Legged locomotion Control for Low-cost Quadruped Robots Using reinforcement learning

Rajesh Tiwari, Shailesh Khapre and Avantika Singh

Department of Data Science & Artificial Intelligence,
Dr. S. P. Mukherjee International Institute of
Information Technology, Naya Raipur
Chhattisgarh, India
rajesht@iiitnr.edu.in , shailesh@iiitnr.edu.in, avantika@iiitnr.edu.in

Abstract-

Low-cost quadruped robots offer an affordable platform for research and education, yet achieving reliable locomotion remains challenging due to servo-driven actuation, sensing noise, and mechanical uncertainty. Although reinforcement learning has shown strong performance on high-end quadrupeds, its effectiveness on low-cost platforms is not well understood. This paper presents a simulation-based comparative study of Augmented Random Search (ARS), Soft Actor-Critic (SAC), and Proximal Policy Optimization (PPO) for locomotion control on the low-cost SpotMicro quadruped robot. A realistic physics-based simulation environment is developed in PyBullet, incorporating servo dynamics, a Bézier-based gait phase generator, and extensive domain randomization. All algorithms are trained and evaluated under identical conditions across varied terrain and sensing scenarios. Results show that ARS achieves fast convergence and the highest average return, while SAC provides smoother, more stable, and more robust locomotion, particularly on uneven terrain and under sensing uncertainty. PPO exhibits lower stability and higher energy consumption, highlighting the importance of algorithm selection for low-cost quadruped locomotion

Keywords-

locomotion, reinforcement learning, quadruped robot, pybullet, bezier trajectory, ars, sac, ppo

1 Introduction

Quadruped robots have attracted significant interest due to their ability to traverse complex and unstructured environments that are inaccessible to wheeled or tracked systems. Recent progress in reinforcement learning has enabled highly dynamic and agile locomotion on quadruped platforms, particularly on robots equipped with torque-controlled actuators and high-fidelity sensing [1], [2]. However, these advances largely target expensive research platforms and do not directly translate to low-cost quadruped robots that rely on servo driven actuation, limited sensing, and mechanically compliant structures.

Low-cost quadruped robots such as SpotMicro represent an important class of platforms for education, rapid prototyping, and applied research [3]. Their affordability and open-source nature make them widely accessible, but these advantages come at the cost of significant hardware limitations. Servo motors introduce actuation delay, backlash, and nonlinear response characteristics, while low grade inertial sensors and the absence of force feedback increase state uncertainty [3], [4]. These factors make stable and efficient locomotion difficult to achieve using traditional model-based control approaches, which typically require accurate dynamics models and careful manual tuning.

Reinforcement learning (RL) offers a promising alternative by enabling control policies to be learned directly through interaction with the environment, allowing the controller to adapt to modelling errors and unmodeled dynamics [2], [5]. Nevertheless, the choice of RL algorithm plays a critical role in determining whether learned policies are stable, robust, and suitable for real world deployment. Existing studies often evaluate reinforcement learning methods in isolation, under different experimental conditions, or on high end hardware, making it difficult to draw meaningful conclusions about their relative performance on low-cost quadrupeds [6].

This work addresses this gap by presenting a unified and controlled comparison of three representative reinforcement learning algorithms for quadruped locomotion. Augmented Random Search (ARS) represents a derivative free and computationally efficient approach [7]. Soft Actor Critic (SAC) provides a stochastic actor critic framework with entropy regularization that promotes smooth and robust behavior [8]. Proximal Policy Optimization (PPO) is a widely

adopted on policy method that serves as a strong baseline in many robotic control tasks [9].

To ensure relevance to real world deployment, a physics-based simulation environment is developed in PyBullet that incorporates servo specific actuation modeling, sensing noise, and extensive domain randomization across physical parameters and terrain conditions [4]. A Bezier based gait phase generator is integrated to provide structured locomotion priors while preserving the flexibility of learning-based control [1]. Performance is evaluated across multiple terrains and sensing configurations using consistent metrics including learning convergence, stability, smoothness, energy efficiency, foot-slip, and fall rate.

The contributions of this paper are threefold. First, it provides a systematic and controlled benchmarking study of reinforcement learning algorithms for locomotion on a low-cost quadruped robot under realistic hardware constraints. Second, it demonstrates how algorithmic design choices influence stability, robustness, and energy efficiency under realistic hardware constraints [6]. Third, it offers practical insights into selecting reinforcement learning methods for deploying locomotion controllers on affordable quadruped platforms.

2 Related Work

Research on quadruped locomotion has traditionally relied on model-based control techniques, including central pattern generators, inverse kinematics, and optimization-based trajectory planning. These methods have demonstrated reliable performance on platforms with accurate dynamics models and torque-controlled actuators, but they require extensive tuning and degrade rapidly when applied to hardware with actuator nonlinearities, sensing noise, or mechanical backlash. Such limitations make classical approaches difficult to deploy on low-cost quadruped robots [10], [11].

Recent advances in reinforcement learning have enabled significant progress in legged locomotion by learning control policies directly from interaction. Several works have demonstrated agile and robust locomotion on high end quadruped platforms using deep reinforcement learning, often leveraging physics simulators and large-scale parallel training [1]. Actor critic methods in particular have shown strong performance by combining value estimation with policy optimization, enabling stable learning in high dimensional continuous control tasks [8], [12]. However, much of this literature focuses on robots equipped with torque control and high-fidelity sensing, which limits direct applicability to servo driven platforms [2].

Sim to real transfer remains a central challenge in reinforcement learning based locomotion. Domain randomization has emerged as an effective strategy to mitigate the reality gap by exposing policies to variations in physical parameters, sensing noise, and environmental conditions during training [4], [13], [18]. While domain randomization has been successfully applied to quadruped locomotion, existing studies often assume actuator models and sensing capabilities that exceed those available on low-cost robots. As a result, the robustness of learned policies on inexpensive hardware remains an open question [3], [7].

Several recent studies have explored learning-based locomotion on lower cost or open-source quadruped platforms, highlighting the importance of smooth control, energy efficiency, and stability [3], [5], [14]. These works suggest that algorithmic choices strongly influence gait quality and robustness, particularly when actuation and sensing are imperfect. Nevertheless, most evaluations consider a single learning method or differ in experimental setup, reward design, or terrain conditions, making fair comparison difficult [6].

Comparative studies of reinforcement learning algorithms for quadruped locomotion are relatively limited. While methods such as PPO and SAC are frequently used, direct comparisons under identical conditions are rare, especially on low-cost platforms [6], [14]. Derivative free approaches such as ARS have received comparatively little attention in legged locomotion, despite their simplicity and robustness to noise [7].

In contrast to prior work, this paper addresses the problem of learning a locomotion controller for a low-cost quadruped robot subject to realistic sensing and actuation constraints. The objective is to achieve stable and energy-efficient forward walking that generalizes across varying terrain conditions while remaining compatible with position-controlled servo hardware. The locomotion task is formulated as a continuous-time control problem modelled as a Markov decision process. At each control time step, the robot observes its current state and outputs continuous control actions that determine leg motion for forward progression while maintaining balance. The control objective is to learn a policy that maximizes the expected cumulative reward over an episode. The reward function balances forward velocity against penalties for instability, excessive joint motion, energy consumption, and foot slip. Episodes terminate upon loss of balance, violation of joint limits, or completion of a fixed time horizon. This formulation captures the fundamental tradeoffs between speed, stability, and energy efficiency inherent to low-cost quadruped locomotion.

3 Methodology

This section describes the reinforcement learning framework used to train and evaluate locomotion policies

for the SpotMicro quadruped robot, as shown in Figure 1. The framework integrates a physics-based PyBullet simulation, a Bezier gait-phase generator that provides structured locomotion priors, and the reinforcement learning algorithms ARS, SAC, and PPO trained under identical conditions. Proprioceptive observations and gait-phase information are processed by the learning algorithms to generate joint-level position targets, while reward signals guide iterative policy updates.

All components are designed to reflect the constraints of low-cost, position-controlled servo hardware, including limited sensing, actuation delays, and saturation effects. Domain randomization over terrain properties, actuation dynamics, and sensor noise is applied during training to improve robustness and enable fair comparison and reliable transfer across learning algorithms [3], [4].

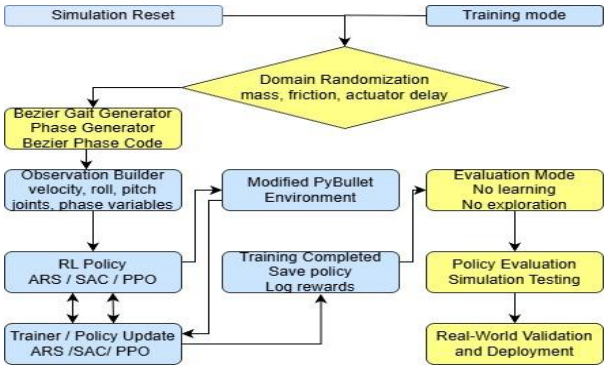


Figure 1. Reinforcement Learning Framework

3.1 Reinforcement Learning Framework

The RL framework employed in this work follows a unified perception action-learning loop implemented within a shared PyBullet-based control architecture for the SpotMicro quadruped. At each control timestep, proprioceptive observations comprising joint positions and velocities, base orientation and angular velocity from the inertial measurement unit, and internal gait phase information are collected and concatenated into a single state vector. This observation space is identical across ARS, PPO and SAC ensuring a fair algorithmic comparison. Rather than directly commanding raw joint trajectories, the learning policy outputs continuous actions that modulate parameters of a Bezier curve-based gait phase generator, including step timing, foot clearance, and residual joint-level offsets. The resulting foot trajectories are converted into joint-level targets and executed through position-controlled servo actuation within the physics-based simulation. Robot terrain interactions, actuator limits, and contact dynamics are modeled in PyBullet, and a scalar reward is computed at each timestep based on forward velocity, posture stability, smoothness, energy efficiency, and foot slip. During training, policy parameters are updated according to the respective learning rules of ARS, PPO, and SAC, with domain randomization applied at episode resets to improve robustness. During evaluation, trained policies are executed deterministically under both nominal and randomized conditions, and performance metrics such as cumulative reward, energy consumption, slip, and stability are logged for comparative analysis.

3.2 Simulation Environment

All experiments are conducted in a physics-based simulation environment developed using PyBullet [15], [18]. The simulation model is derived from the SpotMicro robot description and includes accurate kinematic structure, joint limits, and mass distribution. Unlike torque-controlled quadrupeds commonly used in prior work, SpotMicro relies on position-controlled servo motors. To reflect this, actuator dynamics are modelled with bounded position targets, response delay, saturation effects, and limited control bandwidth. These characteristics introduce realistic lag and nonlinearity into joint motion, closely approximating the behaviour of low-cost servos [3], [16].

Proprioceptive sensing is restricted to quantities that are realistically available on the platform. Observations include joint positions and velocities, body orientation and angular velocity from an inertial measurement unit, and internal gait phase information. Measurement noise is injected into sensor readings to emulate low-cost encoders and inertial sensors. No ground reaction force or joint torque measurements are assumed, ensuring that the learned policies remain deployable on the physical robot [4], [7].

3.3 Gait Representation and Control Structure

Rather than learning raw joint trajectories from scratch, a Bezier-based gait phase generator is employed to provide a structured locomotion prior [1],. For each leg i, the swing motion is generated as a phase-parameterized cubic Bezier trajectory,

$$p_i(\phi) = \sum_{k=0}^{3} \binom{3}{k} (1-\phi)^{3-k} \phi^k \left(p_{i,k} + \Delta p_{i,k}\right), \phi \in [0, 1) \quad (1)$$

where ϕ is a cyclic gait phase variable and $p_{i,k}$ denote nominal control points. The reinforcement learning policy outputs low-dimensional modulation terms $\Delta p_{i,k}$, enabling adaptive adjustment of foot placement, step height, and gait timing in response to terrain variations and external disturbances. Joint commands are obtained via inverse kinematics, while stance motion is handled implicitly through the cyclic phase progression.

This hybrid control structure significantly reduces the

dimensionality of the learning problem and improves training stability, particularly during early exploration [17]. By embedding rhythmic coordination into the control pipeline, the learning algorithm can focus on stability, balance, and robustness rather than discovering basic gait patterns. This approach is especially beneficial for low-cost quadrupeds, where unstable exploratory motions can quickly lead to falls and termination [3].

3.4 Reinforcement Learning Algorithms

ARS is a derivative free optimization method that operates directly in the parameter space of a linear policy. At each iteration, random perturbations are applied to the policy parameters, and updates are computed based on ranked returns. ARS is computationally efficient, insensitive to gradient noise, and well suited for problems with low dimensional policy representations [7], [6]. The parameter update rule used by ARS is given by

$$\theta_{\{t+1\}} = \theta_t + \frac{\alpha}{b\sigma_R}\sum_{i=1}^{b}(R(\theta_t + \nu\delta_i) - R(\theta_t - \nu\delta_i))\delta_i \quad (2)$$

SAC is an off-policy actor critic algorithm that optimizes a stochastic policy while maximizing expected return and entropy. The entropy regularization term encourages exploration and smooth action distributions, which is particularly important for stable locomotion. SAC employs twin critics to mitigate value overestimation and uses replay buffers for efficient sample reuse [8], [14]. The objective optimized by SAC is given by

$$J(\theta) = \mathbb{E}_{(s_t,\, a_t)\sim D}[Q_\phi(s_t,\, a_t) - \alpha \log \pi_\theta(\, a_t|s_t)] \quad (3)$$

PPO is an on-policy gradient method that constrains policy updates using a clipped objective. PPO is widely used in robotic control due to its simplicity and stability under certain conditions. The clipped surrogate objective optimized by PPO is given by

$$J^{PPO}(\theta) = \mathbb{E}_t\big[\, min\,(\, r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big] \quad (4)$$

In this work, PPO serves as a representative on policy baseline for comparison with ARS and SAC [9], [6]. All algorithms use identical observation spaces, action spaces, reward functions, and training horizons to ensure a fair evaluation.

3.5 Algorithms Reward Function Design

The reward function is designed to balance forward locomotion performance with stability, smoothness, and energy efficiency. At each simulation time step t, the instantaneous reward is defined as

$$r_t = w_v v_x - w_\theta\big(|\theta_{roll}| + |\theta_{pitch}|\big) - w_a \sum_{i=1}^{12}\ddot{q}_i^2 - w_u \sum_{i=1}^{14} u_i^2 - w_s \sum_{k\,\in\, C}^{4}\big\|v_{foot,k}^{\perp}\big\| \quad (5)$$

where v_x denotes the forward velocity of the robot base along the desired direction of motion. The terms θ_{roll} and θ_{pitch} represent the body roll and pitch angles obtained from the base orientation, penalizing excessive body tilt. The quantities $\ddot{q}_i$ and u_i correspond to the angular acceleration and normalized joint position commands and gait modulation signals, respectively, and are included to discourage abrupt joint motions and high actuation effort. The final term penalizes lateral foot slip velocity $v_{foot,k}^{\perp}$ for feet in ground contact C, promoting stable stance behaviour. Additional penalties discourage unstable postures and erratic motion that could damage hardware [5], [14].

The reward formulation reflects practical deployment concerns for low-cost quadrupeds, where smooth trajectories reduce servo wear and stable gaits minimize fall risk. Reward weights are selected empirically to ensure that no single term dominates learning while preserving consistent behaviour across algorithms [6].

3.6 Domain Randomization

To improve robustness and sim-to-real transfer, domain randomization [4], [14], [18] is applied at episode reset by perturbing key physical and actuation parameters. Base and leg masses are randomized within ±20% of nominal values, battery voltage is sampled between (7.0 - 8.4 V), motor viscous damping is varied within (0-0.01 N·m·s/rad), and foot ground friction coefficients are randomized in the range (0.8-1.5). These variations introduce uncertainty in inertial, actuation, and contact dynamics while maintaining a consistent control structure.

Beyond stochastic parameter randomization, policies are trained and evaluated on both flat and uneven terrain using a height-field ground model, and under different sensing configurations, including the absence of foot contact information. These settings are applied deterministically per experiment to enable controlled evaluation of terrain robustness and sensing dependence.

4 Experimental Setup

This section describes the training configuration, evaluation protocol, terrain and sensing scenarios, and performance metrics used to assess the locomotion policies learned by the different RL algorithms.

4.1 Training Configuration

All RL algorithms are trained in the same PyBullet simulation environment using identical robot models, observation spaces, action spaces, and reward functions [15]. The observation space has a dimensionality of 16, and the action space consists of 14 continuous control variables corresponding to the Bezier-curve gait parameterization.

Each algorithm is trained for a total of four million environment steps, which was found to be sufficient for convergence across all methods. The control frequency is fixed across experiments to reflect the update rate achievable on the SpotMicro platform. For ARS, a linear policy is used with parameter perturbations sampled symmetrically at each iteration [7]. The number of rollouts per update and step size are selected to balance convergence speed and stability. The complete training configuration of RL algorithm is summarized in Table 1.

SAC and PPO are implemented using multilayer neural network policies with the same architectural design, ensuring that performance differences are not influenced by disparities in model capacity. The complete set of SAC specific hyperparameters is summarized in Table 1(II), while the corresponding PPO hyperparameters configuration are detailed in Table 1(III). To account for stochasticity in training, each algorithm is trained using multiple random seeds. Reported results represent averages across these runs, ensuring that conclusions are not influenced by outlier behaviours.

4.2 Terrain and Sensing Scenarios

To evaluate robustness and generalization, policies are trained and tested across multiple terrain configurations. These include flat terrain, which represents structured indoor environments, and uneven terrain with randomized height variations, which emulates outdoor or unstructured surfaces. Terrain parameters are randomized during training to prevent over fitting to a single surface profile [4].

In addition to terrain variation, different sensing configurations are considered. In the nominal setting, the policy receives all available proprioceptive observations. In a reduced sensing setting, contact information is removed, forcing the policy to rely solely on joint and inertial measurements. This evaluation highlights the sensitivity of each algorithm to sensing uncertainty, which is particularly relevant for low-cost robots with limited feedback [3].

4.3 Evaluation Protocol

After training, policies are evaluated without exploration noise to assess their deterministic performance. Each policy is tested over multiple episodes for each terrain and sensing configuration. Episodes terminate when the robot falls, exceeds joint limits, or reaches a fixed time horizon.

Performance is evaluated using consistent metrics across algorithms. These include average episodic return, forward velocity, energy consumption, smoothness of joint trajectories, body stability, foot slip, and fall rate. Learning curves are obtained by recording episodic returns throughout training, while stability and robustness metrics are computed during post training evaluation runs.

4.4 Performance Metrics

The following metrics are derived from the reward components defined in Equation (5) and are reported separately for analysis.

Forward Locomotion Reward: $R_v = v_x$, encourages the robot to move in the desired direction.
Smoothness Penalty: $R_s = -\sum_{i=1}^{12} \ddot{q}_i^2$, penalizes high-frequency oscillations in joint movements.
Energy Efficiency Penalty: $R_e = -\sum_{i=1}^{14} u_i^2$, minimizes jerky motion, oscillations, and instability
Body Posture Stability: $R_p = -(|roll| + |pitch|)$, ensures stable roll and pitch.
Foot-Slip Penalty: $R_{slip} = -\sum_{k \in C}^{4} \left\| v_{foot,k}^{\perp} \right\|$, penalizes lateral foot motion during stance.
Termination Penalty: C_{fall}, implicit termination penalty.

Together, these metrics provide a comprehensive assessment of locomotion behaviour and practical deploy ability.

Table 1 Training Configuration of Reinforcement Learning Algorithms (ARS, SAC, PPO)

I. ARS

Policy type:	Linear
State dimension:	16
Action dimension	14
Learning rate (α):	0.02
Exploration noise standard deviation (ν)	0.02
Total perturbation	64
Top-performing direction	32
Episode length	5000
Reward normalization	Enabled
Observation normalization	Enabled
Evaluation frequency	Every 10 iterations

II. SAC

Actor–critic architecture	2*256
Learning rate	3×10^{-4}
Entropy coefficient	Log(0.1)
Discount factor (γ)	0.99
Soft target update coefficient (τ)	0.01
Replay buffer capacity	1 million transitions
Batch size	256
Model checkpoint frequency	Every 8192 environment steps

III. PPO

State dimension	16
Action dimension	14
Action range	[−1, 1]
Actor–critic network	2*256
Rollout length	2048
Epochs per update	10
Mini-batch size	256
Discount factor (γ)	0.99
Advantage function (λ)	0.95
Clipping parameter (ε)	0.2
Learning rate	3×10^{-4}
Optimizer	Adam

5 Results

In this section results are reported across learning performance, stability, smoothness, energy efficiency, and robustness under terrain and sensing variations. All metrics are computed using identical evaluation protocols to ensure a fair comparison.

5.1 Learning Performance

Figures 2(a)-2(d) show the reward evolution of ARS, SAC, and PPO over four million environment steps under flat and rough terrain with and without contact sensing. ARS exhibits the fastest reward increase in all configurations and achieves the highest final average return, reaching approximately 17.4 on flat terrain with contact sensing Figure 2(a). Consistent upward trends are also observed in the remaining settings (Figures 2(c)-(d)), with increased variability under rough terrain.

SAC shows a gradual and comparatively smooth improvement across all scenarios, converging to an average return of approximately 10.2 on flat terrain Figures 2(a) and 2(b). Under rough terrain and no-contact conditions Figures 2(c) and 2(d), the learning rate decreases, though stable progression is maintained. PPO demonstrates the slowest convergence and highest variance across all cases. As shown in Figure 2(a), PPO reaches a final average return of approximately 7.2 on flat terrain with contact sensing. Performance variability increases markedly on rough terrain, particularly without contact sensing Figure 2(d), where frequent reward oscillations are observed.

5.2 Stability and Gait Quality

Stability metrics reveal clear differences in gait quality across algorithms, as shown in Figure 3(a) and 3(b). SAC consistently produces the most stable locomotion, exhibiting the lowest relative foot-slip error, Figure 3(a) and the lowest fall rate, Figure 3(b) across all evaluation scenarios. This indicates reliable foot placement and effective balance maintenance during locomotion. The resulting stance and swing transitions remain smooth, leading to coherent and visually natural gait patterns throughout the trials. ARS achieves moderate stability, particularly under flat terrain conditions, as reflected by intermediate foot-slip and fall-rate values in Figures 3(a) and 3(b).

While the learned gaits remain functional and largely consistent, increased slip events and occasional balance losses are observed under uneven terrain or sensing limitations. The linear policy representation restricts the ability to perform fine-grained corrective actions, resulting in higher oscillations compared to SAC. PPO exhibits the highest instability among the three methods. Elevated foot-slip error and the highest fall rate shown in Figures 3(a) and 3(b) indicate inconsistent foot placement and reduced contact reliability. These effects are especially pronounced under sensing uncertainty, where frequent oscillations and disrupted limb coordination lead to degraded gait stability

5.3 Smoothness and Energy Efficiency

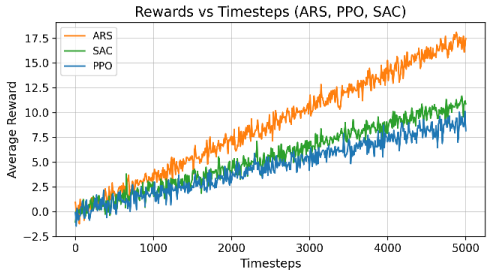
Smoothness and energy efficiency are quantified using normalized energy consumption and trajectory smoothness metrics, as shown in Figure 4, where lower values indicate better performance. As illustrated in Figure 4, ARS records the lowest normalized energy consumption, with a value of approximately 1.0, indicating the most energy-efficient locomotion among the evaluated methods. Its corresponding smoothness metric is approximately 1.2, reflecting moderate trajectory smoothness with limited abrupt control variations.

SAC exhibits a higher normalized energy consumption of approximately 1.5, but achieves the lowest smoothness metric, approximately 0.8, indicating the smoothest joint trajectories across all three algorithms. This result reflects consistent and gradual control updates throughout the gait cycle. PPO shows the highest normalized energy consumption, approximately 2.5, and the highest smoothness metric, approximately 2.2, as shown in Figure 4. These values indicate increased energy usage and less smooth joint motion, consistent with frequent corrective actions during locomotion.

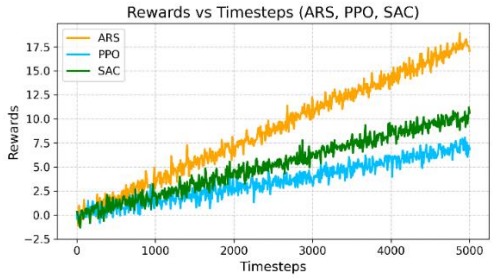
5.4 Terrain Robustness

Performance under terrain variation highlights clear robustness differences between the evaluated methods, as shown in Figure 5. On flat terrain, both ARS and SAC achieve stable and efficient locomotion, exhibiting high relative performance in both contact and no-contact sensing conditions. In contrast, PPO records noticeably lower performance even on flat surfaces, indicating reduced robustness.

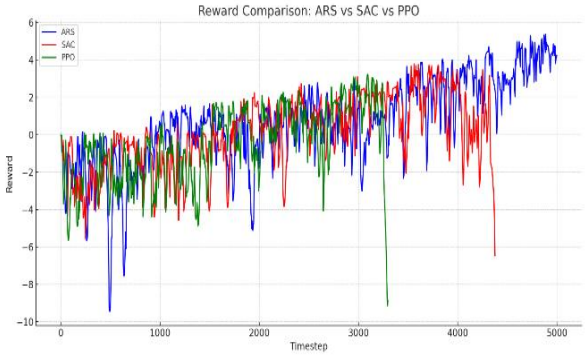
On uneven terrain, SAC consistently outperforms the other methods, achieving the highest relative performance under both contact-enabled and reduced-sensing configurations. ARS shows moderate degradation on rough surfaces, with a clear reduction in performance compared to flat terrain, but continues to maintain forward locomotion. In contrast, PPO exhibits substantial performance degradation on uneven terrain, particularly when contact information is unavailable, reflecting difficulty in maintaining balance and consistent gait execution.



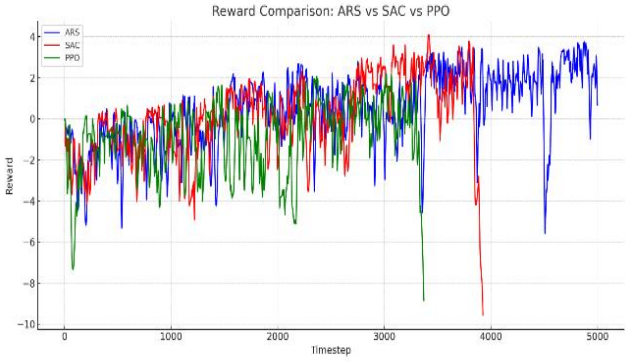
(a)



(b)



(c)



(d)

Figure 2(a) Flat terrain with contact sensing Figure 2(b) Flat terrain with no contact sensing Figure 2(c) Rough terrain with contact sensing Figure 2(d) Rough terrain with no contact sensing.

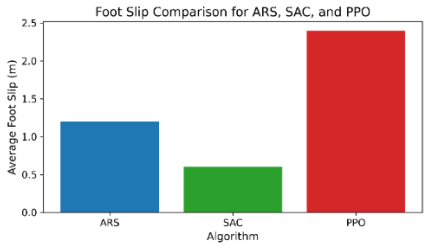


Figure 3(a) Foot Slip comparison

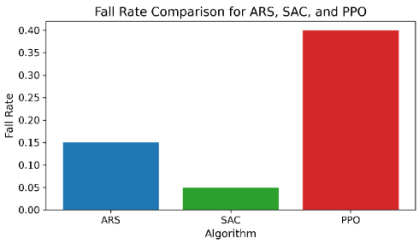


Figure 3(b) Fall-rate Comparison

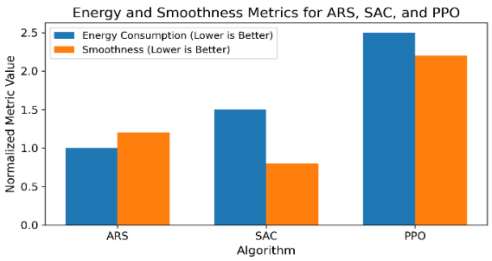


Figure 4. Energy and smoothness metrics

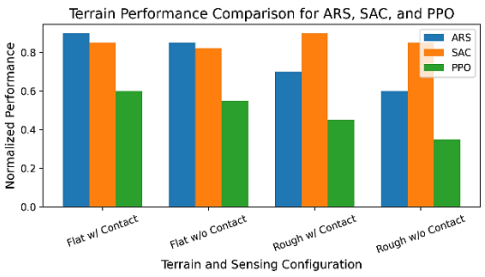


Figure 5. Terrain Performance comparison

6 Discussion

The results highlight clear behavioral differences among reinforcement learning algorithms for locomotion control on a low-cost, servo-driven quadruped. Although all methods achieve forward motion, they vary considerably in stability, robustness, and practical suitability. ARS achieves the fastest convergence and highest reward, demonstrating that simple, derivative-free optimization can be effective with low-dimensional policy representations. Its energy-efficient and repeatable gaits perform well on flat terrain; however, limited policy expressiveness restricts ARS on uneven surfaces where nonlinear balance corrections are required.

SAC provides the most stable and robust performance across all test conditions. Entropy-regularized learning produces smoother control actions, reducing joint stress and foot slip, and improving mechanical safety. Its robustness to terrain variation and sensing noise makes SAC well suited for the uncertainties inherent in low-cost robotic platforms. In contrast, PPO underperforms due to sensitivity to noise and domain randomization, leading to unstable training, higher energy consumption, and frequent failures on challenging terrain. Overall, these results emphasize that deployable locomotion requires stability and robustness beyond reward maximization, positioning SAC as the most practical solution

7 Conclusion

This paper compared ARS, SAC, and PPO for locomotion control on a low-cost, servo-driven quadruped using a unified, physics-based simulation that captures actuation limits, sensing noise, and terrain variability. The results confirm that algorithm choice significantly affects locomotion performance. ARS achieved fast convergence and high rewards, making it effective for rapid training in structured environments. SAC produced the most stable, smooth, and robust gaits, particularly under uncertainty, indicating strong suitability for real-world deployment. PPO showed lower stability and higher energy consumption, reducing its effectiveness on low-cost hardware. Future work will focus on sim-to-real transfer, hybrid ARS-SAC training strategies, and extending the framework to more complex terrains and adaptive behaviours.

References

[1] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics, 4*.

[2] Liu, D., Zhang, T., Yin, J., & See, S. (2025). Unified Locomotion Transformer with Simultaneous Sim-to-Real Transfer for Quadrupeds. *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 12546-12553.

[3] Wang, Y., Sagawa, R., & Yoshiyasu, Y. (2024). Learning Advanced Locomotion for Quadrupedal Robots: A Distributed Multi-Agent Reinforcement Learning Framework with Riemannian Motion Policies. *Robotics*, *13*(6), 86.

[4] Tiboni, G., Arndt, K., & Kyrki, V. (2022). DROPO: Sim-to-Real Transfer with Offline Domain Randomization. *Robotics Auton. Syst., 166*, 104432.

[5] Margolis GB, Yang G, Paigwar K, Chen T, Agrawal P. Rapid locomotion via reinforcement learning. The International Journal of Robotics esearch. 2024;43(4):572-587.

[6] Zhang, X., Mao, W., Mowlavi, S., Benosman, M., & Başar, T. (2023). Controlgym: Large-scale control environments for benchmarking reinforcement learning algorithms. *Conference on Learning for Dynamics & Control*.

[7] Mania, H., Guy, A., & Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. *ArXiv, abs/1803.07055*.

[8] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic Algorithms and Applications.

[9] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *ArXiv, abs/1707.06347*.

[10] C. Cun, Q. Yang, Z. Li, M. C. Zhou, and J. Pang, “Model predictive optimization and control of quadruped whole-body locomotion,” IEEE/CAA J. Autom. Sinica, vol. 12, no. 10, pp. 2103–2114, Oct. 2025.

[11] J. Ding, S. Kim, and C. Atkeson, “Trajectory optimization for legged robots with complex contacts,” IEEE Robotics and Automation Letters, vol. 7, no. 4, 2022.

[12] Fan, J., Cramariuc, A., Portela, T., & Hutter, M. (2025). Pretraining in Actor-Critic Reinforcement Learning for Robot Motion Control. *ArXiv, abs/2510.12363*.

[13] A. Shakerimov, T. Alizadeh and H. A. Varol, "Efficient Sim-to-Real Transfer in Reinforcement Learning Through Domain Randomization and Domain Adaptation," in *IEEE Access*, vol. 11, pp. 136809-136824, 2023.

[14] J. Dowdy and J. C. Vaz, "Isaac Sim-to-Real: Reinforcement Learning based Locomotion for Quadrupeds," *2025 IEEE 21st International Conference on Automation Sci. and Engg. (CASE)*, Los Angeles, 2025, pp. 2194-2199.

[15] Coumans, Erwin, and Yunfei Bai. "Pybullet, a python module for physics simulation for games, robotics and machine learning." 28 Jan. 2016.

[16] Pierre-Brice Wieber, Russ Tedrake, Scott Kuindersma. Modeling and Control of Legged Robots. Springer Handbook of Robotics, Springer International Publishing, pp.1203-1234, 2016.

[17] Feng, W., Wang, Z., Xu, H., Zhou, Y., He, B., & Dong, C. (2025). Multiple gait locomotion generation for quadruped robots based on trajectory planning and reinforcement learning. *Control Engineering Practice*.

[18] Tiwari, R., Khapre, S., & Singh, A. (2026). Reinforcement learning in robotic systems: A review on sim-to-real transfer. *Robotics and Autonomous Systems*, 105327.