Zhang, Zhelin; Liu, Tie; Ding, Liang; Wang, Haoyu; Xu, Peng; Yang, Huaiguang; Gao, Haibo; Deng, Zongquan; Pajarinen, Joni

Imitation-Enhanced Reinforcement Learning With Privileged Smooth Transition for Hexapod Locomotion

Published in:
IEEE Robotics and Automation Letters

DOI:
10.1109/LRA.2024.3497754

Published: 01/01/2025

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Zhang, Z., Liu, T., Ding, L., Wang, H., Xu, P., Yang, H., Gao, H., Deng, Z., & Pajarinen, J. (2025). Imitation-Enhanced Reinforcement Learning With Privileged Smooth Transition for Hexapod Locomotion. *IEEE Robotics and Automation Letters*, *10*(1), 350-357. https://doi.org/10.1109/LRA.2024.3497754

Imitation-Enhanced Reinforcement Learning with Privileged Smooth Transition for Hexapod Locomotion

Zhelin Zhang[1], Tie Liu[1], Liang Ding[1,*], *Senior Member, IEEE*, Haoyu Wang[1], Peng Xu[1,*], Huaiguang Yang[1], Haibo Gao[1], Zongquan Deng[1], and Joni Pajarinen[2]

***Abstract*—Deep reinforcement learning (DRL) methods have shown significant promise in controlling the movement of quadruped robots. However, for systems like hexapod robots, which feature a higher-dimensional action space, it remains challenging for an agent to devise an effective control strategy directly. Currently, no hexapod robots have demonstrated highly dynamic motion. To address this, we propose imitation-enhanced reinforcement learning (IERL), a two-stage approach enabling hexapod robots to achieve dynamic motion through direct control using RL methods. Initially, imitation learning (IL) replicates a basic positional control method, creating a pre-trained policy for basic locomotion. Subsequently, the parameters from this model are utilized as the starting point for the reinforcement learning process to train the agent. Moreover, we incorporate a smooth transition (ST) method to make IERL overcome the changes in network inputs between two stages, and adaptable to various complex network architectures incorporating latent features. Extensive simulations and real-world experiments confirm that our method effectively tackles the high-dimensional action space challenges of hexapod robots, significantly enhancing learning efficiency and enabling more natural, efficient, and dynamic movements compared to existing methods.**

***Index Terms*—Hexapod robot, Imitation learning, Reinforcement learning, Locomotion control.**

I. Introduction

ROBUST and efficient locomotion remains a pivotal area of research in legged robotics. Deep reinforcement learning (DRL) has proven to be immensely beneficial in addressing these intricate control challenges. Numerous studies employing DRL have empowered quadruped robots with exceptional locomotive skills [1], enabling stable operations in outdoor environments [2], enhanced movement speed [3] [4] and adaptability to unfamiliar environments [5] [6] [7]. Furthermore, RL has allowed robots to traverse complex terrains, surmount significant obstacles [8], and perform impressive parkour maneuvers [9] [10] [11].

To achieve remarkable and unforeseen movements, these approaches generally avoid relying on pre-programmed gait patterns or reference motions. Instead, they utilize neural networks to directly generate the desired actions for each actuated joint. The advantage of this approach is evident: it provides the agent with unprecedented autonomy by bypassing limitations associated with manually crafted constructs such as specific foot trajectories or gait patterns, thus enhancing the learning potential of the agent. Predefined gait patterns or reference motions might not even generalize effectively across obstacles of reasonable heights [12]. However, as the number of action dimensions in the output grows, learning complexity skyrockets, making it challenging to apply current methods to legged robots with higher degrees of freedom, like hexapods.

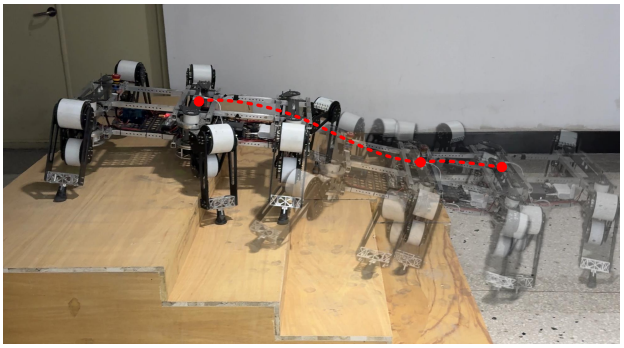

Fig. 1. Using Imitation-Enhanced Reinforcement Learning to learn a robust motion strategy for a hexapod robot to climb stairs.

In the domain of hexapod robot control using reinforcement learning (RL), to simplify the learning process, nearly all existing strategies [13] integrate model-based control techniques with RL. For example, some methods [14] combine Deep Deterministic Policy Gradient (DDPG) with Central Pattern Generators (CPG), while others rely on manually crafted foot trajectories [15] or gait patterns [16]. Almost all of this research points out that with six legs with 3 links each, the hexapod robot actions have 18 dimensions, resulting in an excessively large exploration space. Additionally, the six legs must operate in a coupled and coordinated fashion. As a result, relying purely on RL to advance locomotion in hexapod robots has largely failed to produce effective policies. Consequently, most methods [17] [16] [18] refrain from using neural networks to directly control all the joints of a hexapod robot. Attempts to train direct joint control typically lead to unstable, uncoordinated, and unnatural walking patterns, as the models often settle on suboptimal solutions, resulting in awkward and inefficient movement. Given the motion control challenges of hexapod robots, end-to-end RL generally incorporates simplifications. So far, no method has enabled hexapod robots to learn control of all joints directly and achieve the impressive motion control seen in quadruped robots.

This paper was recommended for publication by Editor Abderrahmane Kheddar upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the NationaKey RD Program of China (2022YFB4702300); the National Natural Science Foundation of China (Grant No.52205011); the Fundamental Research Funds for the Central Universities (FRFCU9803500621, No. HIT. OCEF. 2023042); the National Natural Science Foundation of China (Grant No. 91948202); the Heilongjiang Postdoctoral Fund under Grant LBH-Z20136.

[1] The State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150001, China.

[2] Department of Electrical Engineering and Automation, Aalto University, Finland.

Video materials: https://www.youtube.com/watch?v=8hiOxqW0Vog.

Fig. 2. We tested the performance of the hexapod in a variety of environments, indoor and outdoor, and our method showed good performance. The hexapod robot moves robustly and efficiently with a periodic gait.

We propose a two-stage learning method to effectively address control problems with high-dimensional output action spaces. In the first stage, we pre-train a model to imitate an expert position control strategy. Unlike typical imitation learning, we do not require the pre-trained model to exhibit high performance or consider any forgetting issues. Its objective is merely to establish basic, plausible locomotion patterns, so minor compounding errors are deemed acceptable and not detrimental. In the second stage, the parameters of the pre-trained model are used as initial parameters for RL. Since the pre-trained model can already perform simple movements, the agent can quickly obtain velocity rewards and master various skills based on the task demands of reinforcement learning.

However, a significant challenge emerges when employing complex network structures like the Teacher-Student model for RL training [2] [6] [19]. In this setup, the main network processes latent feature vectors that do not directly correspond to physical attributes, making it difficult to extract actionable data for behavioral cloning from an expert policy. To address this issue, we introduce a smooth transition method that involves automatic updates to the weights of the feature vectors. This strategy refines the two-stage learning process, making it suitable for use with sophisticated reinforcement learning models. The model's input can seamlessly shift from predefined data to the latent feature vectors produced by various encoders. This enables the agent to effectively utilize the pre-trained knowledge while smoothly transitioning to the demands of the reinforcement learning task.

To demonstrate the advantages of our approach, we conduct tests both in simulations and on actual hardware. Our method facilitates robust and omnidirectional locomotion across diverse terrains. Compared to the previous method, our approach doubles the speed and allows climbing over 13cm obstacles. Notably, the locomotion behaviors are optimized compared to the initial imitation, showing a novel and natural gait. We summarize our contributions as follows:

- We introduce the IERL framework for training control strategies for hexapod robots to maximize the motion potential. This framework effectively addresses the issue of uncoordinated motion caused by the high-dimensional action space of hexapod robots, mitigating the sensitivity to reward weights during the initial stages of training.
- Addressing the issue of latent feature vectors corresponding to privileged observations in complex neural network structures, we propose a smooth transition method that automatically interpolates between behavior cloning and reinforcement learning observation features, enabling effective behavioral cloning without the need for real physical quantities corresponding to latent features.
- We employed reinforcement learning methods without traditional models to control a real hexapod robot and achieved a maximum moving speed of 2m/s. As far as we know, this is the fastest speed attained by a similarly sized hexapod robot, demonstrating a highly dynamic gait and robust locomotion on various terrains.

II. Related Work

A. RL approaches for Legged Robots

The reinforcement learning control methods for quadruped robots are well-established, and the introduction of privileged learning has enabled these robots to master various advanced skills, such as blind locomotion on uneven terrain [2], increasing maximum speed [3] [4], robust adaptive locomotion [6], and complex parkour movements [9] [10] [11]. These remarkable achievements are based on the same privileged learning

concept of teacher-student [2] or learning by cheating [19]. However, direct and brute-force learning approaches are not suitable for hexapod robots due to their unique configuration and high-dimensional action control.

Currently, most RL control methods applied to hexapod robots involve a combination of RL and model-based approaches to reduce the learning complexity such as combining SNN with CPG [13], DDPG with CPG [14], manually designing foot trajectories and gaits [15] or distributed approach [17]. Konen et al. [16] [18] have pointed out that locomotion in hexapod robots is a high-dimensional combination problem, and as the joint space increases, learning gaits becomes more difficult, potentially leading to asymmetric gaits or ignoring one or more joints.

B. Imitation based policy warmup

Motion control methods based on imitation learning are typically directly related to reference motion data. Some work [20] [21] [22] use model-based control methods as the initialization for control strategies, while others [23] [24] [25] attempt to mimic various motion patterns or real animal behavior to develop optimal strategies. Several studies [20] [21] [23] [24] [25] [26] [27] [28] adopt IL and RL-centered multi-stage learning frameworks, though their network architectures are relatively simple. In these studies, the expert strategy being imitated significantly impacts the final control strategy. Currently, there is little work providing feasible pre-trained models for privileged learning.

C. Adaptive fine-tuning in multi-stage learning

Some work [26] [28] design a motion adaptor or joint mapping to overcome the differences between real animal and robot joint positions, providing effective strategies for reinforcement learning. Others [25] [26] incorporate a reward for following the reference motion during RL training and fine-tuning the transition by adjusting the weights of imitation and task rewards. [27] introduced an adjustable weight to control terrain adaptation's impact on motion. Additionally, [4] uses a relaxation step to reach a global optimal solution.

III. METHOD

As outlined in Fig. 3, our approach imparts basic locomotion skills to the hexapod robot via imitation learning, establishing a solid base policy. We then use a privileged learning framework to optimize locomotion across various scenarios by incorporating multiple encoders for accurate state estimation, terrain recognition, and self-state recognition. Additionally, our smooth transition method with self-updating parameters ensures seamless transitions between stages, mitigating the interference from non-realistic physical data in imitation learning. By integrating imitation learning with RL, we leverage human expertise and the robot's autonomous learning, achieving superior motion control and providing a stable, reliable solution for hexapod robot locomotion.

A. Imitation learning pre-training

1) Imitation Learning: In this stage, we employ imitation learning to mimic a simple model, creating a pre-trained model for basic walking. Using a basic positional controller as the reference policy, we plan periodic foot trajectories based on the common tripod gait.

$$p_{com}^{T} = p_{com} + R_B^W \cdot v_{cmd} \cdot (T_{sw} - t) \tag{1}$$

$$\psi^T = \psi + w_{cmd} \cdot (T_{sw} - t) \tag{2}$$

$$p_{hip,i}^{T} = p_{com}^{T} + R_z(\psi^T) \cdot p_{hip,i} \tag{3}$$

Where p_{com}^T and p_{com} are target and actual base positions, ψ^T and ψ are target and actual base yaw angles, v_{cmd} and w_{cmd} are linear velocity, and angular velocity commands. R_B^W is the transformation matrix from the body to the world coordinate system and $R_z(\alpha)$ is a rotation matrix that rotates α around the z-axis. T_{sw} is the duration of the leg swing cycle, which is set to 0.5s. Then obtain the target positions of the leg bases $p_{hip,i}^T$ in the world coordinate system.

$$\Delta p_1 = R_B^W \cdot v_{cmd} \cdot \frac{T_{st}}{2} \tag{4}$$

$$\Delta p_2 = R_z(\psi^T) \cdot [R_z(w_{cmd} \cdot \frac{T_{st}}{2}) \cdot p_{hip,i} - p_{hip,i}] \tag{5}$$

$$p_{foot,i}^{T} = p_{hip,i}^{T} + \Delta p_1 + \Delta p_2 \tag{6}$$

Where Δp_1 and Δp_2 represent the translational and rotational components of the foot, respectively. T_{st} is the duration of the leg standing cycle and $p_{foot,i}^T$ represents the expected landing positions. We use a sixth-degree polynomial to plan the foot trajectory from the current to the desired position. The step size adjusts based on the speed command, with a maximum linear velocity of 0.8 m/s and a maximum angular velocity of 1 rad/s. For effective cloning, the cycle is fixed at 1 second. Joint angles are solved via inverse kinematics and tracked using PD control, without feedforward torques.

The positional control policy is deployed in simulation, with control commands randomly sampled and data recorded in 20-second intervals. A simple behavioral cloning approach is then applied to optimize a loss function, converting the expert policy into a neural network policy.

$$Loss = \sum_{obs \in D_E} \left\| \pi^E(obs) - \pi^T(obs) \right\|^2 \tag{7}$$

Where D_E is the collected expert data, which includes the sequences of states and corresponding actions taken by the expert. π^E denotes the expert policy, π^T is the target policy. Using behavioral cloning, we optimize the loss function to minimize the difference between the actions predicted by the target policy π^T and the actions taken by the expert policy π^E in the collected data D_E. This process involves training a neural network to approximate the mapping from states to actions that the expert employs.

Unlike most current approaches, the significance of the pre-trained model lies primarily in helping the agent quickly acquire velocity rewards during the RL stage, avoiding poor local optima. Therefore, the performance of the imitation policy is not critical, almost all motor optimizations occur during reinforcement learning. The pre-trained policy only needs to generate effective movements in most cases.

2) Network architecture: The control policy's network architecture is a Multi-Layer Perceptron (MLP) with three hidden

layers of 512, 256, and 128 neurons. In our approach, we define the observation space of the base network as consisting of three components: base observation information, explicit privileged information, and latent privileged information.

$$Obs = [obs_b, obs_e, obs_p] \tag{8}$$

The base observation information obs_b, like most current methods, refers to conventional sensor data, including base quaternions, measurement of the gravity vector, linear-velocity and angular-velocity commands, joint positions and velocities, the previous actions selected by the policy and a phase of a sine signal with a period of 1 second. This phase information is discarded during the RL stage (set to 0), but this does not cause training failure. Instead, it encourages the policy to explore more suitable motion cycles. Explicit privileged information obs_e is defined as the body state information output from a state estimation module, including the linear velocities and angular velocities of the hexapod. However, in the imitation learning stage, real simulation feedback data is used as a temporary substitute of obs_e due to the lack of a state estimation module. Latent privileged information obs_p refers to the vector output by the privileged information encoders E_p and E_t. However, it is crucial to note that these potential feature vectors cannot have explicit corresponding physical interpretations. Consequently, we are unable to directly acquire these data from the simulator during the imitation learning stage, nor can we construct the privileged information encoders at this stage. This impediment precludes us from directly fitting a base policy. To address this issue, we adopt a straightforward workaround: using a zero vector as a placeholder in this position, meaning that the input at this location (obs_p) during the imitation learning process will always be zero. This approach is both simple and safe, and the network will gradually learn to ignore this portion without affecting further training.

The action space is defined as the desired positions for all joints. This end-to-end network structure directly controls all motors, enabling the agent to fully realize its learning potential without constraints from manually designed models.

B. Reinforcement learning optimization

1) Reinforcement learning: At this point, we have obtained a pre-trained model for the base policy, but this model merely mimics the reference motion in a basic manner, and sometimes the actual motion effect is even worse than the reference. In this section, we further optimize the pre-trained model using RL methods. In fact, most of the robot's skills are acquired during this stage, including the optimization of motor behaviors, the learning of state estimation modules, and the training of privileged encoders. We define the problem as a Markov Decision Process (MDP), represented as a tuple:

$$\langle S, A, P, r, \gamma \rangle \tag{9}$$

The pre-trained model provides a solid starting point, allowing the agent to quickly obtain significant rewards to optimize the control policy. More importantly, it guides the agent towards reasonable motion performance.

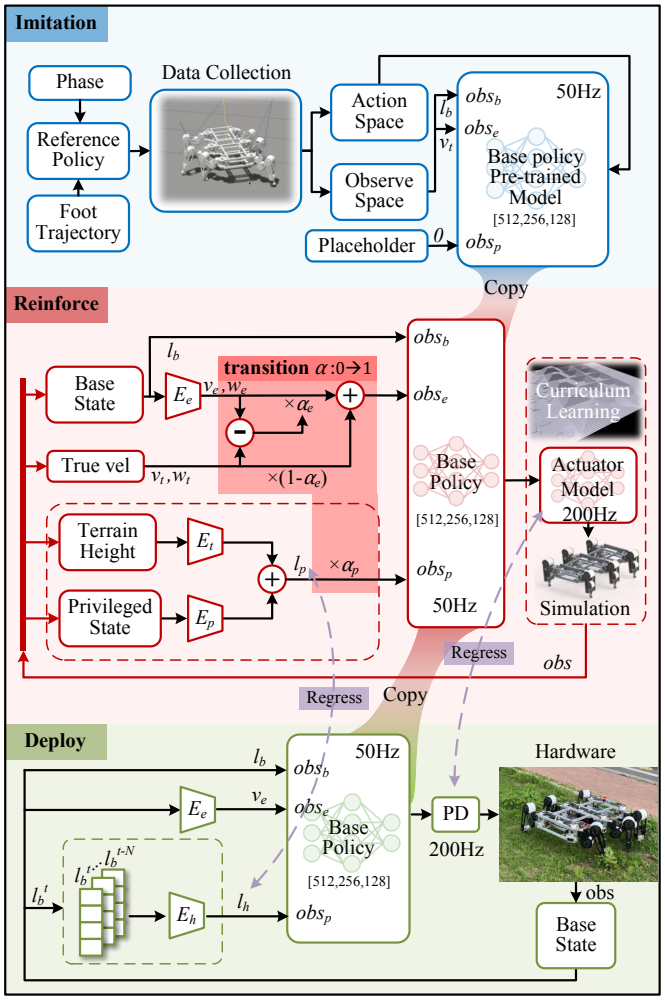

Fig. 3. A pre-trained base policy is constructed through imitation learning. In the RL stage, this model is enhanced with various encoders for privileged learning, with input weights automatically adjusted to ensure seamless data transition. The final student policy is then deployed on the physical robot.

To demonstrate the superiority of our method, we employ a multi-modular reinforcement learning approach with a relatively complex network architecture. This architecture comprises the base network along with several smaller encoder networks, such as the state estimation encoder.

2) Strategy architecture: As shown in Fig. 3, the teacher policy includes four networks: the base policy, state estimation encoder E_e, terrain information encoder E_t, and privileged information encoder E_p. In the student policy, E_t and E_p are replaced by a historical information encoder E_h.

We utilize the pre-trained model obtained from the previous stage as the initial solution for the base network, which maintains the same structure as in the first stage. Its observation space comprises three components: obs_b, obs_e, and obs_p, as mentioned earlier. The action space corresponds to the desired joint positions for the 18 actuated joints, which are then converted into torques using a PD controller.

Considering the high cost of precise state estimation methods, we incorporate a state estimation module, an MLP with layers of size [128, 64], based on supervised learning. This module takes the base state information as input and computes the robot's linear and angular velocities from its own actions.

We use the actual speed returned by the simulator as the label and train this module through supervised learning while training the base policy. In our tests, this method performed significantly better than the state estimation method based on the fusion of IMU and leg odometry, as the learned control policy does not have a fixed leg lift height and lacks contact detection, especially on rough terrain where model-based methods often fail.

Multiple privileged encoder modules are implemented as MLPs with sizes of [128,64,20], compressing additional privileged information into feature vectors for the base policy. The Proximal Policy Optimization (PPO) algorithm simultaneously trains these privileged encoder modules and the base network just like [3] and [6].

In the student policy, we utilize the historical information from the most recent 20 steps (with 0.02s intervals) as the input to identify privileged information. This encoder is implemented as a Temporal Convolutional Network (TCN) with two 1D convolutional layers followed by activation functions, and a final flattening layer. E_h is trained through supervised learning to minimize the difference between the two feature vectors l_p and l_h. When this loss is sufficiently small, it is considered that the agent has successfully identified the privileged information from the historical information of the robot's body. The final student policy is then deployed on the actual robot to achieve the learned advanced skills.

C. Smooth transition method

Due to the links between multiple networks, a smooth transition between imitation learning and reinforcement learning for the control strategy is crucial for the successful implementation of this method. The main issue arises from the changes in the input to the pre-trained network. During the transition between the pre-training and optimization stages, the inputs obs_e and obs_p of the pre-trained network are replaced. Specifically, obs_e is swapped from real observation data to the velocity information output by the state estimation module, and obs_p is replaced from a zero vector to the feature vector l_p output by the privileged information encoding module. Without a smooth transition method, this abrupt change can lead to the failure of training and prevent the correct inheritance of skills learned through imitation learning.

Considering the differences between these two sets of data, we designed two distinct smoothing strategies. Since obs_p, which is the combined output of the privileged encoder and terrain encoder, cannot have explicit corresponding physical interpretations and the pre-trained model only receives a zero vector in that position, a simple linear approach is sufficient for smoothing the transition. The input weight for $obsp$ gradually increases from 0 to 1 as training advances.

$$obs_p = \alpha_p\left(t\right) \cdot l_p \tag{10}$$

The learning speed of these encoder modules is quite fast, thus this instable stage occurs only in the early stages of the RL stage. Consequently, it does not have a significant negative impact on motion control during the RL stage.

As for the obs_e output from the state estimation encoder, in the early stages of RL, the state estimation module is untrained and its output is uncontrollable. However, unlike obs_p, the data in obs_e has physical significance, representing the velocity and angular velocity of the robot. Fortunately, we can still obtain this data from the simulator. Therefore, we design obs_e as a mixed input of the state estimation module and the real information. Gradually, as the training progresses, we increase the weight of the state estimation module until it completely replaces the mixed information.

$$obs_e = \alpha_e \cdot (v_e, w_e) + (1 - \alpha_e) \cdot (v_t, w_t) \tag{11}$$

This approach effectively addresses the issue of data mutation during the RL stage. To minimize the difference between the mixed output and the real information, α_e does not rely on a simple linear weight that increases with training time. Instead, it employs an automatic updating rule to adjust the weight, thereby enhancing learning efficiency and reducing manual effort.

$$\alpha_e = \begin{cases} 1, & e_{\mathrm{MSE}} < 0.1 \\ \frac{1}{1+e^{10(e_{\mathrm{MSE}}-0.55)}}, & 0.1 \le e_{\mathrm{MSE}} \le 1 \\ 0, & e_{\mathrm{MSE}} > 1 \end{cases} \tag{12}$$

In this context, e_{MSE} represents the difference between the estimated information and the real information, which is expressed using the mean squared error. By adopting this weight adjustment method, the pre-trained model can smoothly transition between the two stages. Following this approach, our method can be applied to almost all RL architectures based on the teacher-student paradigm, enabling a seamless change for various types of data. A rare problem is that when the initial action standard deviation introduced by PPO for random actions is too large, it can also lead to training failure. In this method, the initial standard deviation is set to 1.

IV. Training

In this study, we use an insect-like hexapod robot with 18 independently motor-controlled joints to achieve basic velocity command tracking. Apart from the integrated motors, the robot only carries an IMU for body pose feedback.

Simulation: We utilized NVIDIA's ISAAC Gym simulator for training [29], which greatly reduced the training time due to its large-scale parallel training approach, and all training was conducted on a single NVIDIA RTX 4090 GPU. The simulation time step is 0.005 s. The network operates at 50 Hz, while the PD controller is set to 200 Hz.

Actuator modeling: In simulations, the PD controller was overly idealistic, widening the sim-to-real gap. Additionally, deploying the control policy on actual robots required extensive parameter tuning. To mitigate these issues, we modeled the drive motors by analyzing historical position errors and velocities to predict output torque. Unlike previous methods [1] that modeled each motor separately, we use a single 2-layer MLP with 32 neurons per layer to model all three drive motors on one leg of the hexapod robot. This approach addresses the significant leg inertia associated with the hexapod's unique joint configuration. We observed notable load disparities among drive motors at different positions, contributing to coupled dynamic characteristics. We validated

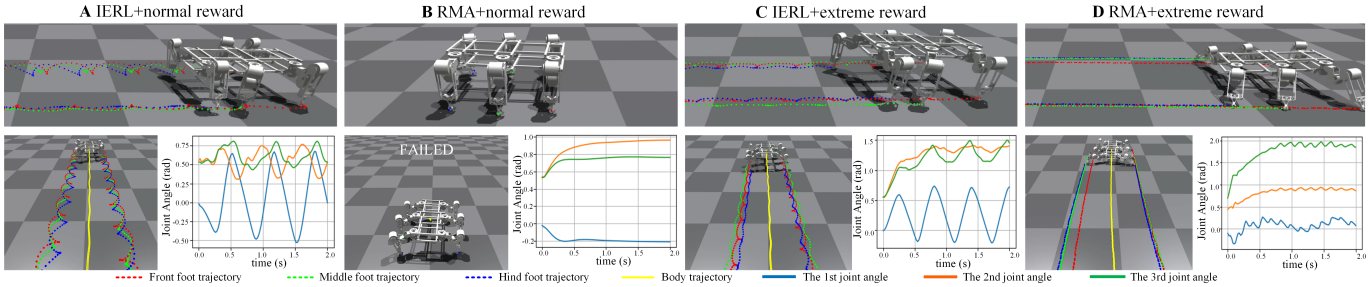

Fig. 4. (A) exhibited stable motion, while (B) was unable to move at all. (C) demonstrated more aggressive movement behavior with slightly reduced body stability but still managed to show reasonable movement behavior. However, (D) resulted in extremely unreasonable motion, with all legs rapidly shaking within a small range and failing to form a coordinated and periodic gait movement.

the accuracy of the model using test data collected from real robots and recorded the Root Mean Square Error(RMSE) between the predicted and actual values, as shown in Table I, our integrated modeling approach enhanced accuracy by over 50% compared to individual motor models.

TABLE I
COMPARISON OF MOTOR MODELS

Method	RMSE				
	Exp.1	Exp.2	Exp.4	Exp.4	Average
Ours	0.5825	0.5602	0.5946	0.6845	0.6054
[1]	1.2217	1.3335	1.2275	1.6202	1.3507

Reward function: Similar to previous work [29], we employed a reward function that encouraged more precise velocity tracking while penalizing unsafe behaviors such as instability in the robot's body, excessively high torque, and rotational speeds.

The specific reward functions and weights are shown in Table II, We did not restrict the desired body height of the robot or include any reward function that limited its gait, as we aimed for the robot to discover more suitable actions for itself through RL. Additionally, we designed two sets of different reward weights. As summarized in the table, the normal weights represent a balanced combination of rewards and penalties, while the extreme weights strongly encourage robot movement while minimizing penalties.

Other details: The terrain used includes flat ground, rough terrain, sloped terrain, stairs, and discrete obstacles. During the IL stage and the initial RL stage, we use only flat ground to ensure that the reference trajectory can be effectively learned and inherited. In the subsequent RL stage, different terrains are introduced in a curriculum-based manner. The range of linear velocity commands gradually increases from an initial 0.8m/s to 2.3m/s. Our curriculum design and randomization methods are based on the approach outlined in [29].

TABLE II
REWARD FUNCTION

Term	Expression	Normal-weight	Extreme-weight
linear vel	$\exp\left(-\frac{\|v_{cmd}-v_t\|^2}{\sigma_v}\right)$	$6dt$	$7.5dt$
angular vel	$\exp\left(-\frac{\|w_{cmd}-w_t\|^2}{\sigma_w}\right)$	$6dt$	$7.5dt$
Z-axis vel	$-\|v_z\|^2$	$2dt$	$0.2dt$
orientation	$-\left(\|g_x\|^2+\|g_y\|^2\right)$	$2dt$	$0.2dt$
joint tor	$-\sum_{i=1}^{18}\|torque_i\|^2$	$0.0006dt$	$0.00001dt$
joint vel	$-\sum_{i=1}^{18}\|vel_i\|^2$	$0.0005dt$	$0.00001dt$
joint acc	$-\sum_{i=1}^{18}\left\|\frac{last\_vel-vel_i}{dt}\right\|^2$	1.5e-6dt	6.e-7dt
CoT	$-\sum_{i=1}^{18}\|torque_i\cdot vel_i\|^2$	$0.008dt$	0

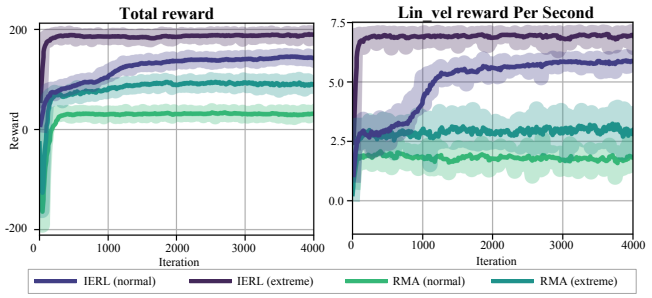

Fig. 5. Regardless of the reward weight, the IERL performs better.

V. RESULT

In this section, we experimentally investigate the following research questions: 1) Can IERL successfully train end-to-end control policy for hexapod robots? 2) Is a smooth transition method necessary? 3) What is the robustness of our approach when deployed in the real world?

We conducted two ablation experiments. First, we compared IERL with a pure RL method to verify the importance of imitation learning pre-training for effective hexapod motion. Then, we explored whether the pre-trained model retains knowledge from the reference policy in the RL stage, with and without the smooth transition method. Finally, we conducted extensive real-world experiments.

A. Motion feasibility analysis

Taking into account the similarities in both network structure and learning difficulty, we employed the RMA [6] method as an RL baseline to directly train a hexapod robot and compared it with IERL to validate the effectiveness of our end-to-end control walking implementation for hexapod robots. To ensure a fair comparison, both methods used the same reward function weights and initial training parameter configurations. The training was conducted under both normal and extreme reward function weight settings as shown in Table II.

To prevent training collapse from early exposure to challenging situations, we initially trained the robot on a simple task (flat terrain with a maximum linear velocity of 0.8 m/s).

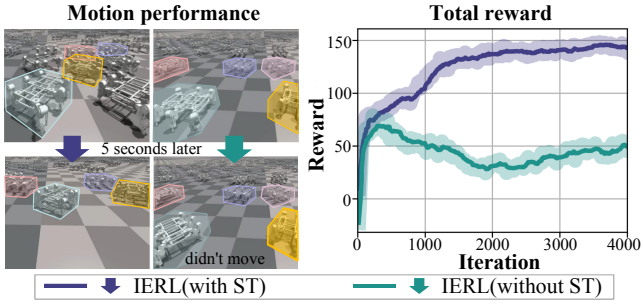

Fig. 6. By incorporating the ST method, the agent successfully inherits expert knowledge. Without the ST method, the robot's movements collapse, rendering it nearly immobile, and resulting in a rapid decline in the total reward value. For more details, see the supplementary video at the 50-second mark.

Fig.5 shows the total reward values and linear velocity rewards for both the baseline and our proposed method. The curves presented here represent only the performance during the reinforcement learning stage and do not include the pre-training stage. The total reward reflects the overall quality of the control strategy, while the linear velocity reward, as detailed in Table II, quantifies tracking accuracy—higher values indicate smaller errors and more effective walking.

Our method achieves superior performance across various reward weights, attaining near-maximum linear velocity rewards, whereas the RMA method without pre-training achieves less than half this value. We applied four policies to track a linear velocity of 0.8 m/s, with the motion trajectory and joint angle depicted in Fig.III-C. IERL produces coordinated, periodic gait movements, while reinforcement learning without pre-training fails to achieve effective control.

B. Smooth transition method analysis

To ensure a smooth transition from the IL to the RL stage, an ST module was introduced to address data inconsistencies during the switch. We used reinforcement learning to optimize the pre-trained model, randomly sampling velocity instructions between [-0.8 m/s, 0.8 m/s], and observed motion performance at the earliest RL stage. Fig. 6 compares the motion performance with and without a smooth transition method based on the same pre-trained model at the earliest RL stage.

The pre-training provided a high starting point for the agent. With the influence of the transition method, the input to the base network was gradually replaced by the complete encoder output, leading to rapid convergence during training. As shown in Fig.6, after five seconds, the robots with the ST method exhibited significant movement, whereas those without the ST method barely moved at all. This validates that through the ST method, the agent successfully inherits the skills from the expert strategy, enabling a consistent increase in the total reward value and ultimately achieving comprehensive optimization of the hexapod robot's motion control.

However, without the smooth transition method, there was a sudden change in some of the input data to the base network, resulting in deviations in the agent's actions. This caused the agent to lose its ability to walk and forget the knowledge obtained from expert policy. Consequently, the reward value rapidly decreased and then slowly rose to a level similar to that of methods without pre-training. In the absence of the transition method, the role of the pre-training stage was minimal, and ultimately, effective walking was not achieved.

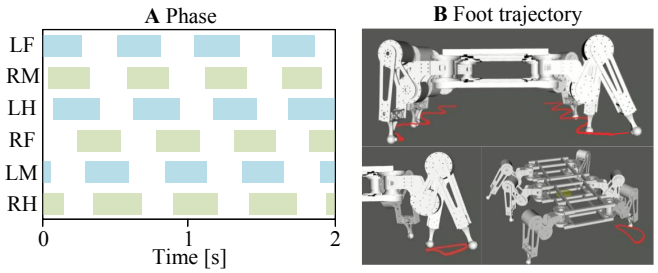

Fig. 7. The gait (A) and foot trajectory (B) of the hexapod robot has been greatly optimized.

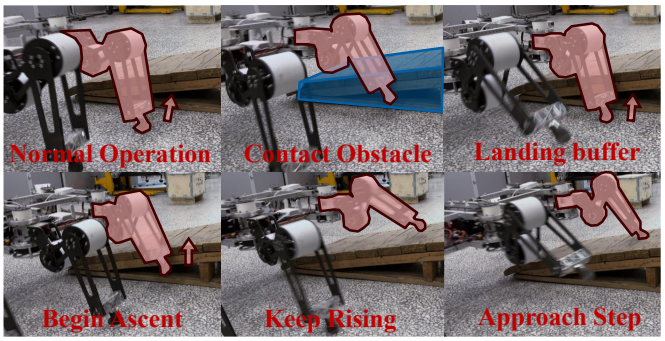

Fig. 8. The hexapod robot identifies obstacles through a period of historical observation information and crosses a 23cm-high fixed obstacle on flat ground.

C. Hardware experiments

The simulation results of the different methods are summarized in Table III and the attached video includes some of the physical comparison experiments. In addition, we have deployed our method on a real hexapod robot.

During the pre-training stage, we used a simple tripod gait and planar cycloidal trajectory as the reference actions. Obviously, this is not an ideal choice, but as we anticipated, the robot's movements were fully optimized during the RL stage to suit the capabilities of the robot better.

Fig. 7 shows the gait and foot trajectory achieved by the hexapod robot using our method. Interestingly, the robot's gait naturally optimized into a metachronal wave, similar to the gait of centipedes in nature, even though this gait is not commonly used for hexapod robots. Additionally, the foot trajectory was optimized into a spatially smooth curve to suit the robot.

Furthermore, it can be observed that we tested our policy on various terrains to verify its robustness, including stone bricks, rubber tracks, grass, and sand in Fig .2. Our method demonstrated significant improvements in both movement speed and terrain passability. It enabled the robot to achieve a maximum moving speed of 2m/s, exhibiting relatively high dynamic motion capabilities. Although this is still slower compared to current quadruped robots, it surpasses almost all hexapod robots of the same size. Additionally, our method achieved a certain degree of flight phase during high-speed running.

As shown in Fig. 8, when encountering abrupt changes in terrain, such as steps, our strategy relies on proprioceptive information to identify these variations, allowing the robot to successfully navigate through vertical obstacles. During normal operation, if an obstacle is encountered, the robot

TABLE III
SIMULATION PERFORMANCE OF COMPARISON METHODS

Method For Hexapod	Max speed (tracking err)		Success Rate					
			Step terrain			Slope terrain		
	lin (m/s)	ang (rad/s)	3cm	8cm	12cm	5deg	15deg	25deg
PC	0.24(±0.22)	0.28(±0.25)	75%	30%	0%	80%	20%	0%
MPC	0.41(±0.08)	0.56(±0.17)	90%	50%	0%	100%	60%	10%
[29]	FAIL	FAIL	\	\	\	\	\	\
RMA [6]	FAIL	FAIL	\	\	\	\	\	\
RL+CPG	0.78(±0.15)	0.42(±0.21)	60%	30%	0%	90%	30%	10%
[25]	0.66(±0.21)	0.85(±0.37)	90%	10%	0%	100%	10%	0%
IERL	**2.02(±0.04)**	**2.26(±0.08)**	**100%**	**100%**	**80%**	**100%**	**100%**	**80%**

becomes aware of the terrain change through contact between its legs and the obstacle. After a brief landing buffer, it immediately lifts its legs to overcome obstacles that would otherwise be impassable. In our testing, the robot was able to continuously traverse steps with a height of 12cm and a width of 35cm, and on flat ground, it could overcome obstacles with a maximum height of approximately 23cm. More details of the experiment can be viewed in our demonstration video.

VI. Conclusion

This letter presents a two-stage method for dynamic hexapod robot locomotion using imitation-enhanced reinforcement learning (IERL). By integrating imitation learning with reinforcement learning, we leveraged human expertise and autonomous capabilities for superior motion control. The smooth transition method ensured seamless data integration between stages. The approach showed robust performance across various terrains, reaching speeds up to 2m/s and overcoming 23cm obstacles. However, due to limited sensor capacity, IERL currently operates blindly, without incorporating vision or map data. There is also limited discussion on using additional sensory information. Future work will integrate visual information for better obstacle navigation and add advanced skills like jumping and object manipulation.

References

[1] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[3] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[4] Y. Jin, X. Liu, Y. Shao, H. Wang, and W. Yang, "High-speed quadrupedal locomotion by imitation-relaxation reinforcement learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1198–1208, 2022.

[5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[7] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, p. eade2256, 2023.

[8] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.

[9] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[10] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," *arXiv preprint arXiv:2309.05665*, 2023.

[11] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[12] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[13] A. S. Lele, Y. Fang, J. Ting, and A. Raychowdhury, "Learning to walk: Spike based reinforcement learning for hexapod robot central pattern generation," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2020, pp. 208–212.

[14] S. Trotta, "Walking motion generation in bio inspired hexapod robot using reinforcement learning," 2021.

[15] Z. Qiu, W. Wei, and X. Liu, "Adaptive gait generation for hexapod robots based on reinforcement learning and hierarchical framework," in *Actuators*, vol. 12, no. 2. MDPI, 2023, p. 75.

[16] K. Konen, T. Korthals, A. Melnik, and M. Schilling, "Biologically-inspired deep reinforcement learning of modular control for a six-legged robot," in *2019 IEEE International Conference on Robotics and Automation Workshop on Learning Legged Locomotion Workshop,(ICRA) 2019, Montreal, CA, May 20-25, 2019*, 2019.

[17] M. Schilling, K. Konen, F. W. Ohl, and T. Korthals, "Decentralized deep reinforcement learning for a distributed and adaptive locomotion controller of a hexapod robot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5335–5342.

[18] T. Azayev and K. Zimmerman, "Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification," *Journal of Intelligent & Robotic Systems*, vol. 99, no. 3, pp. 659–671, 2020.

[19] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.

[20] A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter, "Imitation learning from mpc for quadrupedal multi-gait control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5014–5020.

[21] D. Youm, H. Jung, H. Kim, J. Hwangbo, H.-W. Park, and S. Ha, "Imitating and finetuning model predictive control for robust and symmetric quadrupedal locomotion," *IEEE Robotics and Automation Letters*, 2023.

[22] A. Miller, S. Fahmi, M. Chignoli, and S. Kim, "Reinforcement learning for legged robots: Motion imitation from model-based optimal control," *arXiv preprint arXiv:2305.10989*, 2023.

[23] C. Li, S. Blaes, P. Kolev, M. Vlastelica, J. Frey, and G. Martius, "Versatile skill control via self-supervised adversarial imitation of unlabeled mixed motions," in *2023 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 2944–2950.

[24] Y. Fuchioka, Z. Xie, and M. Van de Panne, "Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5092–5098.

[25] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[26] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[27] T. Li, Y. Zhang, C. Zhang, Q. Zhu, J. Sheng, W. Chi, C. Zhou, and L. Han, "Learning terrain-adaptive locomotion with agile behaviors by imitating animals," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 339–345.

[28] Q. Yao, J. Wang, S. Yang, C. Wang, H. Zhang, Q. Zhang, and D. Wang, "Imitation and adaptation based on consistency: A quadruped robot imitates animals from videos using deep reinforcement learning," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2022, pp. 1414–1419.

[29] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.