actuators

MDPI

Essay

Adaptive Gait Generation for Hexapod Robots Based on Reinforcement Learning and Hierarchical Framework

Zhiying Qiu [1], Wu Wei [1,2,*] and Xiongding Liu [1]

1 School of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China
2 The Key Laboratory of Autonomous Systems and Networked Control, Ministry of Education, Unmanned Aerial Vehicle Systems Engineering Technology Research Center of Guangdong, South China University of Technology, Guangzhou 510641, China
* Correspondence: weiwu@scut.edu.cn; Tel.: +86-139-2601-3970

Abstract: Gait plays a decisive role in the performance of hexapod robot walking; this paper focuses on adaptive gait generation with reinforcement learning for a hexapod robot. Moreover, the hexapod robot has a high-dimensional action space and therefore it is a great challenge to use reinforcement learning to directly train the robot's joint angles. As a result, a hierarchical and modular framework and learning details are proposed in this paper, using only seven-dimensional vectors to denote the agent actions. In addition, we conduct experiments and deploy the proposed framework using a real hexapod robot. The experimental results show that superior reinforcement learning algorithms can converge in our framework, such as SAC, PPO, DDPG and TD3. Specifically, the gait policy trained in our framework can generate new adaptive hexapod gait on flat terrain, which is stable and has lower transportation cost than rhythmic gaits.

Keywords: hexapod robot; reinforcement learning; hierarchical framework; gait generation

check for updates

Citation: Qiu, Z.; Wei, W.; Liu, X. Adaptive Gait Generation for Hexapod Robots Based on Reinforcement Learning and Hierarchical Framework. *Actuators* **2023**, *12*, 75. https://doi.org/10.3390/act12020075

Academic Editor: Chun-Fei Hsu

Received: 6 December 2022
Revised: 2 February 2023
Accepted: 6 February 2023
Published: 9 February 2023

1. Introduction

Compared with wheeled robots, a hexapod robot has many advantages, including rich gait [1], strong load capacity [2] and discrete support dependence [3]. With superior terrain adaptability and excellent performances, hexapod robots have been widely applied to disaster rescue [4], factory automation [5], intelligent maintenance [6] and exploration [7]. Nevertheless, most research still focuses on hexapod robot gait planning and motion control [8]. Regarding robotic control, hexapod robots are high-dimensional, omnidirectional and non-smooth systems with complex kinematic structures, uncertain dynamics and inherently diverse physical constraints [9,10]. In terms of robotic motion, a hexapod robot relies on the alternate support and swing of each limb to advance its body's motion [11], and therefore the movement of the hexapod robot is constrained by gait.

Locomotion in legged animals is characterized as a rhythmic behavior [12]. To achieve a rhythmic gait, some existing studies have used Central Pattern Generators (CPGs) as open-loop oscillators [13], which rely on centrally generated rhythms to drive overall behavior. Support for such approaches has been found, in particular, on fast walking in insects, such as rapid locomotion in cockroaches [14] or the running and swimming behavior of the salamander [15]. Importantly, with a high number of joints, this becomes a challenging problem, and open-loop oscillator solutions are usually not applicable [16]. This and has led to biologically inspired control approaches through combined areas of research. As one example, the Walknet approach for hexapod robots realizes a decentralized and modular structure that reflects insights from walking in stick insects [17]. While this approach can deal with a variety of disturbances during locomotion, it is still limited when dealing with novel and challenging walking situations [16]. In recent years, data-driven-based methods, such as reinforcement learning and neural networks, have attracted significant attention because of their ability to create more accurate and robust control policies, which provides new feasible schemes for hexapod robot motion control [18].

Reinforcement learning (RL) can learn feasible planning and control strategies from data and trials. RL has been extensively studied and applied in terms of legged robot walking tasks. For instance, Peng et al. proposed a deep learning optimization strategy to train a biped robot in simulated animations, enabling the robot to pass through random obstacles freely [19]. In addition, Tan used proximal policy optimization to train a quadruped robot motion strategy and realized the transformation from simulation to physical robot [20]. Similarly, Tsounis constructed the Markov decision process (MDP) and planned trajectories in a high-dimensional, continuous, state-action space, realizing the stable movement of the robot in different environments [21]. In relevant research, Deep Reinforcement Learning (DRL) has proven itself capable of automatically acquiring control policies to accomplish a large variety of challenging locomotion tasks. Fu et al. designed a novel DRL method to implement multi-contact motion planning for hexapod robots moving on uneven plum-blossom piles [22]. Thor and Manoonpong proposed a simple yet versatile modular neural control framework with fast learning in which behavior-specific control modules could be added incrementally to obtain increasingly complex emergent locomotion behaviors [23]. Showing the impressive performance of DRL for special tasks in footed robots, Miki employed a model-free reinforcement learning approach to train a deep policy which enabled the ANYmal robot to balance a light-weight ball robustly using its limbs without any contact measurement sensor [24].

As a series-parallel composite omnidirectional mobile vehicle, the hexapod robot is a complex coupling agent. As a result, using DRL to train an end-to-end controller leads to non-convergence and gait disorder during training [25]. This has led to a growing interest in hierarchical control frameworks and how these frameworks could be exploited to improve behaviors in DRL. For example, Merel et al. proposed several such bio-inspired principles of hierarchical control and also advocated their implementation into robot architectures [26]. As one example, they emphasize a hierarchical framework with a higher-level planner modulating a lower-level controller. In addition, Eppe et al. provided the cognitive foundations of hierarchical problem-solving and proposed steps to integrate biologically inspired hierarchical mechanisms to enable advanced problem-solving skills in artificial agents [27]. Both works above provide a detailed and excellent overview for hierarchical reinforcement learning, which was significant to our paper. Similarly, Azayev and Zimmerman trained policies independently in individual scenarios using DRL, and presented a scalable two-level hierarchy for hexapod locomotion through complex terrain without the use of exteroceptive sensors [1].

This paper studies the gait generation of hexapod robots and realizes adaptive gait generation using RL and hierarchical framework. The main contributions of this paper are as follows. First, we developed a RL-based hierarchical control framework to reduce the dimensionality of the action space, which was then deployed in a real robot. Specifically, for the speed-adaptive gait generation task, we describe the whole learning process, including the MDP formulation, detailed training settings and policy gradient algorithm. Furthermore, we designed simulations and experiments to demonstrate our framework's effectiveness. Overall, the proposed hierarchical framework appears novel and unique enough to be applied in reinforcement learning. In addition, we designed a specific trajectory planner, Inverse Kinematics solver and trajectory tracking controller for the hexapod robot. This paper makes a valuable contribution to the *Actuators* journal.

The rest of this work is organized as follows. Section 2 introduces the hexapod robot VkHex and the RL-based hierarchical framework. Section 3 describes the design of our framework for the hexapod robot gait generation task. Section 4 presents training details and experiments to verify the effectiveness and feasibility of the proposed framework. Finally, conclusions and avenues for future research are presented in Section 5.

2. Robot Prototype and Hierarchical Framework

In this section, we introduce the key points of the physical prototype and gait planning of the hexapod robot VkHex, and propose the corresponding hierarchical architecture based on reinforcement learning.

2.1. Hexapod Robot

We designed a hexapod robot with 18 degrees of freedom (DOF) and constructed a simulation environment using the PyBullet [28] physical engine. As shown in Figure 1, VkHex adopts a rectangular structure with six identical legs symmetrically distributed.

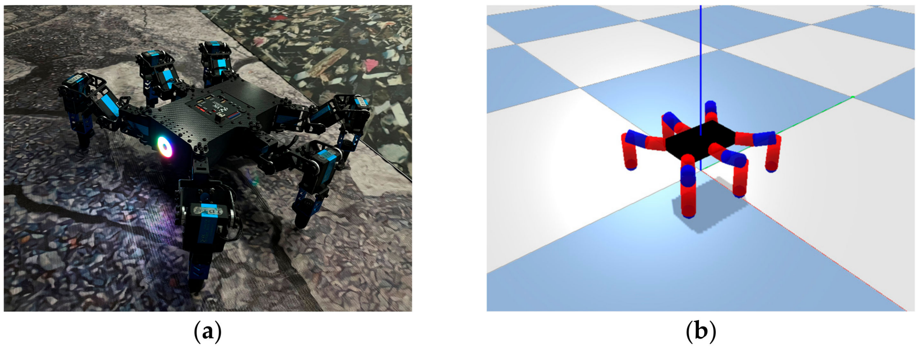


Figure 1. (**a**) Physical prototype of the hexapod robot VkHex; (**b**) simulated prototype of the VkHex in PyBullet.

The six legs of the hexapod robot are distributed as shown in Figure 2; each leg can be abstracted into a three-link mechanism consisting of hip, knee and ankle joints in series. Table 1 shows details on the VkHex, including dimensions, weight and references.

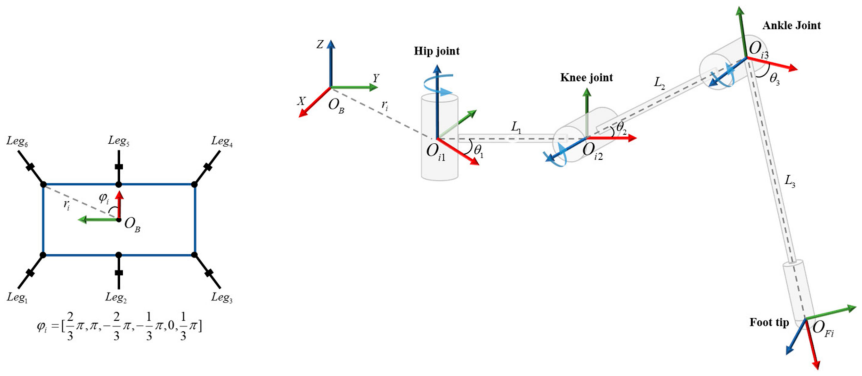


Figure 2. The rotation angles of the three active joints relative to the initial position are denoted as θ_1, θ_2 and θ_3. The lengths of the three links are recorded as L_1, L_2 and L_3. The distance from the center of the body to the hip joint of the Leg_i is r_i ($i = 1, \cdots, 6$). A body coordinate frame $\{B\}$ is located in the center of the body. The z-axis of the joint coordinate frame $\{OJ_{ij}\}$ ($j = 1, 2, 3$) coincides with the joint rotation axis. Additionally, the foot tip coordinate frame is denoted as $\{OFi\}$.

Table 1. VkHex details and parameters.

Parameter	Value	Parameter	Value
Body dimensions	0.3866 × 0.2232 × 0.0821 m	Robot weight	10 kg
DoF	18	Hip link L_1	0.1179 m
Knee link L_2	0.1302 m	Ankle link L_3	0.3996 m
Servo size	72 × 40 × 56 mm	Servo weight	72 g
Servo parameters	27.5 kg/cm 7.4V	Computing device	Nvidia Tx2
9-axis IMU	MPU9250	Power	7.4 V 8000 mAh

2.2. Reinforcement Learning and Hierarchical Framework

Most existing research has applied RL to learn an end-to-end motion controller to control joints directly. However, as the number of robot joints increases, the end-to-end motion controller training becomes difficult and tends to diverge, which is not suitable for a hexapod robot [29]. To address this, we introduced a policy network, gait planner and trajectory planner to design a RL-based modular hierarchical framework. Figure 3 shows the designed modular hierarchical framework based on RL.

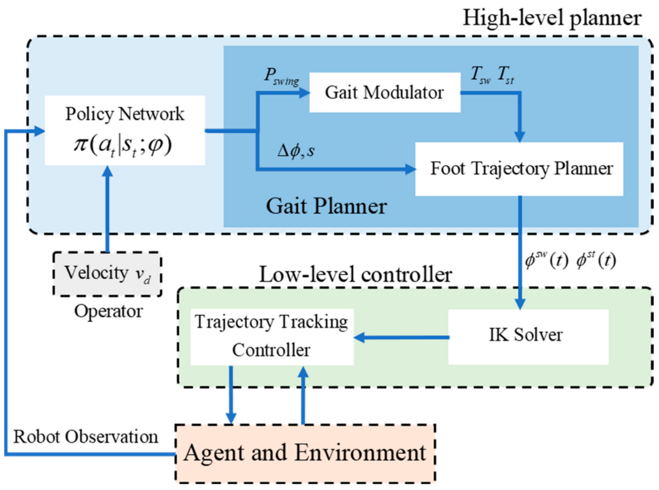


Figure 3. Gait generation and motion control architecture based on reinforcement learning and modular hierarchical framework. This architecture decouples the planning and motion control problem of a hexapod robot into two levels, including a motion planner (shown in blue) and joint trajectory controller (shown in green).

We divided the RL-based hierarchical control architecture into a high-level planner and a low-level controller. The high-level planner includes a gait policy network $\pi(a_t|s_t;\varphi)$ and a gait planner. The gait policy network trained by RL outputs the optimal gait parameters to the gait planner, including P_{swing}, $\Delta\phi$ and s. Then, the gait modulator and foot trajectory planner generate the swing and support phase functions $\phi^{sw}(t)$ and $\phi^{st}(t)$. The Inverse Kinematics (IK) solver and the trajectory tracking controller make up the lower-level controller, and the IK solver converts the foot trajectories into joint angles $\theta_{ij}(i = 1\ldots6; j = 1,2,3)$. Finally, the trajectory tracking controller receives joint commands and joint position errors to outputs motor control signals.

(a) *Gait policy network*

We used a two-layer multi-layer perceptron (MLP) [30] with a tanh activation function to estimate the RL policy function. As shown in Figure 4, the MLP input is a 17D robot observation vector, and the 7D output is directly sent to the gait planner, including the gait phase difference, duty factor and step length.

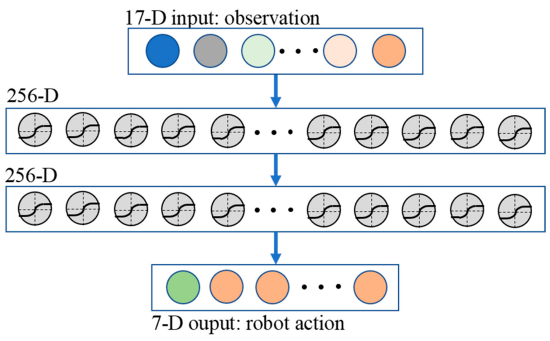


Figure 4. Network architecture used for gait policy is a two-layer MLP with tanh activation function. The input is the observation vector of the agent. The output is an extremely simple 7D vector.

(b) Gait modular

When walking on a flat road, hexapod robots alternate between supporting and swing phases to advance the body's motion. In general, the gait duty factor P_{swing} indicates the proportion of the swing phase to the whole gait cycle:

$$P_{swing} = \frac{T_{sw}}{T_{st} + T_{sw}}, \tag{1}$$

where T_{st} is the support time in a gait cycle and T_{sw} is the swing time.

Then, we used a phase function $\phi(t) \in [0,2)$ and the relative phase differences $\Delta\phi_i$ to parameterize the state of each leg. $\phi(t^{sw}) \in [0,1)$ indicates that the leg is in the swing phase, and $\phi\left(t^{st}\right) \in [1,2)$ represents the support phase. Where leg-1 is the reference leg and the phase is ϕ_1, the phases of leg-i can be represented by the following:

$$\phi_i = \phi_1 + \Delta\phi_i \quad (i = 2,3,4,5,6). \tag{2}$$

(c) Trajectory planner

The foot trajectory of a hexapod robot includes the swing and support phase trajectories. In this work, we adopt different phase functions for foot trajectories in three directions:

$$\begin{cases} x = y = s \cdot \phi_x^{sw}(t), \\ z = h \cdot \phi_z^{sw}(t), \end{cases} \tag{3}$$

where (x,y,z) is the position of the foot tip relative to the body coordinate system at time t, s is the step length and h is the step length.

When using the cycloid as the swing phase trajectory, the foot tip slides when touching the ground and drags when walking [31]. As shown in Figure 5, we designed a new cycloidal function as the swing phase trajectory:

$$\begin{aligned} &\phi_x^{sw}(t) = \phi_y^{sw}(t) = \tfrac{t}{T_{sw}} - \tfrac{1}{2\pi}\sin\left(\tfrac{2\pi t}{T_{sw}}\right), \\ &\phi_z^{sw}(t) = sgn\left(\tfrac{T_{sw}}{2} - t\right)\left(2\left(\tfrac{t}{T_{sw}} - \tfrac{1}{4\pi}\right)\sin\left(\tfrac{4\pi t}{T_{sw}}\right) - 1\right) + 1, \end{aligned} \tag{4}$$

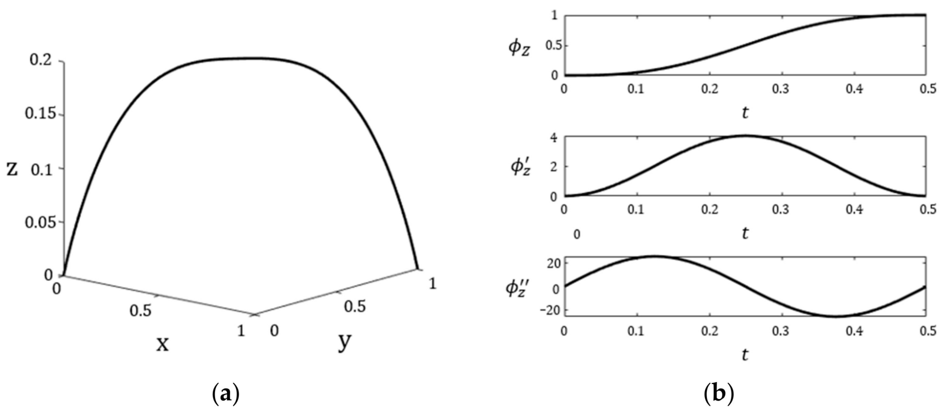


Figure 5. Improved cycloidal trajectory (**a**) and cycloidal curve in z-axis (**b**). The modified cycloidal curve eliminates sudden acceleration change in the z-axis direction, as shown in (**b**).

To complete the periodic motion, the robot must satisfy momentum conservation in the vertical direction, but direct force control to ensure the stability of robot motion is extremely difficult [32]. Considering the hexapod robot equilibrium-point hypothesis [33], we indirectly generated the required support force through position control. Therefore, we used a sinusoidal function as the support phase trajectory:

$$z = -\delta \sin\left(\pi \frac{t}{T_{st}}\right) - \eta \cdot \delta \sin\left(2\pi \frac{t}{T_{st}}\right), \tag{5}$$

where the amplitude δ is the virtual depth of the foot tip proportional to the support force. Figure 6 shows the support phase trajectories with different parameters.

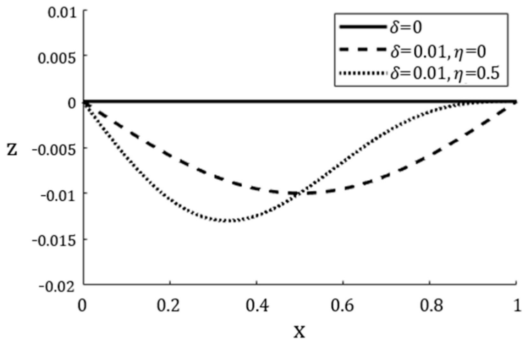


Figure 6. The support phase trajectory. In function expression, different amplitudes δ and η can be set to achieve the ideal support phase position control; we adopted $\delta = 0.01$ and $\eta = 0.5$ in this work.

(d) *Inverse Kinematics solver*

If the position of the foot tip in the hip joint coordinate system $\boldsymbol{P} = \left({}^{J_1}x,\ {}^{J_1}y, {}^{J_1}z\right)$ is known, the IK solver calculates the relationship between the joint rotation angles θ_1, θ_2, θ_3 and the foot tip position $\boldsymbol{P}$:

$$\begin{cases} \theta_1 = \tan^{-1}\left(\frac{{}^{J_1}y}{{}^{J_1}x}\right), \\ \theta_2 = \cos^{-1}\left(\frac{{d_1}^2+{l_2}^2-{l_3}^2}{2d_1 l_2}\right) - \cos^{-1}\left(\frac{{d_1}^2+{d_2}^2-{}^{J_1}z^2}{2d_1 d_2}\right) \\ \theta_3 = \cos^{-1}\left(\frac{{l_2}^2+{l_3}^2-{d_2}^2}{2l_2 l_3}\right) - \pi, \end{cases}, \tag{6}$$

where, $d_1 = \sqrt{{}^{J_1}x^2 + {}^{J_1}y^2} - l_1, d_2 = \sqrt{{d_1}^2 + {}^{J_1}z^2}$.

(e) *Trajectory Tracking Controller*

In the hierarchical framework, we adopted a sliding mode controller as the joint trajectory tracking controller and used a nonlinear potential-like function to place the integral:

$$\rho(\lambda \cdot e) = \lambda \frac{e^{\lambda x} - e^{-\lambda x}}{e^{\lambda x} + e^{-\lambda x}} = \lambda\ tanh(\lambda \cdot x), \tag{7}$$

where λ is the regulator and e is the joint position error.

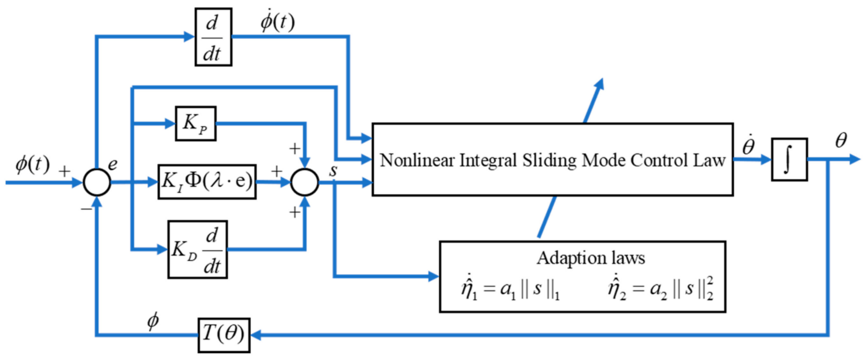
Then, a sliding surface with an error integral term was designed to improve the tracking accuracy, reduce the steady-state error, and avoid the difficult convergence problem, expressed as:

$$\boldsymbol{s} = K_P e + K_I \int_0^t \rho(\lambda \cdot e) d\tau + K_D \dot{e} \tag{8}$$

The control law of the nonlinear integral sliding mode control can be expressed as:

$$\dot{\theta} = J_v^{\dagger}(\theta)\left\{K_P^{-1}\left[\hat{\eta}_1 sgn\boldsymbol{s} + \hat{\eta}_2 \boldsymbol{s} + K_I \cdot \rho(\lambda \cdot e) + K_D \ddot{e}\right] + \dot{\phi}(t)\right\}, \tag{9}$$

where $J_v^{\dagger}(\theta) = J_v^T(\theta)\left[J_v(\theta) J_v^T(\theta)\right]^T$ is the pseudo-inverse matrix of the linear velocity Jacobi matrix, $\boldsymbol{s}$ is the sliding surface, $\hat{\eta}_1$ and $\hat{\eta}_2$ are estimates for adaptation laws and $\dot{\phi}(t)$ is the ideal foot-end velocity trajectory. Figure 7 shows the system block diagram of the nonlinear integral sliding mode controller.



$\frac{d}{dt}$
$\dot{\phi}(t)$
K_P
$\phi(t)$
e
$K_I\Phi(\lambda \cdot e)$
s
$K_D\frac{d}{dt}$
Nonlinear Integral Sliding Mode Control Law
$\dot{\theta}$
$\int$
θ
Adaption laws
$\dot{\hat{\eta}}_1 = a_1 \| s \|_1$
$\dot{\hat{\eta}}_2 = a_2 \| s \|_2^2$
ϕ
$T(\theta)$

Figure 7. The block diagram of the nonlinear integral sliding mode controller.

3. Learning Process

This section introduces the learning process for the robot gait generation task, including the environment's Markov decision process, RL model description, and policy training algorithm.

3.1. Markov Decision Process

The adaptive gait generate problem of the hexapod robot can be described as a Markov decision process. Here, the MDP is a 5-tuple $\{S, A, r, \Gamma, \gamma\}$, where S is the set of observations, A is the set of actions, $r : S \times A \to \mathbb{R}$ represents the reward given after action and observation, $\Gamma : S \times A \times S \to \mathbb{R}$ is the transition probability function and $\gamma \in (0, 1]$ is the discount factor of the MDP. The robot's goal is to interact with the environment through the optimal policy network that maximizes future rewards.

3.2. Action

Unlike common robotic arms and quadruped robots with fewer joints, VkHex has 18 driving joints and 42 alternative support states [34], meaning that the gait has to be searched in an extremely large space. In our framework, we use different parameter groups $\{P_{swing}, \Delta\phi_i, s\}$ to represent different motion gaits, which significantly reduces the dimension of the action space while being intuitive and simple. As a result, in the gait generation task and RL framework, one action includes a 7D vector:

- Gait duty factor P_{swing} (1D);
- Phase difference $\Delta\phi_i$ (5D);
- Step length s (1D).

3.3. Observation

The attributes in the observation space consist of only those measurable by VkHex. We categorized the observed attributes into three types: (1) values sensed by the robot, (2) values associated with historical moments, and (3) values related to the designed goals. The observation consists of the following attributes, which add up to a 17D vector:

- Current velocity of the VkHex v_{base}^{cur} (3D);
- Current angular velocity of the VkHex w_{base} (3D);
- Body platform height H (1D);
- Last moment action a_{t-1} (7D);
- Target velocity of the VkHex v_{base}^{target} (3D).

3.4. Reward Function

Our reward design encourages the robot to generate the most efficient gaits according to different velocity commands while completing the corresponding joint motion control without falling. Therefore, we used one primary reward and four penalties.

(i) In our study, the gait stability reward $r_{balance}$ is the primary reward. The new generated gaits must be stable enough. Only when the hexapod robot reaches the target position without falling can it receive a positive reward. The functional expression of this reward is as follows:

$$r_{balance} = \begin{cases} \lambda, & \Phi(x_{balance}) \geq 0, \\ -\alpha\lambda, & \Phi(x_{balance}) < 0, \end{cases} \tag{10}$$

where $\lambda > 0$ is the fixed reward value and α denotes discount factor. The function $\Phi(x_{balance})$ judges whether the robot is stable from the termination condition.

(ii) The learning task is that the hexapod robot can track speed commands and generate new RL-based gaits. The velocity tracking penalty r_{vel} forces the agent to move at the desired velocity:

$$r_{vel} = \exp\left(\|\overline{v}_{base} - v_{base}\|^2\right), \tag{11}$$

where $\overline{v}_{base}$ is the desired velocity and v_{base} is the actual velocity of the robot.

(iii) The energy consumption penalty r_{energy} is the penalty for gait motion efficiency and energy consumption. We used the cost of transportation (CoT) [35] as the penalty index. The expression is:

$$r_{energy} = \frac{\sum_{i=1}^{18} \tau_i \cdot \omega_i}{mg \cdot |v_{base}|}, \tag{12}$$

where τ_i is the joint torque, w_i is the joint angular velocity, m is the mass of the robot and g is the gravitational acceleration.

(iv) The joint tracking penalty r_{joint} is the penalty for the joint tracking error, which aims to improve the joint tracking control accuracy under the premise of stable motion:

$$r_{joint} = \exp\left(-p\|\Delta\theta_i\|^2\right) + \exp\left(-q\|\Delta\dot{\theta}_i\|^2\right), \tag{13}$$

where $\Delta\theta_i$ is the joint position error, $\Delta\dot{\theta}_i$ is the joint velocity error and p and q are the coefficients.

(v) The roll and pitch penalty r_{rp} penalizes the roll-pitch-yaw angle of the body, which can further improve the stability of RL-based gait:

$$r_{rp} = \exp\left(-\beta\|\omega_{base}\|^2\right), \tag{14}$$

where ω_{base} is the angular velocity of the robot platform and β is the coefficient.

3.5. Termination Condition

We used an early termination strategy to avoid falling into the local minimum and improve sampling efficiency. If one of the following conditions was met, the agent terminated the training and started again from the initial state:

- The robot is involved in a self-collision.
- The pitch or roll degree of the base exceeds the allowable range.
- The base height is less than the set threshold.
- Any link except the foot-tip collides with the ground.

3.6. Policy Training

The policy and critic networks are MLPs with two hidden layers each, while the action and observation vectors are the output and input, respectively. We adopted the *Soft Actor-Critic* (SAC) algorithm [36] to maximize the expected reward return:

$$\varphi^* = \underset{\varphi}{\operatorname{argmax}} E_{\pi_\varphi}\left[\sum_{t=0}^{\infty}\gamma^t\left(r_{balance} + w_1 \cdot r_{vel} + w_2 \cdot r_{energy} + w_3 \cdot r_{joint} + w_4 \cdot r_{rp}\right)\right] \quad (15)$$

where γ is the discount factor and $w_i (i = 1, 2, 3, 4)$ is the penalty factor. SAC is an off-policy maximum-entropy DRL algorithm where the actor aims to maximize expected reward and entropy. In the RL framework, SAC provides sample-efficient learning while retaining the benefits of entropy maximization and stability. Algorithm 1 summarizes the essential steps of SAC, where λ_V, λ_Q and λ_π are the gradients and $\hat{\nabla}_{\cdot} J(\cdot)$ are the approximate gradient functions [36]. The specific hyperparameters found empirically in preliminary experiments are shown in Table 2.

Algorithm 1: Soft Actor-Critic

```
1  Initialize policy parameters $\varphi$, replay buffer $D = \{\}$, Soft $Q$-function parameters $\theta_i$, Soft value
   function $V$ parameters $\Psi$ and Target critic function $V'$ parameters $\overline{\Psi}$.
2  for iteration = 1, M do:
3      for environment step = 1, N-1 do:
4          $a_t \sim \pi_\varphi(a_t|s_t)$
5          $s_{t+1} \sim \Gamma(s_{t+1}|s_t, a_t)$
6          $D \sim D \cup \{(s_t, a_t, r_t, s_{t+1})\}$
7      end
8      for gradient step = 1, T-1 do:
9          Update $V$ via minimizing the squared residual error: $\Psi \leftarrow \Psi - \lambda_V \hat{\nabla}_\Psi J_V(\Psi)$
10         Update $Q$ and $Q'$ via minimizing the soft Bellman residual: $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for
   $i \in \{1, 2\}$
11         Update $\pi_\varphi$ via minimizing the expected KL divergence: $\varphi \leftarrow \varphi - \lambda_\pi \hat{\nabla}_\varphi J_\pi(\varphi)$
12         Update $V'$: $\overline{\Psi} \leftarrow \tau\Psi + (1 - \tau)\,\overline{\Psi}$
13     end
14 end
```

Table 2. Algorithm and model hyperparameters.

Parameter	Value	Parameter	Value
Learning rate	3×10^{-4}	Replay buffer size	1×10^5
Discount factor	0.99	Entropy regularization	0.005
Policy network hidden layer nodes	[256,256]	Critic network hidden layer nodes	[400,300]
Parameter update frequency	1	Gradient update steps	1
Velocity tracking penalty factor w_1	-0.1	Energy consumption penalty factor w_2	-0.3
Joint tracking penalty factor w_3	-0.1	Roll and pitch penalty factor w_4	-0.35
Roll and pitch penalty coefficient ω	2	Joint tracking penalty coefficient m, n	1.5

4. Experiments and Results

In this section, we designed simulation and comparative experiments to verify the superiority and effectiveness of the RL-based hierarchical framework proposed in this paper.

4.1. Implementation Details

Environment bias and modeling uncertainties between the simulation and physical robot affects the porting of the RL model. We adopted the following details in model training to improve training efficiency and model robustness.

(i) Random model parameters [22]: We used the randomized model parameter strategy, which can improve the policy robustness against modeling errors and noise. Parameters of the robot model were sampled uniformly inside the range provided in Table 3.

Table 3. Parameter disturbance range of VkHex model.

Parameter	Lower Bound	Upper Bound
Centroid position	-2 cm	2 cm
Link mass	0.04 kg	0.06 kg
Rotational inertia	80%	120%
Joint max torque	80%	120%
Friction coefficient	0.5	1.2

(ii) Introduce sensor noise [30]: All the simulated sensors were noise-free, while the sensors caused data deviation because of the interference and noise in the actual observation. Therefore, we added normal distributed noise to the simulation robot observation and parameters, as shown in Table 4.

Table 4. Sensor noise sampling parameters.

Parameter	Sample Distribution
Joint position (rad)	$N(0, 0.003)$
Joint velocity (rad/s)	$N(0, 0.03)$
Joint torque (Nm)	$N(0, 0.5)$
Body height (m)	$N(0, 0.01)$
Body velocity (m/s)	$N(0, 0.1)$
Body angular velocity (rad/s)	$N(0, 0.1)$

(iii) External disturbance: Applying random disturbance force has been shown to be effective in achieving sim-to-real transfer and virtual load simulation [17]. During training, the external force was applied to the body from a random direction for every certain number of steps. The disturbance force was generated randomly within $(0, 5\text{N}]$ and lasted for 0.5 s.

4.2. Training Result

We compared the SAC with some of the algorithms with superior performance, including PPO [37], DDPG [38] and TD3 [39]. Figure 8 shows the learning curves, random sampling and sensor noise cause jitter. Compared with the three algorithms, SAC had the highest learning efficiency and the fastest rise rate in the initial stage. After 150,000 steps, the speed gradually decreased and finally converged. In addition, the convergence of different algorithms also proves the generality of the RL-based hierarchical framework proposed in this paper.

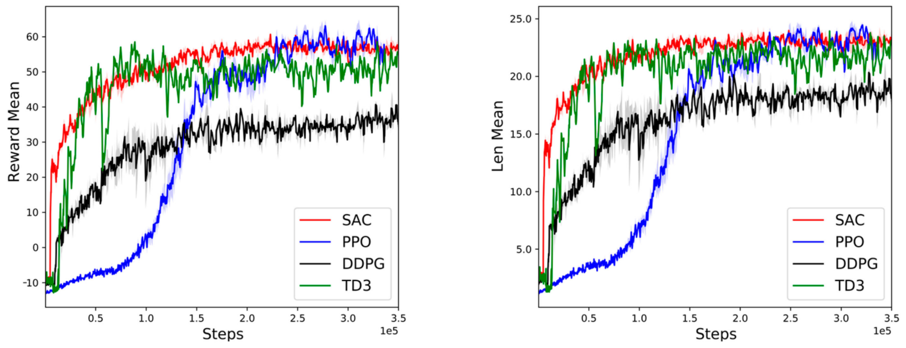


Figure 8. The reward curves and learning curves on reinforcement learning hierarchical framework for using SAC algorithm. We stopped training when the parameters were stable. The abscissa represents the number of steps and the ordinate represents the return value (the average epoch length is on the left and the average epoch reward on the right). Our framework learned the gait strategy and converged eventually.

4.3. Motion Verification

The initial height of the robot was 0.1 m and the direction was the y-axis of the body coordinate frame. We trained our robot by giving a specific target 1m in front. Once it reached its initial target, the next goal was given again to be 1m ahead repeatedly. We trained the gait policy network in the physics simulation, gradually increasing the expected velocity to 0.6 m/s. Figure 9 shows the motion process of the hexapod robot in seven continuous motion cycles, and Figure 10 shows the corresponding gait phase.

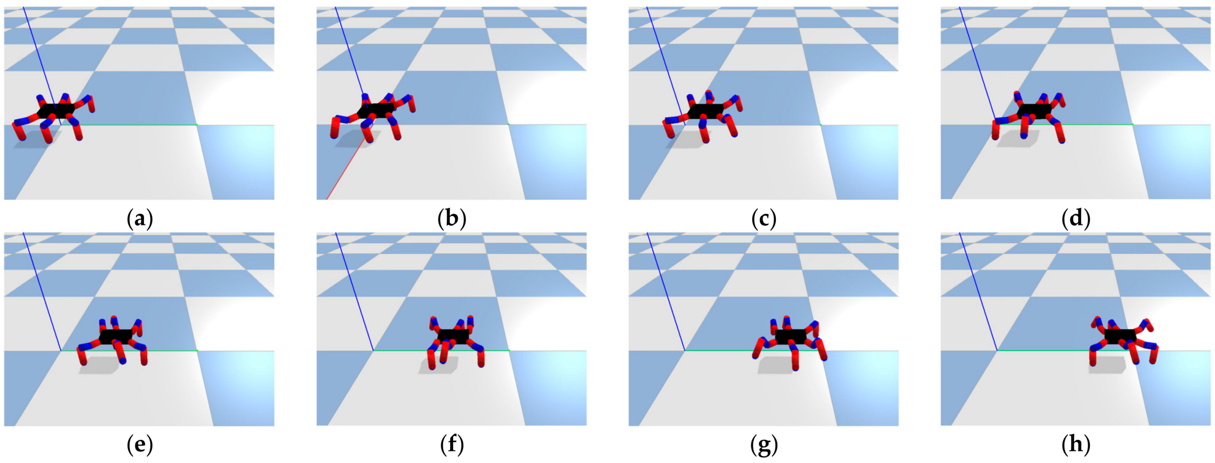


Figure 9. Speed-adaptive gaits of the hexapod robot trained in the hierarchical framework for reinforcement learning. From (**a**–**h**), the robot walked in a fixed direction and the motion speed gradually increased to 0.6 m/s. As the speed increased, the hexapod robot gradually transitioned from a wave-like gait to a tripod-like gait, and the robot was stable enough not to fall.

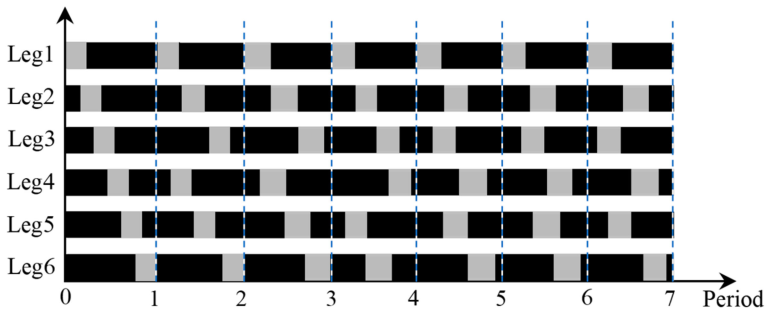
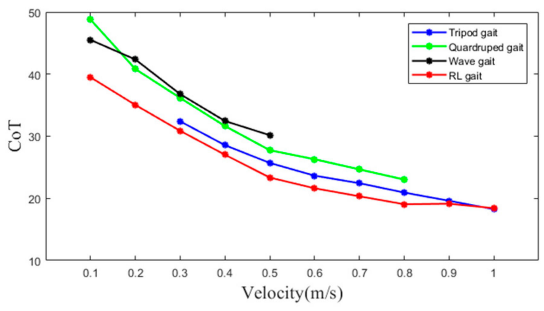


Figure 10. Visualization of legs' phase for the RL-based gaits. Black rectangles indicate support phases, gray rectangles indicate swing phases and block lengths indicate duration. The hexapod robot walked with a wave-like gait in the first cycle, a quadruped-like gait in the second to fourth cycles, and finally walked in a tripod-like gait. Within each gait cycle, RL-based gait had a different duty cycle and phase difference.

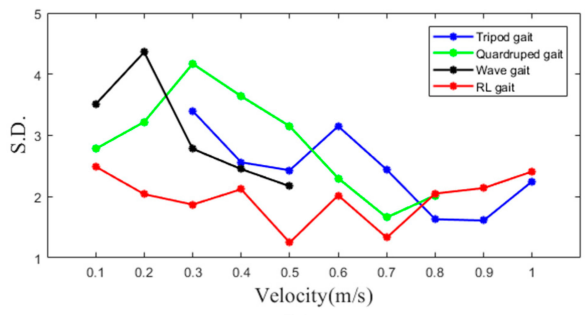
The results show that the gait strategy network can generate new gaits according to different velocity commands and motion velocity. The RL-based gait is similar to the tripod, quadruped and wave gaits, but the RL-based gait phase difference and duty factor differ. In addition, all these gaits were both new and stable. Since the reward function of the RL-based framework is dominated by stability, the swing phase took less time in the RL-based gait circle to achieve stability.

4.4. Motion Efficiency Comparison

Referring to Equation (10), we compared the transportation cost of the hexapod robot under the RL-based gaits and three rhythmic gaits. The robot motion gait period was set to 10, and the motion velocity was 0.1 to 1.1 m/s. We added up the transportation cost of all periods to be the total CoT and repeated the same experiment three times. Figure 11 shows the mean transportation cost and standard deviation (S.D.) of the hexapod robot in the four gaits.



(**a**)



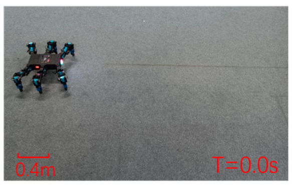
(**b**)

Figure 11. The mean CoT (**a**) and standard deviation (**b**) of the hexapod robot in four different gaits. Considering structural features and motion stabilization, a hexapod animal would not use a wave gait for fast walking and a tripod gait for slow walking.

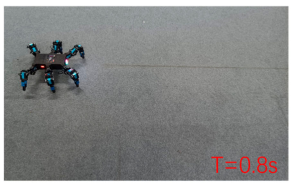
Comparing three rhythmic gaits, the RL-based gait had the lowest CoT and the highest motion efficiency. Additionally, the RL-based generated in our framework was speed-adapted. In the reward, the energy consumption penalty factor was second only to the gait stability reward, so the RL-generated gaits considered both gait stability and energy consumption. The experimental results show that the RL-based gait was superior to the three rhythmic gaits in motion efficiency.

4.5. Sim-to-Real

The previous simulations illustrate that gait policy network can be trained by our framework and is optimized enough to perform adaptive and stable gaits. In this experiment, we deployed the same policy on the physical VkHex robot and the policy ran 100 Hz on VkHex. As the movement speed increased from 0m/s to 0.5 m/s, the RL-based gaits of the hexapod robot behaved as shown in Figure 12. Compared with fixed rhythmic gait, the RL-based gaits performed more flexibly. As speed increased, VkHex could adaptively generate rhythmic-like gaits. Furthermore, all gaits were stable during locomotion, which, when performed in the real world, were similar to the simulated RL-based gaits. Since the environmental disturbances and modeling uncertainties between simulation and reality are different, such as servo force, friction coefficient, sensor noise and inaccurate motor models, the variety and stability of the gait in reality was slightly poorer than in simulation. The results of the simulated and real experiments further illustrate the effectiveness of the proposed framework.



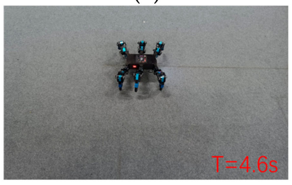
(**a**)



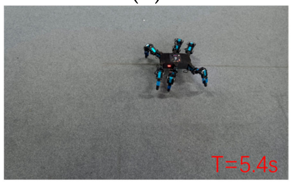
(**b**)

(**c**)

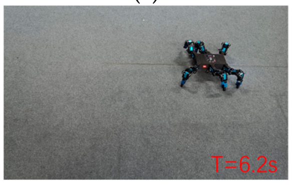
(**d**)



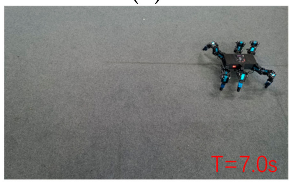
(**e**)



(**f**)



(**g**)



(**h**)

Figure 12. RL-based hierarchical framework tested on the VkHex prototype on flat terrain. Similarly, from (**a**–**h**), the robot walked in a fixed direction while the motion speed gradually increased to 0.5 m/s. The hexapod robot walked primarily in a triangular-like gait and slightly to the left.

5. Conclusions

In this paper, a RL-based hierarchical framework is proposed to simplify the hexapod action and learn a RL-based gait generation task, including a policy network, gait planner, IK solver and trajectory tracking controller. Furthermore, to train an adaptive gait generation policy network, we designed a whole RL framework for the hexapod robot and tested our framework using SAC, PPO, DDPG and TD3 in the physics simulation. Our framework enabled the DRL algorithms to converge and allowed the hexapod to generate adaptive new gaits on flat terrain. Finally, we ran the same policy in the real robot without any modification. Through comparison and experiments, it was verified that this framework is effective for RL tasks. This paper realizes the speed-adaptive gait generation and planning of the hexapod robot. We only tested using flat terrain, so further research could include a study of the similarities and differences in reward design under the RL framework for challenging terrains and tasks.

Author Contributions: All authors contributed to this work. Conceptualization, Z.Q.; methodology, Z.Q.; software, Z.Q.; validation, Z.Q. and W.W.; writing—original draft, Z.Q.; writing—review and editing, Z.Q. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China [Grant Nos. 61573148, 61603358] and the Science and Technology Planning Project of Guangdong Province, China [Grant Nos. 2015B010919007, 2019A050520001].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Azayev, T.; Zimmerman, K. Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification. *J. Intell. Robot. Syst.* **2020**, *99*, 659–671. [CrossRef]
2. Chen, Z.; Wang, S.; Wang, J.; Xu, K.; Lei, T.; Zhang, H.; Wang, X.; Liu, D.; Si, J. Control strategy of stable walking for a hexapod wheel-legged robot. *ISA Trans.* **2021**, *108*, 367–380. [CrossRef] [PubMed]
3. Gao, Y.; Wei, W.; Wang, X.; Li, Y.; Wang, D.; Yu, Q. Feasibility, planning and control of ground-wall transition for a suctorial hexapod robot. *Appl. Intell.* **2021**, *51*, 5506–5524. [CrossRef]
4. Sun, Q.; Gao, F.; Chen, X. Towards dynamic alternating tripod trotting of a pony-sized hexapod robot for disaster rescuing based on multi-modal impedance control. *Robotica* **2018**, *36*, 1048–1076. [CrossRef]
5. Melenbrink, N.; Werfel, J.; Menges, A. On-site autonomous construction robots: Towards unsupervised building. *Autom. Constr.* **2020**, *119*, 103312. [CrossRef]
6. Teixeira Vivaldini, K.C.; Franco Barbosa, G.; Santos, I.A.D.; Kim, P.H.C.; McMichael, G.; Guerra-Zubiaga, D.A. An intelligent hexapod robot for inspection of airframe components oriented by deep learning technique. *J. Braz. Soc. Mech. Sci. Eng.* **2021**, *43*, 494. [CrossRef]
7. Deepa, T.; Angalaeswari, S.; Subbulekshmi, D.; Krithiga, S.; Sujeeth, S.; Kathiravan, R. Design and implementation of bio inspired hexapod for exploration applications. *Mater. Today Proc.* **2021**, *37*, 1603–1607. [CrossRef]
8. Coelho, J.; Ribeiro, F.; Dias, B.; Lopes, G.; Flores, P. Trends in the Control of Hexapod Robots: A survey. *Robotics* **2021**, *10*, 100. [CrossRef]
9. Schilling, M.; Konen, K.; Ohl, F.W.; Korthals, T. Decentralized deep reinforcement learning for a distributed and adaptive locomotion controller of a hexapod robot. In Proceedings of the IROS 2020-International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 25–29 October 2020; pp. 5335–5342. [CrossRef]
10. Flores, P. Modeling and Simulation of Frictional Contacts in Multi-rigid-Body Systems. In *International Symposium on Multibody Systems and Mechatronics*; Springer: Cham, Switzerland, 2021; pp. 77–84. [CrossRef]
11. Gao, Y.; Wei, W.; Wang, X.; Wang, D.; Li, Y.; Yu, Q. Trajectory Tracking of Multi-Legged Robot Based on Model Predictive and Sliding Mode Control. *Inf. Sci.* **2022**, *606*, 489–511. [CrossRef]
12. Cai, Z.; Gao, Y.; Wei, W.; Gao, T.; Xie, Z. Model design and gait planning of hexapod climbing robot. *J. Phys. Conf. Ser. IOP Publ.* **2021**, *1754*, 012157. [CrossRef]

13. Ijspeert, A.J. Central pattern generators for locomotion control in animals and robots: A review. *Neural Netw.* **2008**, *21*, 642–653. [CrossRef] [PubMed]
14. Fuchs, E.; Holmes, P.; Kiemel, T.; Ayali, A. Intersegmental coordination of cockroach locomotion: Adaptive control of centrally coupled pattern generator circuits. *Front. Neural Circuits* **2011**, *4*, 125. [CrossRef] [PubMed]
15. Knüsel, J.; Crespi, A.; Cabelguen, J.M.; Lispeert, A.J.; Ryczko, D. Reproducing five motor behaviors in a salamander robot with virtual muscles and a distributed CPG controller regulated by drive signals and proprioceptive feedback. *Front. Neurorobot.* **2020**, *14*, 604426. [CrossRef] [PubMed]
16. Schilling, M.; Melnik, A. An approach to hierarchical deep reinforcement learning for a decentralized walking control architecture. *Biol. Inspired Cogn. Archit. Meet.* **2018**, *848*, 272–282. [CrossRef]
17. Schilling, M.; Hoinville, T.; Schmitz, J.; Cruse, H. Walknet, a bio-inspired controller for hexapod walking. *Biol. Cybern.* **2013**, *107*, 397–419. [CrossRef]
18. Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.* **2020**, *5*, eabc5986. [CrossRef]
19. Peng, X.B.; Berseth, G.; Yin, K.K.; Panne, M.V.D. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.* **2017**, *36*, 1–13. [CrossRef]
20. Tan, J.; Zhang, T.; Coumans, E.; Iscen, A.; Bai, Y.; Hafner, D.; Bohez, S.; Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv* **2018**. [CrossRef]
21. Tsounis, V.; Alge, M.; Lee, J.; Farshidian, F.; Hutter, M. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3699–3706. [CrossRef]
22. Fu, H.; Tang, K.; Li, P.; Zhang, W.; Wang, X.; Deng, G.; Wang, T.; Chen, C. Deep Reinforcement Learning for Multi-contact Motion Planning of Hexapod Robots. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 21 August 2021; pp. 2381–2388. [CrossRef]
23. Thor, M.; Manoonpong, P. Versatile modular neural locomotion control with fast learning. *Nat. Mach. Intell.* **2022**, *4*, 169–179. [CrossRef]
24. Miki, T.; Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.* **2022**, *7*, eabk2822. [CrossRef] [PubMed]
25. Lele, A.S.; Fang, Y.; Ting, J.; Raychowdhury, A. Learning to walk: Spike based reinforcement learning for hexapod robot central pattern generation. In Proceedings of the IEEE International Conference on Artificial Intelligence Circuits and Systems, Genoa, Italy, 31 August–4 September 2020; pp. 208–212. [CrossRef]
26. Merel, J.; Botvinick, M.; Wayne, G. Hierarchical motor control in mammals and machines. *Nat. Commun.* **2019**, *10*, 5489. [CrossRef] [PubMed]
27. Eppe, M.; Gumbsch, C.; Kerzel, M.; Butz, M.V.; Wermter, S. Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nat. Mach. Intell.* **2022**, *4*, 11–20. [CrossRef]
28. Panerati, J.; Zheng, H.; Zhou, S.Q.; Xu, J.; Prorok, A.; Schoellig, A.P. Learning to fly-a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Prague, Czech Republic, 27 September–1 October 2021; pp. 7512–7519. [CrossRef]
29. Khera, P.; Kumar, N. Role of machine learning in gait analysis: A review. *J. Med. Eng. Technol.* **2020**, *44*, 441–467. [CrossRef] [PubMed]
30. Shi, F.; Homberger, T.; Lee, J.; Miki, T.; Zhao, M. Circus anymal: A quadruped learning dexterous manipulation with its limbs. In Proceedings of the International Conference on Robotics and Automation, Xi'an, China, 30 May–5 June 2021; pp. 2316–2323. [CrossRef]
31. Kim, J.; Ba, D.X.; Yeom, H.; Bae, J. Gait optimization of a quadruped robot using evolutionary computation. *J. Bionic Eng.* **2021**, *18*, 306–318. [CrossRef]
32. Han, Y. Action Planning and Design of Humanoid Robot Based on Sports Analysis in Digital Economy Era. *Int. J. Multimed. Comput.* **2022**, *3*, 37–50. [CrossRef]
33. He, J.; Gao, F. Mechanism, actuation, perception, and control of highly dynamic multilegged robots: A review. *Chin. J. Mech. Eng.* **2020**, *33*, 79. [CrossRef]
34. Xu, P.; Ding, L.; Wang, Z.; Gao, H.; Zhou, R.; Gong, Z.; Liu, G. Contact sequence planning for hexapod robots in sparse foothold environment based on monte-carlo tree. *IEEE Robot. Autom. Lett.* **2021**, *7*, 826–833. [CrossRef]
35. Owaki, D.; Ishiguro, A. A quadruped robot exhibiting spontaneous gait transitions from walking to trotting to galloping. *Sci. Rep.* **2017**, *7*, 277. [CrossRef]
36. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. International conference on machine learning. *Proc. Mach. Learn. Res.* **2018**, *80*, 1861–1870. [CrossRef]
37. Zhang, Z.; Luo, X.; Liu, T.; Xie, S.; Wang, J.; Wang, W.; Li, Y.; Peng, Y. Proximal policy optimization with mixed distributed training. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 4–6 November 2019; pp. 1452–1456. [CrossRef]

38. Hou, Y.; Liu, L.; Wei, Q.; Xu, X.; Chen, C. A novel DDPG method with prioritized experience replay. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; Volume 12, pp. 316–321. [CrossRef]
39. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. International conference on machine learning. *Proc. Mach. Learn. Res.* **2018**, *80*, 1587–1596. [CrossRef]