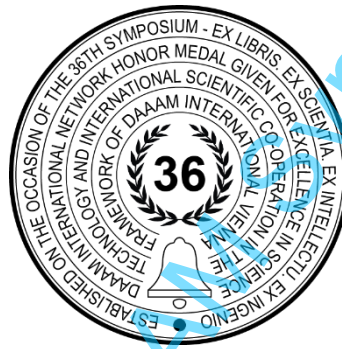


ON- AND OFF-POLICY DEEP REINFORCEMENT LEARNING FOR HEXAPOD LOCOMOTION

Felix Dauser, Kemajl Stuja, Mohamed Aburaia & Ali Aburaia



This Publication has to be referred as: Dauser, F[elix]; Stuja, K[emajl]; Aburaia, M[ohamed] & Aburaia, A[li] (2026). Title of Paper, Proceedings of the 37th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/37th.daaam.proceedings.xxx

Abstract

Locomotion policies trained in simulation are only valuable if their behaviour remains reliable once deployed on physical hardware. Most work on legged locomotion relies on the on-policy method Proximal Policy Optimization (PPO), whereas the off-policy method Soft Actor-Critic (SAC) has been explored mainly on other robotic platforms. Moreover, sim-to-real performance is often summarised as a single aggregate metric. A controlled comparison of PPO and SAC is presented for straight-line locomotion on the PhantomX AX Hexapod Mark II, holding the task, reward function, hidden-layer sizes, and velocity curriculum constant and avoiding motion priors. Transfer is evaluated in two stages—first from the training simulator (Isaac Lab) to an independent simulator (Gazebo Harmonic), and then from Gazebo to the physical robot—and execution parameters outside the learned policy (e.g., control frequency, action scaling, filtering) are examined as an additional experimental factor. The results show that the dominant source of transfer loss depends on both the algorithm and the execution configuration; notably, modifying only the execution configuration can change hardware velocity by up to a factor of three. The evaluation draws on three training seeds per algorithm, fifty evaluation runs, and camera-based reference measurements on the physical system.

Keywords: Deep Reinforcement Learning; Hexapod Locomotion; Sim-to-Real Transfer; Proximal Policy Optimization; Soft Actor-Critic.

1. Introduction

Hexapods are well suited to locomotion on unstructured ground because six independently actuated legs provide high static stability and mechanical redundancy. Before such terrain can be addressed on a physical platform, however, a controller developed in simulation must first transfer to the robot while maintaining stable locomotion. The PhantomX AX Hexapod Mark II used in this study has eighteen actuated joints. Classical gait generators can coordinate these joints; however, adapting them to changes in actuator behaviour requires explicit modelling and manual tuning. Deep reinforcement learning instead learns a policy that maps proprioceptive observations directly to joint commands, and comparable approaches have already been transferred successfully to physical quadrupeds [1], [2].

Two gaps motivate the present comparison. First, the on-/off-policy choice has not been isolated experimentally for hexapod locomotion without motion priors. Second, sim-to-real transfer is usually reported as a single aggregated step, which obscures where performance loss originates.

Accordingly, a controlled comparison of Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) is conducted under an identical locomotion task, transfer is separated into two measurable stages, and the execution configuration is treated as an additional experimental factor. The analysis addresses three questions: (i) how many environment steps each algorithm requires to reach the commanded velocity; (ii) how locomotion performance changes from Isaac Lab to Gazebo and from Gazebo to the physical robot; and (iii) how strongly the result changes when execution parameters are modified while the learned policy itself remains unchanged.

2. State of the art

On-policy methods update the policy using recently collected data and discard those samples after a limited number of optimisation steps, whereas off-policy methods can reuse past transitions stored in a replay buffer. Proximal Policy Optimization (PPO) is an on-policy algorithm that restricts updates via a clipped probability ratio between the new and the previous policy [3]. Soft Actor-Critic (SAC) is off-policy and augments the objective with an entropy term, with a temperature parameter that controls the exploration–exploitation trade-off (often tuned automatically) [4]. For quadruped locomotion, Rudin et al. train several thousand ANYmal agents in parallel on the GPU and transfer the resulting policy to the physical robot [2]. Hwangbo et al. identify inaccurate actuator dynamics as an important contributor to the remaining simulation-to-reality gap [5]. Domain randomisation is a common strategy to reduce this sensitivity by varying physical and sensory parameters during training [6]. DAAAM studies have also documented the design of SCARA and pick-and-place manipulators [7], [8], providing examples of robotic platforms with different kinematic architectures.

Published work on learned hexapod locomotion is more limited and often introduces additional motion structure. Kim et al. and Liu et al. reduce the exploration burden of the eighteen-dimensional action space through reference trajectories and motion priors [9], [10]. Wang et al. use the learned policy to adjust parameters of a central pattern generator rather than commanding the joints directly [11]. While SAC has been widely applied to continuous control, it has only recently been scaled to training regimes comparable to PPO in massively parallel settings [4], [12]. A controlled comparison of these two learning paradigms without additional gait priors has not been reported for the hexapod considered here. Such priors are deliberately excluded in order to avoid introducing another confounding factor into the comparison between PPO and SAC.

3. Material and methods

3.1 Platform and Task Formulation

The experimental platform is a PhantomX AX Metal Hexapod Mark II with eighteen Dynamixel AX-12A servos (three per leg). Each servo provides a holding torque of 1.5 N·m at 12 V [13]. A Raspberry Pi 5 runs ROS 2 and the policy inference node; a BNO085 inertial measurement unit supplies angular velocity and projected gravity (in the body frame); and an OpenCR 1.0 board handles communication with the servos. Policy inference is configured at 50 Hz. The firmware closes the servo position loop at 20 Hz, while the measured effective update rate is approximately 16.7 Hz.

The policy receives a 66-dimensional proprioceptive observation consisting of body linear and angular velocity, projected gravity, the velocity command, joint positions, joint velocities, and the previous action. Only the forward component of the three-dimensional velocity command is non-zero in the reported runs, and no exteroceptive information enters the policy. On the physical robot, body linear velocity is estimated rather than measured directly. Observations are normalised using statistics collected during training and kept fixed at deployment; any difference between the training and hardware observation distributions can therefore affect the normalised policy input.

The policy outputs an action vector $a \in \mathbb{R}^{18}$. Joint targets are computed as $q_{\text{target}} = q_{\text{def}} + s \cdot \text{clip}(a, -1, 1)$, with elementwise clipping $\text{clip}(x, l, u) = \min(\max(x, l), u)$ and amplitude $s = 0.5$ rad. The resulting joint targets are passed to a PD controller with gains $k_p = 10.0$ and $k_d = 2.0$, a torque limit of 1.5 N·m, and a joint velocity limit of 3.5 rad/s. The policy does not output torques, gait parameters, or foot positions, so both algorithms operate on the same action representation.

The reward is a weighted sum of thirteen terms and was tuned manually over forty training runs per algorithm. The dominant terms are forward-velocity tracking with a weight of +12.0 and a penalty of -25.0 for exceeding the commanded speed; further terms account for yaw rate, body height, orientation, and action rate. A tripod term with a weight of +3.0 rewards alternating contact between the two tripod groups, as a policy without an explicit alternation incentive can otherwise remain in a statically stable three-leg support configuration. Four of the thirteen reward terms depend on contact information available only in Isaac Lab. The reward is therefore not used as a cross-domain performance metric; transfer is evaluated with quantities that can be obtained consistently across domains.

Each episode lasts 80 s, corresponding to 4,000 policy steps.

An episode terminates early if any of the following conditions is met: (i) the squared horizontal component of projected gravity exceeds 0.05, (ii) any body contact force exceeds 1 N, or (iii) the body height leaves the interval [0.05, 0.40] m.

3.2 Policy Training

Training is performed in Isaac Lab using the GPU-accelerated PhysX backend [14]. Both algorithms are implemented with the skrl library and therefore share the same software framework [15]. Each run uses 128 parallel environments and a budget of 1.2 million policy steps per environment, corresponding to approximately 153.6 million transitions in total. The curriculum thresholds and the column “steps to 0.10 m/s” in Tab. 1 are given in the same unit. The hidden networks consist of three fully connected layers with 256 units and ELU activations. Observation-normalisation statistics are learned during training and frozen for deployment.

The algorithm-specific components remain unchanged. PPO uses a shared actor-critic trunk, a state-independent action standard deviation and rollouts of 12,288 transitions optimised over five epochs. SAC uses a separate actor, a state-dependent action standard deviation, two critics with target networks and a replay buffer.

A three-stage curriculum increases the commanded forward velocity from 0.00 to 0.05 m/s at 150,000 steps and to 0.10 m/s at 600,000 steps. Domain randomisation samples the static foot friction between 0.4 and 0.6 and the base mass offset within ± 0.5 kg. Gaussian observation noise with $\sigma = 0.01$ is added to angular velocity, projected gravity and joint positions, and noise with $\sigma = 0.05$ is added to joint velocities. Each algorithm is trained with three random seeds [16]. All runs were executed on the same workstation with an NVIDIA RTX 5070 Ti.

3.3 Transfer Chain and Execution Configurations

For deployment, the selected policy is exported as a TorchScript model that contains the actor and the frozen observation-normalisation statistics. Joint order, action scaling, joint limits, rest pose, smoothing and the limit on setpoint changes remain outside the exported model. The experiment therefore keeps the learned policy fixed while varying selected parts of the downstream execution path. Fig. 1 illustrates the resulting transfer chain.

Gazebo Harmonic [17] executes the same ROS 2 nodes on the DART physics backend. Because its position interface does not provide a parameterisable torque-level PD controller, an effort interface with an additional PD bridge is used. The effective foot friction is set to 0.5, which lies within the 0.4 to 0.6 range used during training. Foot friction alone therefore does not explain the transfer differences discussed below.

In Isaac Lab, observations are passed to the policy without smoothing, and the joint targets are applied to the internal actuator model without additional filtering or rate limiting. Two downstream execution configurations are compared with this reference. Configuration A transfers the Isaac Lab settings directly, using $k_p = 10.0$, $k_d = 2.0$ and no smoothing or rate limiting. Configuration B is calibrated in Gazebo through a grid search comprising 3001 runs over k_p , k_d , the smoothing factors and the maximum setpoint change. The smoothing factors are denoted α_{act} for the exponential moving average on the action and α_{obs} for the one on the observation, and $madf$ denotes the maximum change of the joint target per policy step in multiples of the nominal step. The selected PPO parameters are $k_p = 6.0$, $k_d = 0.03$, $\alpha_{act} = 0.10$, $\alpha_{obs} = 0.05$ and $madf = 2.0$, while SAC uses $k_p = 6.5$, $k_d = 0.08$, $\alpha_{act} = 0.15$, $\alpha_{obs} = 0.10$ and $madf = 2.2$. On the physical robot, the external PD bridge is omitted because the AX12A servos close the position loop internally. Configurations A and B therefore differ on the physical robot only in the smoothing factors and the rate limit, and the controller gains have no effect there. The smoothing and rate-limit settings are transferred unchanged.

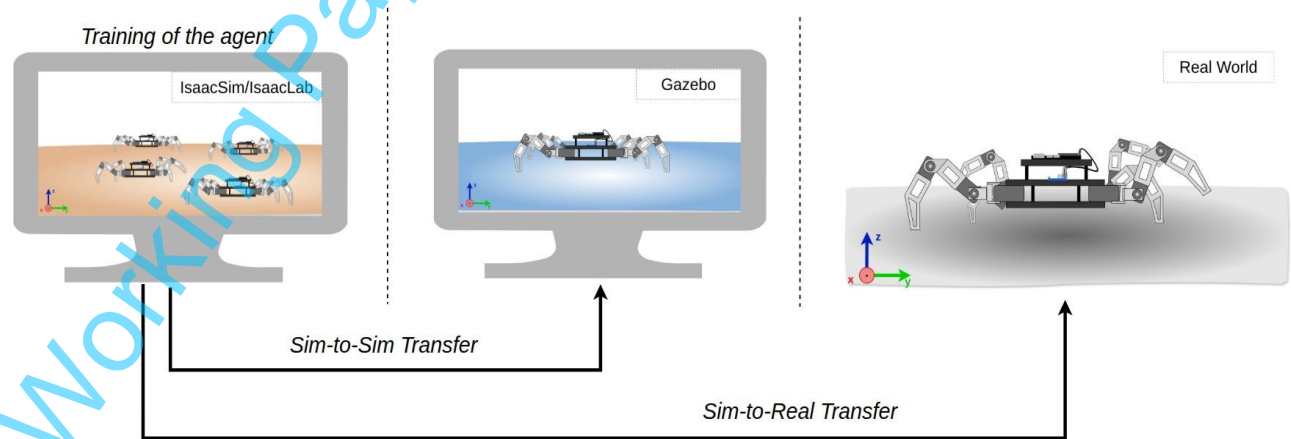


Fig. 1. Transfer chain from training in Isaac Lab to execution in Gazebo and on the physical robot.

3.4 Evaluation Protocol and Metrics

The downstream evaluation combines the two learning algorithms with the two execution-parameter sets, yielding four conditions. Each hardware condition is repeated five times for 60 s at a constant command of 0.10 m/s. The curriculum is disabled and the deterministic mean action is used. In addition, all six trained checkpoints are evaluated five times in Isaac Lab. The evaluation therefore comprises 30 Isaac Lab runs and 20 hardware runs.

Hardware motion is recorded by a stationary camera at approximately 30 fps. An AprilTag with an edge length of 189 mm is mounted on the robot, and four stationary ground markers spanning 1.50×2.00 m define a homography to a metric ground plane. In an additional manual validation, one trajectory measured 224.4 cm manually and 225.0 cm with camera-based tracking, corresponding to a relative deviation of $|225.0 - 224.4| / 224.4 \approx 0.27\%$.

The reported metrics are the velocity ratio $\rho = v_{\text{net}}/v_{\text{cmd}}$, path straightness, yaw drift, the body oscillation ω_{rms} defined as the root mean square of roll and pitch rate, and the tripod phase offset, for which 180° represents ideal alternation. The step frequency is the mean over all six legs, whereas the cycle duration annotated in the gait diagrams refers to the tripod cycle; the two quantities therefore coincide only for a regular alternating gait. No significance tests are performed because of the small number of repetitions. The scope is restricted to one robot, one selected policy per algorithm, and straight-line locomotion at a constant command on level ground; the results do not cover cornering, changing target velocities, or uneven terrain.

Run	R_{end}	v_{end}	p_{TO}	steps to	t
		[m/s]		0.10 m/s	[h]
PPO seed 1	1801	0.122	0.89	630,000	5.75
PPO seed 2	1665	0.121	0.83	624,000	5.68
PPO seed 3	1640	0.118	0.80	624,000	5.85
SAC seed 1	438	0.123	0.14	612,000	7.82
SAC seed 2	922	0.124	0.29	612,000	7.32
SAC seed 3	1551	0.122	0.69	612,000	7.52

Tab. 1 Training results per seed averaged over the final ten percent of the training budget: p_{TO} denotes the fraction of episodes reaching the maximum episode length

4. Experimental results

4.1 Training Results

Tab. 1 summarises the six training runs. All runs exceeded the commanded forward velocity by the end of training, and all six seeds settled above the command despite the overspeed penalty. Accordingly, the velocity ratio is interpreted as a deviation from the command rather than as a performance advantage, and values closer to 1 are considered better throughout. SAC reached both curriculum velocity thresholds in fewer or the same number of environment steps as PPO. After the command increased to 0.10 m/s, all SAC seeds required a further 12,000 steps to reach the target, whereas PPO required between 24,000 and 30,000 steps. This advantage in environment steps did not translate into shorter wall-clock training. At the end of training, PPO completed 80–89% of its episodes via the regular time limit, compared with 14–69% for SAC (Tab. 1). PPO processed approximately 57–59 training steps per second, compared with 43–46 for SAC. Variation across seeds differed substantially by metric. Final episodic returns varied by approximately 9.8% for PPO and 254% for SAC, whereas the spread in final forward velocity was only 3.6% for PPO and 1.6% for SAC. The large variation in SAC returns is largely explained by differences in episode length. Lower episodic returns therefore stem primarily from shorter episodes rather than reduced forward velocity, rendering episodic return insufficient as a standalone performance measure for SAC. Seed 3 is selected for both algorithms and used for the subsequent transfer analysis.

4.2 Baseline and Transfer to Gazebo

All thirty Isaac Lab evaluation runs finish without a fall or early termination. Tab. 2 lists the baseline metrics of the six trained checkpoints. Values are the mean and standard deviation over five evaluation seeds. Straightness values slightly above one result from the discretised path-length calculation. The selected PPO checkpoint reaches $\rho = 1.230 \pm 0.008$, a straightness of 1.030 and a lateral RMS deviation of 0.16 m. The other two PPO seeds reach lateral deviations of 1.06 and 1.53 m. The selected SAC checkpoint reaches $\rho = 1.013 \pm 0.031$ and therefore follows the commanded speed more closely within the training simulator.

Checkpoint	v [m/s]	Straightness	Yaw drift [°/s]	$dlat_{RMS}$ [m]
PPO seed 1	0.1064 ± 0.0057	0.907	-1.62	1.06
PPO seed 2	0.0888 ± 0.0064	0.740	+2.56	1.53
PPO seed 3	0.1230 ± 0.0008	1.030	-0.06	0.16
SAC seed 1	0.0884 ± 0.0098	0.890	+1.72	0.91
SAC seed 2	0.0857 ± 0.0057	0.908	+1.62	0.83
SAC seed 3	0.1013 ± 0.0031	0.995	-0.80	0.50

Tab. 2. Baseline metrics of the six checkpoints in Isaac Lab at a commanded velocity of 0.10 m/s.

Checkpoint selection was performed in the training domain before transfer using the same criterion for both algorithms. The selection was based on the Isaac Lab baseline metrics for velocity ratio and trajectory quality in Tab. 2, rather than on episodic return.

When transferred to Gazebo with the original Isaac Lab execution parameters, both selected policies lose most of their forward motion. PPO reaches only 0.0105 ± 0.0052 m/s and SAC 0.0126 ± 0.0075 m/s, corresponding to a reduction of 88 to 92% relative to the respective Isaac Lab reference. At the same time, the body height decreases to 0.070 m. Calibrating the execution path almost completely restores PPO. Its velocity increases to 0.1316 ± 0.0004 m/s with a straightness of 1.032. The straightness slightly above one results from the discretised path-length calculation and becomes 0.999 when the same discretisation is used consistently. The body height recovers to 0.106 m, and the body oscillation decreases from 0.532 to 0.029 rad/s (Fig. 2).

SAC responds much less strongly to the same calibration procedure. Its velocity increases only to 0.0247 ± 0.0051 m/s, which is approximately one quarter of the commanded value. The parameter search reflects this difference in robustness. PPO converges after 798 runs over two sweeps. SAC requires 2203 runs over eight sweeps. The larger search effort required for SAC is consistent with a narrower region of viable downstream parameters. The two searches did not cover identical parameter ranges, so this comparison is indicative rather than conclusive. For PPO, the recovery is associated with lower controller gains together with stronger smoothing and a stricter limit on setpoint changes.

The joint-level behaviour provides additional evidence for the transfer difference. The mean absolute tracking error across all eighteen joints changes from 0.285 to 0.385 rad for PPO and from 0.261 to 0.416 rad for SAC. The error also changes from predominantly dynamic tracking deviations to a more persistent offset. For PPO, the joint amplitudes decrease while their relative distribution remains similar. SAC shows a different pattern. All six tibia amplitudes fall below 0.02 rad, while individual femur joints exceed their Isaac Lab amplitudes. The motion is therefore redistributed rather than uniformly scaled. Correspondingly, the number of nearly stationary joints changes from three to zero for PPO but from two to five for SAC.

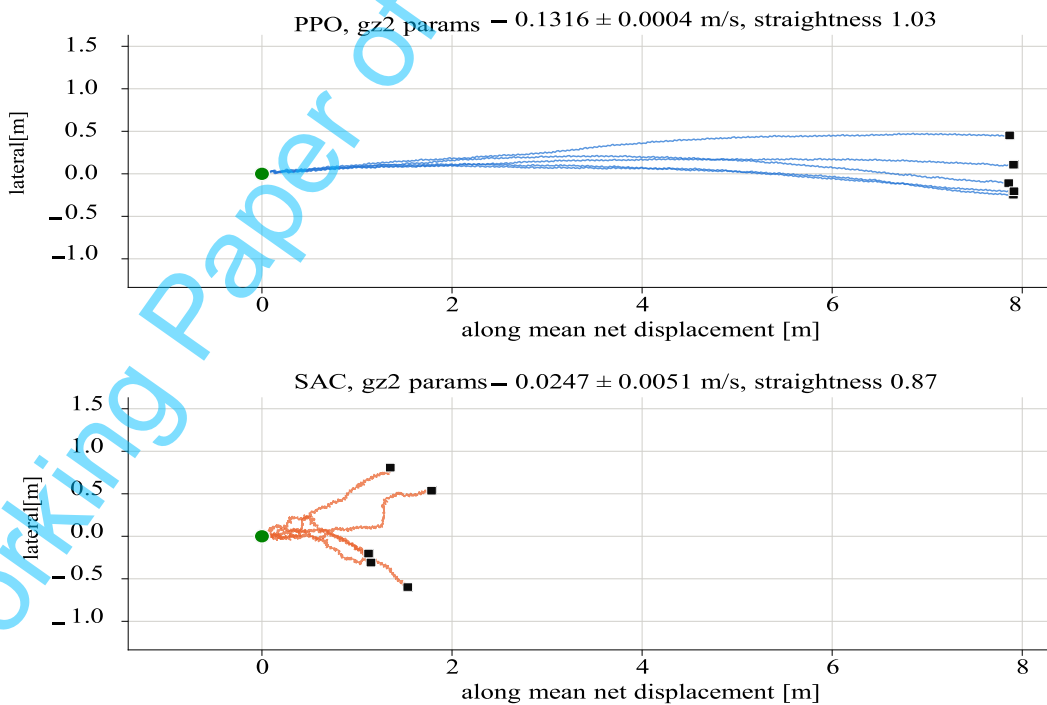


Fig. 2. Planar Gazebo trajectories using the calibrated execution parameters. PPO is shown above and SAC below.

Condition	v [m/s]	Straightness	Yaw drift [°/s]
PPO, Isaac par.	0.1218 ± 0.0374	0.99 ± 0.08	-3.11 ± 1.57
PPO, cal. par.	0.0416 ± 0.0029	0.94 ± 0.05	-0.10 ± 0.09
SAC, Isaac par.	0.0203 ± 0.0086	0.51 ± 0.29	$+1.18 \pm 1.90$
SAC, cal. par.	0.0118 ± 0.0055	0.50 ± 0.14	-0.73 ± 0.25

Tab. 3. Trajectory metrics on the physical robot over five runs per condition.

4.3 Transfer to the Physical Robot

Tab. 3 summarises the trajectory metrics of the twenty hardware runs, and Fig. 3 shows the corresponding paths. Forward velocity is evaluated over the walking interval. The effect of the two execution-parameter sets on forward velocity is opposite to the result in Gazebo. PPO reaches 0.1218 m/s with the original Isaac Lab parameters and 0.0416 m/s with the Gazebo-calibrated parameters. Changing only the execution configuration therefore changes the PPO hardware velocity by a factor of 2.93. For SAC, the corresponding factor is 1.72. The higher PPO velocity does not correspond to a regular gait. Across both execution configurations, PPO reaches between 3.5 and 6.0 times the forward velocity of SAC. Its straightness lies between 0.94 and 0.99, compared with approximately 0.50 for SAC. Although the calibrated configuration reduces forward velocity on the physical robot, it improves the other measured motion characteristics. PPO body oscillation decreases from 0.789 to 0.089 rad/s and SAC body oscillation from 0.756 to 0.136 rad/s.

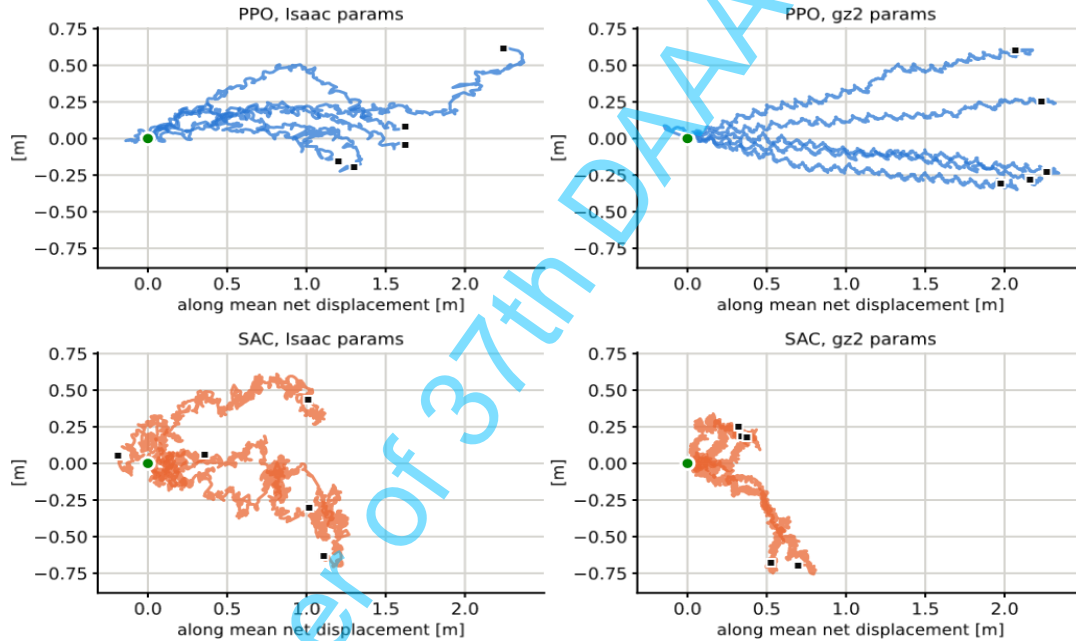


Fig. 3. Planar hardware trajectories for both execution parameter sets over five repetitions. PPO is shown above and SAC below, using identical axes.

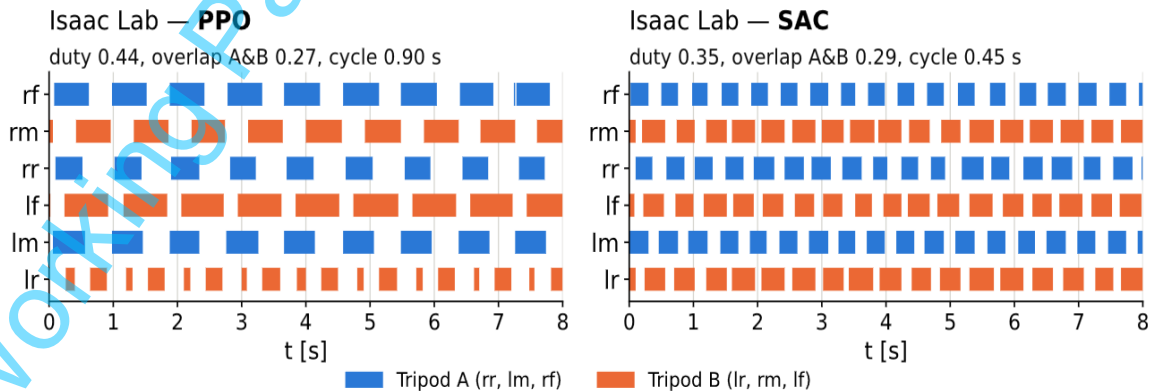


Fig. 4. Gait diagrams of the selected checkpoints in Isaac Lab. Tripod A contains rr, lm and rf, while tripod B contains lr, rm and lf.

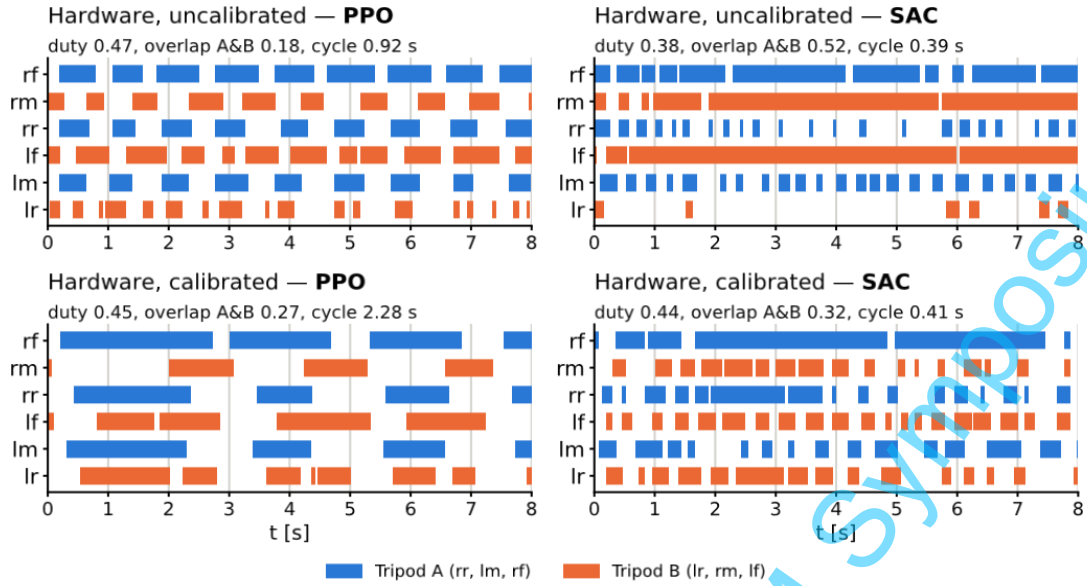


Fig. 5. Gait diagrams on the physical robot for both execution parameter sets using the encoding of Fig. 4.

4.4 Gait Structure and Observation Path

Tab. 4 summarises the velocity ratio and body oscillation across the transfer chain. The gait analysis ranks the algorithms differently from the velocity-based training results. In Isaac Lab (Fig. 4, previous page), SAC is closer to the ideal alternating tripod pattern, with a phase offset of 159.2° compared with 135.7° for PPO. At the same time, SAC uses a considerably higher step frequency of 1.987 Hz versus 1.127 Hz for PPO. Outside Isaac Lab, foot contacts are not measured directly but reconstructed from joint angles. The direct transfer to Gazebo disrupts the alternating pattern for both policies. The hardware gait diagrams in Fig. 5 show that PPO also loses a regular alternating tripod under the original Isaac Lab parameter set. The deviation from the ideal tripod phase reaches 110.0° . Video recordings indicate that forward motion in this condition results from irregular pushing rather than coordinated alternating support. Despite its lower speed, the calibrated motion is more regular, with the phase deviation decreasing to 88.4° .

Five stationary recordings of 30 s characterise the hardware observations. Within a run, the standard deviations of angular velocity, projected gravity, and joint position remain below the training noise of $\sigma = 0.01$. Across runs, however, the projected-gravity components vary by 6.4 to 9.0 times the within-run variation, indicating that the dominant difference is a run-dependent offset rather than short-term random noise.

4.5 Discussion

The results lead to three main findings. First, the two stages of the transfer chain cannot be treated as equivalent, and the dominant loss depends on both the algorithm and the execution configuration. When the Gazebo-calibrated configuration is followed through the downstream chain, PPO loses 68.4% of its velocity ratio between Gazebo and hardware, whereas SAC loses 75.6% between Isaac Lab and Gazebo. Reporting only the final sim-to-real ratio would hide this difference. The behaviour changes again when the original Isaac Lab parameters are kept fixed. Under this configuration, the velocity ratio increases from Gazebo to hardware, from 0.105 to 1.218 for PPO and from 0.126 to 0.203 for SAC. For forward velocity alone, the simulator-to-simulator transfer is therefore more detrimental than the subsequent transfer to hardware under the unchanged parameter set.

		Isaac Lab	Gazebo		Hardware	
		Isaac par.	Isaac par.	cal. par.	Isaac par.	cal. par.
ρ	PPO	1.230	0.105	1.316	1.218	0.416
	SAC	1.013	0.126	0.247	0.203	0.118
ω_{rms} [rad/s]	PPO	0.101	0.532	0.029	0.789	0.089
	SAC	0.066	0.589	0.082	0.756	0.136

Tab. 4. Velocity ratio and body oscillation along the transfer chain. Bold values mark the better execution parameter set within each domain. Neither metric describes gait quality.

Second, the execution path is a major experimental factor. The policy network remains unchanged in every condition, yet altering the execution parameters changes the hardware velocity by as much as a factor of 2.93. On hardware, this difference is produced solely by the output smoothing and rate-limit settings, because the servo position loop is closed internally; consequently, the derivative-gain argument below applies to the Gazebo results only. A parameter set calibrated in one domain can therefore reduce performance after transfer to another. The same effect is visible in the gait structure: on hardware, the calibrated PPO configuration reduces the tripod phase deviation from 110.0° to 88.4° and lowers body oscillation, even though forward velocity decreases. The execution configuration thus affects not only how fast the robot moves but also how coordinated that motion is. In Gazebo, the substantially higher derivative gain of the uncalibrated configuration produces stronger counteracting torques during fast joint motion; combined with the limited torque of the AX-12A servos, this provides a plausible explanation for the poor Gazebo behaviour under the original settings.

Third, strong training-domain metrics do not reliably predict transfer performance. SAC reaches the velocity thresholds with fewer or the same number of environment steps and tracks the commanded speed more accurately in Isaac Lab, but neither advantage persists along the transfer chain. The threshold crossing characterises adaptation speed after a curriculum change; it does not measure how accurately the command is maintained thereafter. In Isaac Lab, SAC's gait cycle is almost twice as fast as that of PPO, and its joint targets change more abruptly. These commands are later executed by torque-limited servos with an effective update rate of only 16.7 Hz. The larger joint-tracking error and the five nearly stationary joints observed in Gazebo are consistent with this mismatch. Algorithms and execution path are therefore not independent factors. PPO and SAC respond differently to the same parameter change, with hardware velocity factors of 2.93 and 1.72, respectively, and the substantially larger calibration effort required for SAC is consistent with a narrower region of viable downstream parameters. The learning algorithm thus influences how tolerant the resulting policy is to the execution system.

Four limitations apply to these findings. First, only one checkpoint per algorithm is transferred through the complete chain. Second, the number of repetitions per condition is small. Third, several execution parameters are changed simultaneously during calibration. Fourth, the observation randomisation used during training represents short-term, zero-mean noise but does not reproduce the run-dependent sensor offsets observed on the physical robot. Future work will increase the number of independent training seeds and hardware evaluation trials per condition to enable formal statistical hypothesis testing.

5. Summary and outlook

Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) were trained for straight-line locomotion of the PhantomX AX Hexapod Mark II under the same task, reward function, hidden-layer sizes, and velocity curriculum, and were then evaluated separately in Isaac Lab, Gazebo, and on the physical robot. In the training simulator, SAC reached both velocity thresholds in fewer or the same number of environment steps and followed the commanded velocity more accurately. However, its training required roughly one third more wall-clock time, and the spread of its final episodic return across seeds was more than twenty times larger than for PPO. Neither training-domain advantage persisted after transfer.

On the physical robot, PPO reached between 3.5 and 6.0 times the forward velocity of SAC. Along the calibrated transfer path, the largest PPO loss was 68.4% between Gazebo and hardware, whereas the largest SAC loss was 75.6% between Isaac Lab and Gazebo. Changing only the execution parameters altered the hardware velocity by up to a factor of 2.93. In Isaac Lab, SAC started closer to the ideal tripod pattern, with a deviation of 20.8° compared with 44.3° for PPO, but it lost the alternating contact pattern on hardware under both execution configurations.

The execution parameters should be varied individually to isolate the contributions of controller gains, output smoothing, and rate limiting. For SAC, adding an explicit penalty on large action changes during training should be investigated to reduce commands that the physical servos cannot track. The actuator model should also be adapted more closely to the AX-12A dynamics, including a realistic torque limit, as inaccuracies in actuator modelling are a known source of sim-to-real error.

6. References

- [1] Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V. & Hutter, M. (2020). Learning quadrupedal locomotion over challenging terrain, *Science Robotics*, Vol. 5, No. 47, October 2020, p. eabc5986, DOI: 10.1126/scirobotics.abc5986.
- [2] Rudin, N.; Hoeller, D.; Reist, P. & Hutter, M. (2022). Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning, *arXiv preprint arXiv:2109.11978*, August 2022, Available from: <https://arxiv.org/abs/2109.11978> Accessed: 2026-09-22.
- [3] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A. & Klimov, O. (2017). Proximal Policy Optimization Algorithms, *arXiv preprint arXiv:1707.06347*, August 2017, Available from: <https://arxiv.org/abs/1707.06347> Accessed: 2026-09-22.

- [4] Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P. & Levine, S. (2019). Soft Actor-Critic Algorithms and Applications, arXiv preprint arXiv:1812.05905, January 2019, Available from: <https://arxiv.org/abs/1812.05905> Accessed: 2026-09-22.
- [5] Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, C. D.; Tsounis, V.; Koltun, V. & Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots, *Science Robotics*, Vol. 4, No. 26, January 2019, p. eaau5872.
- [6] Peng, B.; Andrychowicz, M.; Zaremba, W. & Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization, *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 3803-3810, IEEE.
- [7] Stuja K.; Bruqi, M.; Markl E. & Aburaia, M. (2016). Lightweight 4-Axis Scara Robot for Education and Research, *Proceedings of the 27th DAAAM International Symposium*, pp. 0102-0108, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-08-2, ISSN 1726-9679, Vienna, Austria.
- [8] Sicha, M.; Stuja, K.; Aburaia, M. & Aburaia, A. (2025). Design and Implementation of an Additively Manufactured, Aluminium-Reinforced 4-Axis Pick-and-Place Robot, *Proceedings of the 36th DAAAM International Symposium*, pp. 0293-0297, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-47-1, ISSN 1726-9679, Vienna, Austria.
- [9] Kim, T.; Choi, M.; Choi, S.; Yoon, T. & Choi, D. (2026). Deep Reinforcement Learning Control of a Hexapod Robot, *Actuators*, Vol. 15, No. 1, January 2026, p. 33.
- [10] Liu, X.; Wu, J.; Li, Y.; Qi, C.; Xue, Y. & Gao, F. (2025). Learning Natural and Robust Hexapod Locomotion over Complex Terrains via Motion Priors based on Deep Reinforcement Learning, arXiv preprint arXiv:2511.03167, November 2025, Available from: <https://arxiv.org/abs/2511.03167> Accessed: 2026-09-22.
- [11] Wang, L.; Li, R.; Wang, X.; Gao, W. & Chen, Y. (2025). A Motion Control Strategy for a Blind Hexapod Robot Based on Reinforcement Learning and Central Pattern Generator, *Symmetry*, Vol. 17, No. 7, July 2025, p. 1058.
- [12] Sabatini, G.; Li, C. & Hutter, M. (2026). Bridging the Gap: Enabling Soft Actor Critic for High Performance Legged Locomotion, arXiv preprint arXiv:2605.24975, May 2026, Available from: <https://arxiv.org/abs/2605.24975> Accessed: 2026-09-22.
- [13] ROBOTIS Co., Ltd. (2026). AX-12A, Available from: <https://docs.robotis.com/docs/dxl/model-reference/ax-series/ax-12a/> Accessed: 2026-08-13.
- [14] Mittal, M.; Roth, P.; Tigue, J.; Richard, A.; Zhang, O.; Du, P.; Serrano-Muñoz, A.; Yao, X.; Zurbrugg, R.; Rudin, N.; Wawrzyniak, L.; Rakhsha, M.; Denzler, A.; Heiden, E.; Borovicka, A.; Ahmed, O.; Akinola, I.; Anwar, A.; Carlson, M. T.; Feng, J. Y.; Garg, A.; Gasoto, R.; Gulich, L.; Guo, Y.; Gussert, M.; Hansen, A.; Kulkarni, M.; Li, C.; Liu, W.; Makoviychuk, V.; Malczyk, G.; Mazhar, H.; Moghani, M.; Murali, A.; Noseworthy, M.; Poddubny, A.; Ratliff, N.; Rehberg, W.; Schwarke, C.; Singh, R.; Smith, J. L.; Tang, B.; Thaker, R.; Trepte, M.; Van Wyk, K.; Yu, F.; Millane, A.; Ramasamy, V.; Steiner, R.; Subramanian, S.; Volk, C.; Chen, C. Y.; Jawale, N.; Varghese Kuruttukulam, A.; Lin, M. A.; Mandlekar, A.; Patzwaldt, K.; Welsh, J.; Zhao, H.; Anes, F.; Lafleche, J.-F.; Moënnelocoz, N.; Park, S.; Stepinski, R.; Van Gelder, D.; Amevor, C.; Carius, J.; Chang, J.; Chen, A. H.; de Heras Ciechowski, P.; Daviet, G.; Mohajerani, M.; von Muralt, J.; Reutsky, V.; Sauter, M.; Schirm, S.; Shi, E. L.; Terdiman, P.; Vilella, K.; Widmer, T.; Yeoman, G.; Chen, T.; Grizan, S.; Li, C.; Li, L.; Smith, C.; Wiltz, R.; Alexis, K.; Chang, Y.; Chu, D.; Fan, L. J.; Farshidian, F.; Handa, A.; Huang, S.; Hutter, M.; Narang, Y.; Pouya, S.; Sheng, S.; Zhu, Y. (2025). Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning, arXiv preprint arXiv:2511.04831, November 2025, Available from: <https://arxiv.org/abs/2511.04831> Accessed: 2026-09-22.
- [15] Serrano-Munoz, A.; Chrysostomou, D.; Bøgh, S. & Arana-Arexolaleiba, N. (2023). skrl: Modular and flexible library for reinforcement learning, *Journal of Machine Learning Research*, Vol. 24, No. 254, pp. 1-9, 2023, Available from: <https://jmlr.org/papers/v24/23-0112.html> Accessed: 2026-09-22.
- [16] Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D. & Meger, D. (2018). Deep Reinforcement Learning That Matters, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, No. 1, pp. 3207-3214, April 2018.
- [17] Koenig, N. & Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator, *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 3, Sendai, Japan, 2004, pp. 2149-2154, IEEE.