in the modified environment. In this respect, the results should
    - compare the performance of different RL algorithms and baselines,
    - evaluate stability, convergence, and robustness under environment modifications, and
    - report success rate (target reached) and average distance to goal.
4. The results should be elaborated through
    - ablation experiments, e.g., removing entropy regularization, varying reward density, and
    - providing discussions on trade-offs between algorithm complexity and performance.

Milestones The following milestones are to be accomplished through semester.

1. Literature Review and Setup
    - Review RL for robotic arm control.
    - Choose environment and finalize environment modifications.
2. Implementation
    - Implement heuristic baseline.
    - Implement RL agent (PPO or SAC) and validate on unmodified environment.
    - Train agent on modified environment.
3. Evaluation and Analysis
    - Collect and plot performance metrics (success rate, learning curves).
    - Compare across algorithms and conditions.
    - 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 using small networks, limited episodes, and short rollouts.
3. Students are expected to balance training feasibility with achieving meaningful results.
4. 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] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

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