HasRL Robot: A Heterogeneous Asynchronous Reinforcement Learning System for High-Dimensional Bipedal Control

Jingyang Mai[†*], Zechen Guo[†*], Zhengding Luo[‡], Haozhe Ma[†§]
[†] National University of Singapore, Singapore
E-mails: maij@u.nus.edu, e1373526@u.nus.edu, haozhe.ma@u.nus.edu
[‡] Nanyang Technological University, Singapore
E-mail: LUOZ0021@e.ntu.edu.sg
[*] The authors contributed equally to this work.
[§] Corresponding author.

***Abstract*—Efficient utilization of computational resources remains a central challenge in reinforcement learning (RL)-based locomotion control, especially for high-dimensional bipedal robots. The robot learning process typically comprises two stages: data collection through environment interaction and policy optimization. Among these, the data collection phase is considered the primary bottleneck limiting overall learning speed. To address these challenges, we present HasRL Robot, a heterogeneous asynchronous RL-based learning system for bipedal locomotion. In this framework, trajectories are collected in parallel across CPUs, while the policy and value networks are updated asynchronously on the GPUs. This design enables effective use of computational resources and improves overall training throughput. Furthermore, our framework is extensible to other RL-driven tasks on high-performance machines, thereby maximizing computational efficiency. Experimental results demonstrate that HasRL Robot achieves superior training and testing performance, enabling the emergence of stable and human-like gait behaviors in simulation.**

I. Introduction

Bipedal robotics shows promising research prospects as an interdisciplinary subject of robotics and human coordinated movement, as evidenced by increasing research efforts. Cassie robot is a dynamic bipedal robot developed by Agility Robotics in 2016, designed to walk and run with human-like agility. It features 10 degrees of freedom in its legs, enabling movement across diverse terrains, which is a practical but complicated robot model to study robot control.

Research shows that reinforcement learning (RL) methods offer effective solutions for bipedal robot locomotion control, such as training the robot to imitate walking with prior knowledge of human gaits and to overcome complex physical terrain, e.g., stairs. Xie et al. [1] applied Proximal Policy Optimization (PPO) [2] to train a feedback controller for the Cassie bipedal robot using a realistic simulation model. Their method formulates locomotion as an imitation-based RL problem, where the policy learns to track a reference motion while adapting to real-world physical constraints such as torque limits and joint ranges. The learned controllers demonstrate strong robustness, maintaining stable walking even under sensory delays, random pushes, and blind walking over uneven terrain. This shows that deep RL can effectively exploit full robot dynamics to produce adaptive, human-like gaits in challenging environments.

To move beyond simulated robustness and enable real-world deployment, Li et al. [3] proposed a reinforcement learning framework for bipedal locomotion that combines a Hybrid Zero Dynamics-based gait library with model-free deep reinforcement learning. The gait library provides a diverse set of parameterized, periodic walking motions, which serve as reference trajectories during training. The resulting controller tracks commands for walking speed, height, and turning rate. To ensure generalization, the authors applied domain randomization over dynamics, sensor noise, and control delays. The learned policy transferred directly to the real Cassie robot without additional fine-tuning, and demonstrated robustness to motor failures, uneven terrain, and external disturbances. This work highlights how diverse, randomized simulation training—anchored by a structured gait library—can effectively bridge the sim-to-real gap. Similarly, Siekmann et al. [4] demonstrated successful sim-to-real transfer of blind stair traversal on the Cassie bipedal robot by relying solely on proprioceptive feedback. The authors achieved this by introducing terrain and dynamics randomization in simulation—varying stair height, depth, slope, and physical parameters like mass and friction—without modifying the reward function used for flat-ground walking. A memory-enabled LSTM policy is trained with PPO and learned robust foot placement and recovery behaviors. The resulting policy generalized effectively to real-world stairs and uneven terrain, highlighting that diverse simulation exposure alone is sufficient to enable real-world deployment, even without exteroceptive sensing.

Despite recent progress, a key limitation persists: long training times and high computational costs, particularly for high-dimensional robots like Cassie. This challenge underscores the need for more efficient and scalable reinforcement learning frameworks. A notable example is the Asynchronous Advantage Actor-Critic (A3C) algorithm proposed by Mnih et al. [5], which employs multiple worker agents to collect

979-8-3315-7206-8/25/$31.00

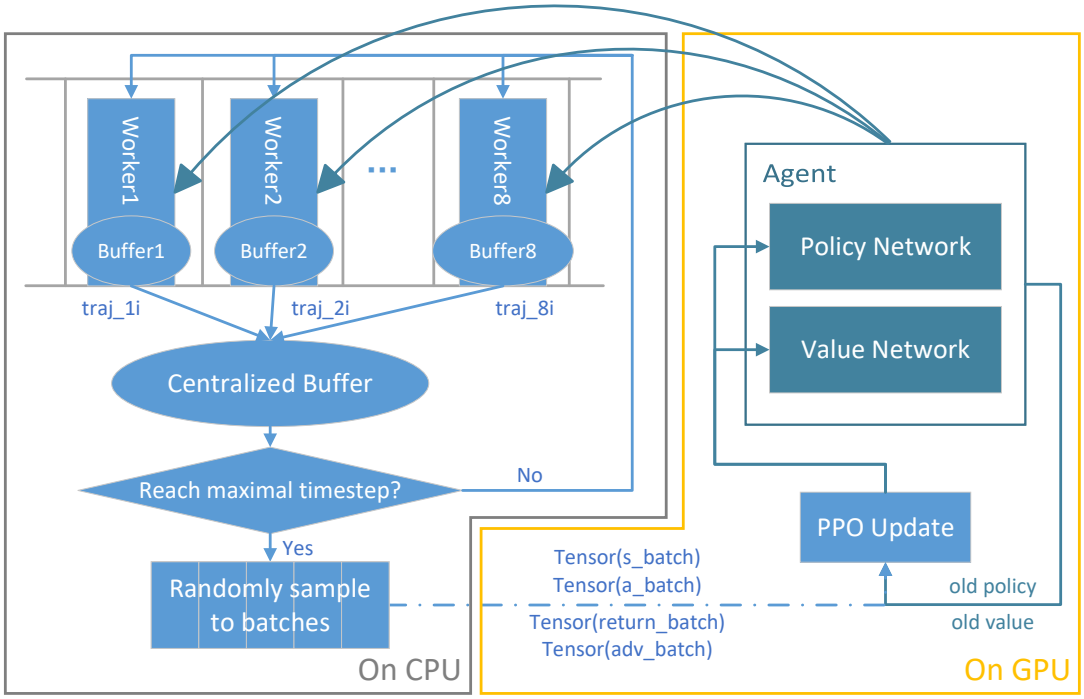


Fig. 1: **Overview of HasRL Robot in Each Iteration.** It consists of parallel collecting on the CPU and asynchronous training on the GPU. In each iteration, a centralized buffer on the CPU will be filled with trajectories collected from multiple parallel simulation workers until a predefined maximal timestep is hit. In this figure, eight workers are used, each running a single agent that may gather one or more trajectories as needed. Trajectories will then be reorganized into batches on the GPU, used for policy and value network updates. In the next iteration, agents in all workers utilize the most recently updated policy to take actions.

experience in parallel across independent environments while asynchronously updating shared networks. This design reduces policy gradient variance, enhances exploration diversity, and significantly accelerates training.

Motivated by the success of asynchronous frameworks like A3C, the main contribution of our paper is a **h**eterogeneous, **as**ynchronous **r**einforcement **l**earning-based robot system (HasRL Robot[1]) for locomotion control, a novel framework that collects trajectories in parallel on the CPU and updates policy and value network asynchronously on the GPU. We evaluate HasRL Robot on a high-dimensional bipedal robot, Cassie, in simulation. We show that our framework enables stronger capability of model convergence and more efficient computing consumption. We also show that the framework enables the Cassie robot to achieve robust walking in the human gait pattern.

The remainder of this paper is organized as follows. Section II introduces fundamental components of our RL formulation. Section III describes the architecture of HasRL Robot, detailing the heterogeneous processes, trajectory collection across CPUs and training on the GPU, as well as the asynchronous network update implementation. Section IV presents experimental results, evaluating learning performance and simulated locomotion performance. Finally, Section V concludes our contributions, discusses limitations, and highlights directions for future work.

[1]The source code is accessible at: https://github.com/whitbrunn/HasRLRobot

II. Reinforcement Learning Formulation

A. State Space

The state space is a 50-dimensional vector comprising the robot internal observations and externally provided motion commands. It consists of two main components: (i) the robot state, encoding proprioceptive information such as joint positions, velocities, and sensor readings; and (ii) the command input, specifying target motion directives.

The robot state observation includes the pelvis height and spatial orientation, represented as a unit quaternion, as well as the pelvis's linear velocity, angular velocity, and partial linear acceleration measurements. The positions and velocities of ten actuated joints are provided, corresponding to the motorized degrees of freedom in the hips, knees, and feet on both legs: left/right hip roll (Motor $0/5$), yaw (Motor $1/6$), pitch (Motor $2/7$), left/right knee (Motor $3/8$), and left/right foot (Motor $4/9$). In addition, six passive joint states are included, which cover the positions and velocities of the shin, tarsus, and foot joints: left/right shin (Joint $0/3$), left/right tarsus (Joint $1/4$), and left/right foot (Joint $2/5$).

The external input consists of two command variables encoding the target walking speeds in the sagittal and lateral directions. To capture the temporal structure of locomotion, a

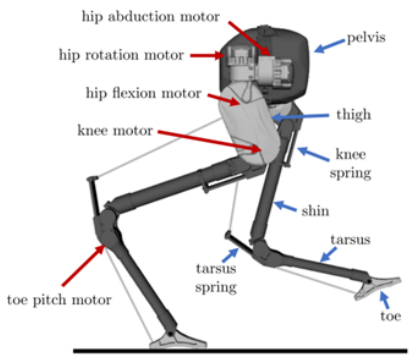


Fig. 2: **Joint and actuator configuration of the Cassie bipedal robot.** Red arrows indicate motor positions actuating hip abduction, hip rotation, hip flexion, knee, and toe pitch. Blue labels denote key passive joints, including the shin, tarsus, and springs. This mechanical arrangement defines the actuated and passive joint groups represented in the observation state. *Figure adapted from [6].*

two-dimensional phase clock signal is added, computed as the sine and cosine of a normalized gait phase. This signal provides a periodic reference reflecting the progression through the gait cycle and supports synchronization with foot contact patterns.

B. Action Space

The action space is defined as a 10-dimensional vector, where each element specifies the target position for one of actuated joints of Cassie robot. These targets are interpreted as the desired setpoints for a low-level PD controller, which subsequently computes the joint torques applied to the robot. This formulation enables stable and smooth control by decoupling high-level policy outputs from direct torque commands, leveraging the inherent robustness of PD-based motor control.

C. Reward Function

The reward function is designed to encourage stable and gait-synchronized walking while penalizing unstable motions, inefficient torque usage, and abrupt control signals. It integrates multiple physically meaningful terms, each weighted to balance locomotion quality and control smoothness.

Stability is primarily evaluated through pelvis-related components, since the pelvis reflects overall posture and balance. One term penalizes deviations from a nominal forward-facing orientation, another penalizes velocity tracking errors, and additional penalties address lateral displacement, height variation, and high accelerations to suppress instability. To enforce gait synchronization, foot force and velocity terms are modulated by sinusoidal clock signals aligned with the locomotion phase, rewarding timely contact and lift-off. These signals are normalized and passed through a tangent mapping to emphasize correct timing and penalize mistimed contacts, effectively enhancing the temporal sensitivity of the policy [7]–[12].

Formally, the reward is expressed as

$$\begin{aligned} r_t = s\Big(&w_1 S_F + w_2 S_V + w_3 e^{-(E_{\text{pel}}+E_{\text{foot}})} \\ &+ w_4 e^{-P_m} + w_5 e^{-E_v} + w_6 e^{-H_p} \\ &+ w_7 e^{-T_p} + w_8 e^{-A_p}\Big) \end{aligned}$$

where s is a global scaling factor and $w_1, \ldots, w_8$ are fixed coefficients. The symbols are summarized in Table I.

TABLE I: The definition of reward terms.

Symbol	Meaning
S_F	Clock-shaped normalized foot–force score
S_V	Clock-shaped normalized foot–velocity score
E_{pel}	Pelvis orientation error
E_{foot}	Foot orientation error
P_m	Pelvis motion
E_v	COM forward velocity tracking error
H_p	Hip–roll angular velocity penalty
T_p	Torque change penalty
A_p	Action change penalty

Throughout training, the agent collects sequences of interactions, which we denote as trajectory τ. Each timestep of each trajectory consists of four components, a state, an action, a corresponding reward, and a next state. Trajectories that characterize the behavioral evolution of the agent over time and serve as the basic units for policy optimization. In our paper, we define trajectory τ w.r.t. timestep $t = 0 \sim T$ as:

$$\begin{aligned} \tau &= \{s_0, a_0, r_0, s'_0, \ldots, s_T, a_T, r_T, s'_T\} \\ \tau_a &= \{a_0, a_1, \ldots, a_T\} \\ \text{or} \quad \tau_s &= \{s_0, s_1, \ldots, s_T\} \\ \tau_r &= \{r_0, r_1, \ldots, r_T\} \\ \tau_{s'} &= \{s'_0, s'_1, \ldots, s'_T\} \end{aligned} \tag{1}$$

where $s'_t = s_{t+1}$ $(t = 0, 1, \ldots, T)$, T is the terminal timestep that may vary with each trajectory.

D. Dynamics Randomization

To enhance the robustness and sim-to-real transferability of the learned locomotion policy, we apply dynamics randomization during training. At the start of each episode, physical parameters such as link masses, joint damping, and ground friction are randomly perturbed within predefined ranges. Additionally, encoder noise is injected into motor and joint readings, and slight variations are applied to commanded speeds and simulation rate. This encourages the policy to perform reliably under diverse and uncertain conditions.

E. Policy Representation and Learning

Siekmann et al. [4] demonstrate that memory is a critical component for robust locomotion control of the Cassie robot on stair-like terrain, providing strong empirical evidence for its effectiveness. Motivated by these findings, we represent the locomotion control policy using a long short-term memory (LSTM) recurrent neural network [13], which consists of two recurrent hidden layers. As for the value function, we define a

three-layer MLP network, using tanh as the activation function for each hidden layer. For both policy and value networks, the dimensionality of each hidden layer is configurable via hyperparameters, with a default value of 128.

It is important to note that the inputs to both the policy and value networks are sequences of states. However, the size of these inputs differs between the collecting and training stages. During the collecting stage on the CPU, each network processes one state corresponding to one timestep at a time until the end of the state trajectory. Thus, the input to the network has a shape of [1 × 50]. In contrast, during the training stage on the GPU, trajectories of states corresponding to sequences of timesteps from the centralized buffer are reorganized, resulting in an input shape of [batch size × trajectory length × 50]. To support this variability, we introduce a scalable mechanism in both the policy and value networks: (1) if the input has two dimensions (i.e., $[1 \times 50]$), an additional dimension is prepended to match the training input format; (2) regardless of whether the input has two or three dimensions, the output shape is preserved to match the corresponding input shape.

For training of the policy, we use PPO. The PPO loss is decomposed into two components, actor loss and critic loss, corresponding to the policy and value networks, each optimized separately. The actor loss is designed to improve the policy by maximizing a clipped surrogate objective, which stabilizes updates by preventing large policy shifts, expressed as:

$$\begin{aligned}
\mathcal{L}_{\text{Actor}} &= -\mathcal{J}_{\text{CLIP}}(\theta) - c_1 H(\theta) \\
r_t(\theta) &= \exp\left(\log\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\right)\right) \\
\mathcal{J}_{\text{CLIP}}(\theta) &= \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon,\ 1+\epsilon)\ \hat{A}_t\right)\right] \\
H(\theta) &= -\mathbb{E}_{s_t}\left[\sum_a \pi_\theta(a|s_t)\log\left(\pi_\theta(a|s_t)\right)\right]
\end{aligned}$$

where $\hat{A}_t$ is the advantage estimate and c_1 is the entropy coefficient. Especially, we define the advantage function by adopting a simpler off-policy approach[14] instead of using Generalized Advantage Estimation (GAE), where the advantage is calculated as the difference between the empirical return, the sum of discounted rewards along each trajectory, and the value prediction, the output of the value network:

$$\hat{A}_t = R_t - V(s_t) \tag{2}$$

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t-1} r_{T-1} \tag{3}$$

The critic loss minimizes the mean squared error between the predicted state value $V(s_t)$ and the empirical return R_t:

$$\begin{aligned}
\mathcal{L}_{\text{Critic}} &= c_2\, \mathcal{L}_{\text{VF}} \\
\mathcal{L}_{\text{VF}} &= \frac{1}{2}\,\hat{\mathbb{E}}_t\left[\left(V(s_t) - R_t\right)^2\right]
\end{aligned}$$

where c_2 is a hyperparameter balancing the value loss.

III. HASRL ROBOT

We provide an overview of our method in Figure 1. It combines parallel trajectory collection on the CPU with asynchronous policy training on the GPU. During each iteration, a centralized buffer accumulates trajectories from multiple simulation workers running in parallel until a preset timestep threshold is reached. Once the buffer is full, the collected data is transferred to the GPU, where it is organized into batches for updating the policy and value networks. Subsequently, in the next iteration, all agents act based on the same recently updated policy parameters.

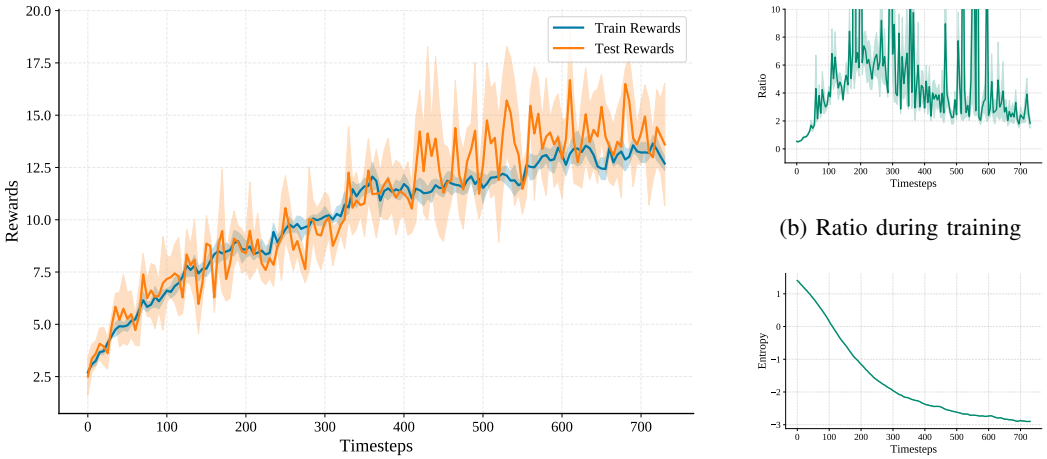
A. Parallel Collecting on CPU

Each parallel worker on the CPU is independently simulated using MuJoCo [15]. It consists of three components: (1) an OpenAI Gym [16] environment that includes a Cassie robot and flat ground, (2) an actor-critic agent corresponding to the policy and value networks, and (3) a buffer to store trajectories. In each iteration, the predefined maximum number of timesteps is divided by the number of workers to assign a timestep budget to each worker (i.e., to each agent). At each timestep, the buffer of each worker stores a tuple consisting of a state, an action, a reward, and the subsequent state as part of a trajectory, until the agent reaches either its maximum allotted timesteps or a terminal state, as defined in Equation 1. Note that in the same timestep during implementation, a value predicted by the value network and an action logit derived from the policy network output are also stored as part of the trajectory, to be used in subsequent calculations. In addition, the buffer records the start and end indices of each complete trajectory and it may store multiple trajectories one after another per iteration.

Once the total number of timesteps across all workers reaches the predefined maximum, the return and advantage corresponding to each timestep of every trajectory are calculated using Equation 2 and 3, respectively. With each trajectory extended to include the computed return and advantage values, all trajectories are then stored sequentially in a centralized buffer, which shares the same structure as the buffers used by individual workers. However, during the storing process, the start and end indices of each trajectory are adjusted to reflect their new positions in the centralized buffer. These indices are later used during the training stage to reorganize trajectories into batches on the GPU.

B. Asynchronous Training on GPU

In each iteration, trajectory sequences in the centralized buffer regarding states, action logits, returns, and advantages are segmented according to individual trajectory lengths. A random batch sampler collects trajectory segments from CPUs and assembles them into fixed-size batches on the GPU, with zero-padding applied as needed on the right. Given the input state batch, the latest policy and value network compute both a value batch and a new action logit batch. All batches are then used to independently compute the PPO actor and critic losses, which in turn update the policy and value networks.



(a) Train/Test rewards

(b) Ratio during training

(c) Entropy during training

Fig. 3: **PPO training statistics.** Visualization of key metrics during PPO: (a) episode rewards for both training and testing phases, (b) policy ratio indicating update stability, and (c) entropy tracking exploration dynamics.

IV. Experiments

Our experiments are conducted in the MuJoCo-based Cassie simulation environment, interfaced via the OpenAI Gym API. The agent is trained using PPO algorithm, and its performance is evaluated. The experiments are designed to examine two aspects: the learning performance of the control architecture and the robustness analysis in simulation.

A. Learning Performance

Throughout training, the learned policy achieves progressively higher performance, with the average training return rising from below 5 to over 12.5, eventually stabilizing in the later stages. The evaluation return reaches a peak around 14 and remains within a narrow band thereafter, suggesting that the policy generalizes well beyond the training trajectories. The close alignment between training and evaluation curves indicates that the policy is neither underfitted nor overfitted, and is able to consistently produce stable walking behavior across diverse initializations and environment conditions.

Entropy decay during training provides additional insight into the policy's evolving behavior. In the early stages, higher entropy encourages policy exploration over diverse locomotion strategies, while in later stages, the entropy decreases as the agent becomes more confident in its action selections. The gradual reduction in entropy reflects a smooth transition from early-stage exploration to more confident and deterministic action selection in later stages, as the policy becomes increasingly certain about the appropriate actions to take in different locomotion scenarios.

The PPO clipping ratio reflects how closely the new policy remains within the trust region of the old policy. Early in training, the mean ratio exceeds 10 due to large policy updates, but gradually converging to a tighter band between 1.0 and 2.0. This behavior indicates that the clipping mechanism effectively limits policy updates and prevents instability. The early fluctuations align with the phase when the agent is rapidly improving and exploring the policy space.

B. Hyperparameter Tuning Analysis

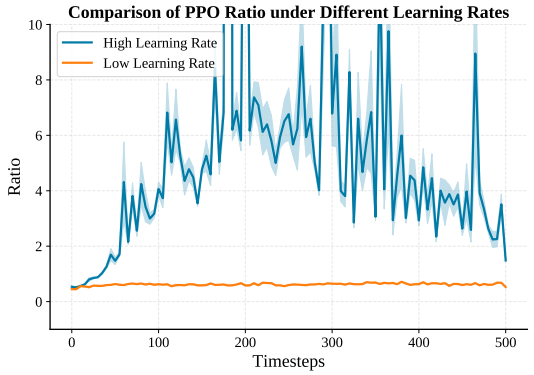
During the training process, we observes that the *Ratio*—the mean probability ratio between new and old policies—has a significant impact on policy improvement. As shown in Fig. 4, when the ratio remains close to 0.6, the corresponding training rewards fluctuate heavily and failed to show meaningful growth. In contrast, training runs where the ratio consistently exceeded 1.0 demonstrated steady increases in return, indicating more effective policy updates.

To promote larger ratios, we increase the learning rate for the policy network. This adjustment amplifies the update step in gradient ascent, thereby increasing the likelihood of the ratio exceeding 1.0. However, we also observe that overly large learning rates may lead to instability or premature convergence. Thus, a careful balance must be maintained between update magnitude and policy entropy to ensure both learning speed and long-term performance.

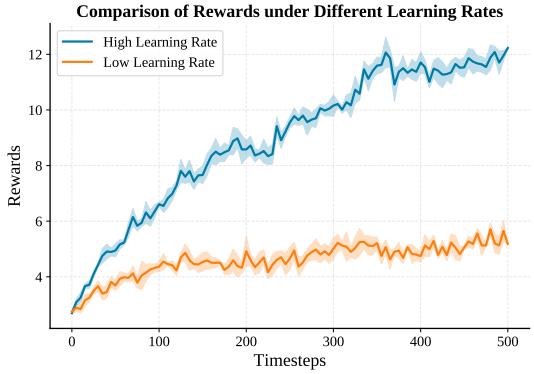
This observation aligns with the PPO objective, where overly conservative updates (i.e., ratios close to 1) result in minor policy changes, while moderately larger ratios—still within the clipping bound—encourage more expressive learning.

V. Conclusion and Future Works

In this work, we introduce HasRL Robot, a heterogeneous asynchronous reinforcement learning framework developed to facilitate robust policy learning for bipedal locomotion. By parallelizing trajectory collection across CPUs and performing asynchronous updates of policy and value networks on the GPU, our system achieves significantly improved computational efficiency and learning throughput. Additionally, experimental results demonstrate that HasRL Robot exhibits fast and stable model convergence, resulting in robust, human-like walking behaviors in simulation. Our findings highlight



(a) Ratio with different learning rates



(b) Rewards with different learning rates

Fig. 4: **PPO performance under different learning rates.** (a) Policy ratio reflects the magnitude of policy updates; (b) training rewards indicate learning stability and effectiveness.

the importance of addressing the data collection bottleneck in high-dimensional RL for locomotion control and show that heterogeneous hardware utilization can lead to substantial gains in learning speed.

In future work, training hyperparameters will be further optimized to improve convergence speed and enhance the quality of the final policy, as reflected in the value of the converged reward. Since reinforcement learning for bipedal locomotion is highly sensitive to the design of the reward function, optimizing the reward formulation will be a critical direction for further improving learning performance. Another key direction will be decoupling the data collection and training stages into parallel processes with independent iterations. This will involve synchronizing training frequency with data generation to prevent overfitting on outdated experiences.

References

[1] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. Van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1241–1246.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[3] Z. Li, X. Cheng, X. Peng, *et al.*, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," May 2021, pp. 2811–2817.

[4] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Proceedings of Robotics: Science and Systems (RSS)*, 2021.

[5] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," *arXiv preprint arXiv:1602.01783*, 2016.

[6] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, "Rapid trajectory optimization using c-frost with illustration on a cassie-series dynamic walking biped," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 4722–4729.

[7] H. Ma, F. Li, J. Y. Lim, Z. Luo, T. V. Vo, and T.-Y. Leong, "Catching two birds with one stone: Reward shaping with dual random networks for balancing exploration and exploitation," in *Forty-second International Conference on Machine Learning*, 2025.

[8] H. Ma, Z. Luo, T. V. Vo, K. Sima, and T.-Y. Leong, *Centralized reward agent for knowledge sharing and transfer in multi-task reinforcement learning*, 2025. arXiv: 2408.10858.

[9] H. Ma, K. Sima, T. V. Vo, D. Fu, and T.-Y. Leong, "Reward shaping for reinforcement learning with an assistant reward agent," in *Forty-first International Conference on Machine Learning*, 2024.

[10] Z. Luo, H. Ma, D. Shi, and W.-S. Gan, "Gfanc-rl: Reinforcement learning-based generative fixed-filter active noise control," *Neural Networks*, vol. 180, p. 106 687, 2024.

[11] H. Ma, T. V. Vo, and T.-Y. Leong, "Mixed-initiative bayesian sub-goal optimization in hierarchical reinforcement learning," in *Proceedings of the 23rd international conference on autonomous agents and multiagent systems*, 2024, pp. 1328–1336.

[12] H. Ma, T. V. Vo, and T.-Y. Leong, "Hierarchical reinforcement learning with human-ai collaborative subgoals optimization," in *Proceedings of the 22nd international conference on autonomous agents and multiagent systems*, 2023, pp. 2310–2312.

[13] C. Olah, *Understanding LSTM networks*, 2015.

[14] D. Silver, *Lecture 5: Model-free control*, Part of *Lectures on Reinforcement Learning*, 2015.

[15] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2012, pp. 5026–5033.

[16] G. Brockman, V. Cheung, L. Pettersson, *et al.*, *Openai gym*, 2016. eprint: arXiv:1606.01540.