RealAIGym: Education and Research Platform for Studying Athletic Intelligence

Felix Wiebe*, Shubham Vyas*, Lasse Jenning Maywald*, Shivesh Kumar* and Frank Kirchner*†
* Robotics Innovation Center DFKI GmbH, Bremen Germany, email: firstname.lastname@dfki.de
† AG Robotics Department of Mathematics and Computer Science University of Bremen

I. Introduction

Traditional robots today (such as the ones used in factories) have a fixed base and are fully actuated under their operating conditions. However, modern robots inspired by animals are not bound to one place and are always underactuated. Like animals, these robots can perform dynamic movements, demonstrate compliance, and are robust to contact during their movements. The interest in dynamic robot behaviors has increased significantly due to the impressive athletic behaviors shown by robots developed by e.g. Boston Dynamics [3], MIT mini cheetah [17] and Agility Robotics [2]. This gives rise to the need for canonical robotic hardware setups for studying underactuation and comparing learning and control algorithms for their performance and robustness. These hardware setups and the accompanying software should be affordable, open and accessible. Similar to OpenAIGym [10] and Stable Baselines [20] which provide simulated benchmarking environments and baselines for Reinforcement Learning (RL) algorithms, the concept of RealAIGym (Real Athletic Intelligence Gym) is introduced for benchmarking dynamic behaviors on real robots. RealAIGym provides instructions for building reproducible robotic systems based on Quasi-Direct Drives as well as software to operate them for establishing a baseline for the application of dynamic control algorithms on real hardware.

II. RealAIGym

RealAIGym aims to provide an affordable and open-source robotics platform for studying athletic intelligence. The availability of Quasi-Direct Drives allow for high bandwidth torque control as well as standard position/velocity feedback control in robotic systems with low friction and high mechanical transparency. These drives were made popular with the MIT mini-cheetah robot [17] and are now easily available from multiple manufacturers such as *mjbots* [7] or *CubeMars* [4] along with open source firmware and CAN communication drivers. This allows to use a standardized actuation principle based on torque control in RealAIGym.

While there exist some platforms for robotics/controls education from commercial companies such as Quanser [8] and Acrome [1], recently the community has been moving to open-source platforms such as Open Dynamic Robot Initiative [12]. There are also standalone systems such as HOPPY [21], MIT mini-cheetah [5] and mjbots quad [6]. However, these do not yet fully reduce the entry barrier to education and research for dynamic robots (see Table I for a comparison). The following points are addressed by RealAIGym:

	Quanser	Acrome	Open Dynamic Robot Initiative	RealAIGym
Cost	🔴	🔴	🟡	🟡
Entry Barrier	🟢	🟢	🟡	🟡
API	MATLAB™	MATLAB™[1]	C++[2]	Python
Reproducibility	🔴	🟢	🟢	🟢
Open Source	🔴	🔴	🟢	🟢
Built-in Controllers	🔴	🔴	🟡	🟢
Benchmarking	🔴	🔴	🔴	🟢
Education	🟢	🟢	🟡	🟢

🟢 = good, 🟡 = partial, 🔴 = poor
[1] Interfaces exist for some plants.
[2] Python binding available.

TABLE I: Brief comparison between similar platforms.

Overall Cost: The currently available hardware platforms are either commercially sold or are open-source. The commercial platforms are more expensive than their open-source counterparts. Additionally, Quanser and Acrome both use MATLAB/Simulink as the programming language which adds an additional cost burden. RealAIGym embraces a complete open-source model for the motor firmware, communications, and controllers which makes the software free for anyone to use, modify, and improve without the need of any commercial licenses.

Entry Barrier and Programming Language: In order to ensure accessibility, we use Python 3 as the main programming language for RealAIGym. Components that require higher performance are written in C++ with Python bindings. We provide motor drivers in Python as well as a high level API for using the motors with custom designed controllers. We benchmarked closed loop control for a single motor in Python at a maximum of $\approx 1.5\,\text{kHz}$. Having multiple motors divides the control frequency accordingly. Using Python allows us to quickly prototype and test new control methods along with having a short onboarding time for new members. For example: A bachelors intern student needed less than 3 months for simulating a hopping and backflip on a 2-DoF hopping leg and then successfully demonstrated it on the real system.

Reproducibility: The reproducibility of experimental results is a central aspect of this project. Standardized hardware and a detailed documentation from the motor drivers to the controllers is supposed to make it possible for different working groups to obtain the same results.

Open Source: The transparency of open source software at all levels facilitates other researchers to confirm the implemented functionalities and allows students to learn about all aspects of robot control without barriers. The project also benefits from other open projects. For example the increase in availability of open-source Optimal Control (OC) ([19, 22, 15]) and Machine Learning (ML) ([11]) libraries reduces the access barrier to modern control methods. Using these allows the users to quickly synthesize and test new controllers on real systems to test their performance.

Built-in Controllers: A collection of built-in controllers shows the capabilities of the systems, provides examples for the APIs within the library and establishes the basis for reproducible results.

Benchmarking: Benchmarking controllers by comparing their performances is a scientific contribution of this project. Quantitative measurements of various control methods' advantages and disadvantages gives an objective characterisation of the controllers' capabilities.

Education: We consider a low entry barrier via the choice of programming language and API, a detailed documentation and free software are the cornerstones for a library to be suitable for education.

III. Gym Equipment

The intention of RealAIGym is to study dynamic control of underactuated systems. On one hand, the studied systems should be simple in order to be easily reproducible and accessible, whereas on the other hand the complexity of the control problems they offer should be non-trivial, so that qualitative and quantitative results can be obtained by testing, comparing and benchmarking various control methods. The current version of RealAIGym accommodates five robotic systems (see Fig. 1) of varying Degrees of Freedom (DoF):

Simple Pendulum: A torque-limited simple pendulum is arguably the simplest underactuated robotic system one could imagine. With only one motor, a link and a weight it can be reproduced easily. A natural control problem for the pendulum is to swing up from the hanging position to the upright position and stabilize the pendulum there with a torque limitation. The RealAIGym simple pendulum [24] can be found on github [9] and youtube https://www.youtube.com/watch?v=JVvwMGYiH3A.

Double Pendulum: A double pendulum is a simple 2-DoF system with chaotic dynamics. The system consists of two links and two motors at the joints. By disabling one of the motors and using it as a passive joint, the system can be operated either as an acrobot or as a pendubot system. Similar to the simple pendulum, a natural control problem is to swing up the double pendulum from the hanging position to the upright position and stabilize it there.

Hopping Leg: The hopping leg is a 3-DoF system with two motors and one passive vertical DoF. The leg is able to jump by pushing itself off from the ground. Interesting control problems are maintaining a constant jumping height and performing a backflip during the aerial phase.

Acromonk: An acromonk consists of two links which are joined together by a single motor. Two hooks at the external

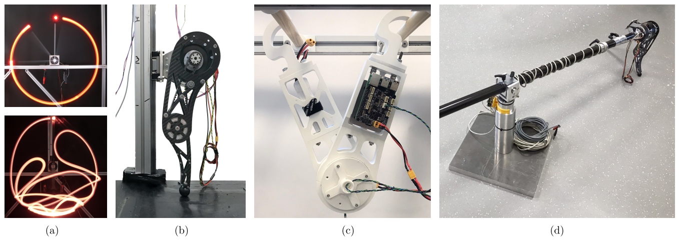
Fig. 1: The gym equipment currently available in the RealAIGym include: (a) simple and double pendulum, (b) hopping leg, (c) acromonk and, (d) boomstick hopper

ends of both links facilitate the acromonk to hold onto horizontal bars. As a free and wireless system the acromonk is capable of performing brachiation behavior similar to primates. The system is similar to an acrobot system from dynamics perspective but provides an additional control challenge in form of the brachiation task. The RealAIGym Acromonk is explained in more detail here [16].

Boomstick Hopper: The boomstick hopper is similar to the hopping leg but instead of the upright pole the hopper is attached to a boomstick which allows it to hop along a circle.

IV. Exercise Results

RealAIGym implements a variety of learning and control methods on the same system to achieve a certain task and benchmark the performances with full transparency. For example, in case of the simple pendulum, several control methods have been compared with respect to different criteria in simulation and and their proper functioning has been shown on the real hardware [24, 9]. The stabilization of the pendulum can be achieved with a Linear Quadratic Regulator (LQR). The LQR controller can be combined with an energy shaping controller which swings the pendulum up towards the upright position before applying the LQR stabilization. Alternatively, a swing up trajectory can be computed via trajectory optimization for which iterative LQR (iLQR) [23], direct collocation (dircol) [14] and Differential Dynamic Programming (DDP) [19] have been used. Trajectories found on this way can be executed on the system either by simply applying feed-forward torque (ff) or by stabilizing the trajectory with a PID controller or a Time-Varying LQR (TVLQR) controller. The iLQR optimization can also be processed online as part of the control loop resulting in a Model Predictive Control (MPC) controller. Lastly, two reinforcement Learning (RL) algorithms, Soft Actor Critic (SAC) [13] and Deep Deterministic Policy Gradient (DDPG) [18] have been tested for the comparison of optimal control and learning based control. The benchmark criteria are:

- Frequency: The inverse of the time the controller needs to process state input and return a control signal.
- Swing up time: The time it takes for the controller to swing up the pendulum.
- Energy consumption: The energy the controller uses during the swing up and stabilization.

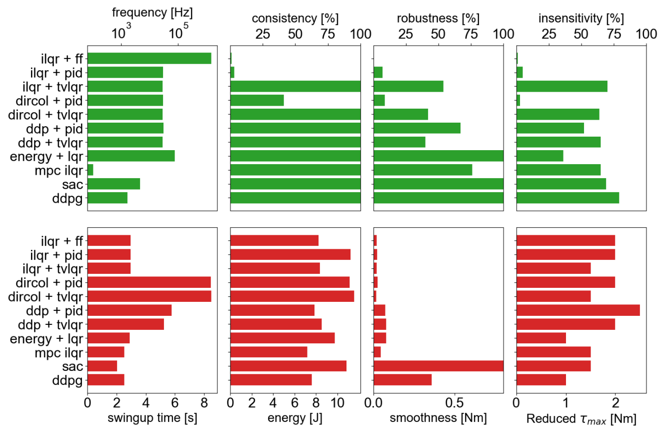

Fig. 2: Benchmark results for various control methods. Long green bars and short red bars indicate better results.

- Smoothness: Measures how much the controller changes the control output during execution.
- Consistency: Measures if the controller is successful for different starting states.
- Robustness: Tests the controller abilities to recover from perturbations during the swing up motions.
- Insensitivity: The pendulum model parameters (mass, length, friction, inertia) are modified without using this knowledge in the controller.
- Reduced torque limit: The minimal torque limit with which the controller is still able to swing up.

The benchmark results for the simple pendulum shown in Fig. 2 are the average or successful percentage of 100 repetitions for every controller and criterion. See [24] for more details on the results. Similar benchmark experiments including system specific criteria are planed for the other systems of the RealAIGym. The software and results for the other RealAIGym systems will be made open source in the near future.

References

[1] Acrome. https://acrome.net/. Accessed: 20-05-2022.

[2] Agility Robotics. https://agilityrobotics.com/robots. Accessed: 20-05-2022.

[3] Boston Dynamics. https://www.bostondynamics.com/. Accessed: 20-05-2022.

[4] CubeMars Motion Advanced Robotic System. https://cubemars.com/. Accessed: 20-05-2022.

[5] MIT mini Cheetah. https://github.com/mit-biomimetics/Cheetah-Software. Accessed: 20-05-2022.

[6] mjbots Quad A1. https://github.com/mjbots/quad, . Accessed: 20-05-2022.

[7] mjbots Robotic Systems. https://mjbots.com/, . Accessed: 20-05-2022.

[8] Quansar. https://quanser.com/. Accessed: 20-05-2022.

[9] RealAIGym Simple pendulum github. https://github.com/dfki-ric-underactuated-lab/torque_limited_simple_pendulum. Accessed: 08-06-2022.

[10] Greg Brockman, et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[11] TensorFlow Developers. Tensorflow. May 2022. doi: 10.5281/zenodo.6555127.

[12] F. Grimminger, et al. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020. doi: 10.1109/LRA.2020.2976639.

[13] Tuomas Haarnoja, et al. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, 2018.

[14] Charles R Hargraves and Stephen W Paris. Direct Trajectory Optimization Using Nonlinear Programming and Collocation. *Journal of guidance, control, and dynamics*, 10(4):338–342, 1987. doi: 10.2514/6.1986-2000.

[15] Taylor A Howell, et al. Altro: A fast solver for constrained trajectory optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679. IEEE, 2019.

[16] Mahdi Javadi. Mechatronic design and control of an underactuated brachiation robot. Master's thesis, Technical University of Kaiserslautern, Kaiserslautern, Germany, 2022.

[17] Benjamin Katz, et al. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[18] Timothy P. Lillicrap, et al. Continuous Control with Deep Reinforcement Learning, 2019.

[19] Carlos Mastalli, et al. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542, 2020. doi: 10.1109/ICRA40945.2020.9196673.

[20] Antonin Raffin, et al. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[21] Joao Ramos, et al. Hoppy: An open-source kit for education with dynamic legged robots. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4312–4318. IEEE, 2021.

[22] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

[23] Li Weiwei and Emanuel Todorov. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. *International Conference on Informatics in Control, Automation and Robotics*, pages 222–229, 2004. doi: 10.5220/0001143902220229.

[24] Felix Wiebe, et al. Torque-limited simple pendulum: A toolkit for getting familiar with control algorithms in underactuated robotics. *Journal of Open Source Software*, 7(74):3884, 2022. doi: 10.21105/joss.03884. URL https://doi.org/10.21105/joss.03884.