Intelligent Service Robotics (2019) 12:209–218
https://doi.org/10.1007/s11370-019-00279-6

ORIGINAL RESEARCH PAPER

Adaptation to environmental change using reinforcement learning for robotic salamander

Younggil Cho[1] · Sajjad Manzoor[2] · Youngjin Choi[3]

Received: 11 July 2018 / Accepted: 30 May 2019 / Published online: 10 June 2019

Abstract

In the paper, a reinforcement learning technique is applied to produce a central pattern generation-based rhythmic motion control of a robotic salamander while moving toward a fixed target. Since its action spaces are continuous and there are various uncertainties in an environment that the robot moves, it is difficult for the robot to apply a conventional reinforcement learning algorithm. In order to overcome this issue, a deep deterministic policy gradient among the deep reinforcement learning algorithms is adopted. The robotic salamander and the environments where it moves are realized using the Gazebo dynamic simulator under the robot operating system environment. The algorithm is applied to the robotic simulation for the continuous motions in two different environments, i.e., from a firm ground to a mud. Through the simulation results, it is verified that the robotic salamander can smoothly move toward a desired target by adapting to the environmental change from the firm ground to the mud. The gradual improvement in the stability of learning algorithm is also confirmed through the simulations.

Keywords Reinforcement learning · Adaptation to environmental change · Central pattern generator (CPG)

1 Introduction

With the advancement of technologies, robotic tasks have gradually become much more complex and difficult. In order to implement these difficult tasks in real environment, bio-inspired robots have been developed which are similar to human being or other animals. Different attempts to develop human or animal-like robots have resulted in a variety of applications in the different fields. Bipedal robots that walk like human and single-legged hopping robots can easily move in the environment with obstacle and uneven terrain [1]. Fish-like swimming robots are helpful in exploring ocean, river and different water reservoirs [2]. Snake robots and other crawling robots can move on land, go through confined spaces and swim in water [3,4]. Likewise, flying robots such as birds and some insects have been developed in [5,6].

Along with the construction of bio-inspired robots, there have also been researches on the development of algorithms for periodic locomotion of vertebrates in [7–10]. Among them, a salamander-like robot has been one of the intensive research topics for the locomotion of vertebrates in [11–13]. In particular, Ijspeert et al. proposed a numerical model of central pattern generator (CPG) for salamander with a spinal cord model composed of a network of coupled neural oscillators to drive a salamander robot in [14]. The salamander robot was developed in order to use it as a tool to experiment with biological hypotheses about evolution in animal locomotion, since the salamander is considered to resemble the

This work was supported by the Convergence Technology Development Program for Bionic Arm through the National Research Foundation of Korea Funded by the Ministry of Science, ICT & Future Planning (NRF-2015M3C1B2052811), Republic of Korea.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11370-019-00279-6) contains supplementary material, which is available to authorized users.

✉ Youngjin Choi
cyj@hanyang.ac.kr

Younggil Cho
brian111001@gmail.com

Sajjad Manzoor
sajjad.ee@must.edu.pk

[1] Korea Institute of Science and Technology (KIST), Seoul 02792, South Korea

[2] Faculty of Engineering, Mirpur University of Science and Technology (MUST), Mirpur, AJK 10250, Pakistan

[3] Department of Electrical and Electronic Engineering, Hanyang University, Ansan 15588, South Korea

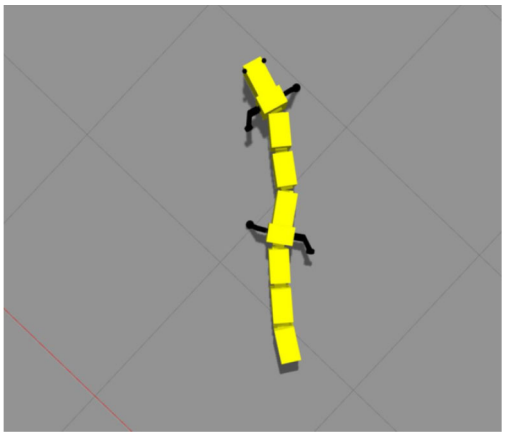


Fig. 1 Robotic salamander simulator made in the ROS and Gazebo environment

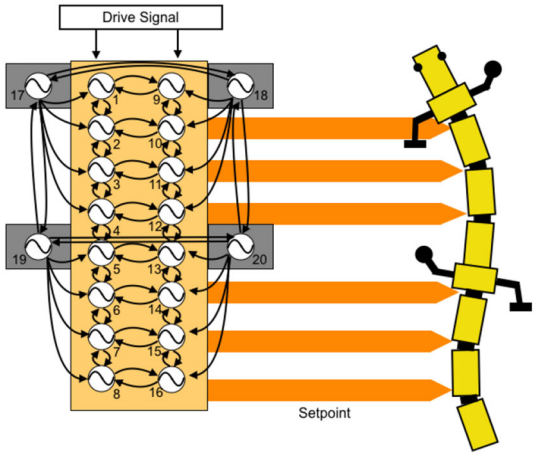


Fig. 2 The left side is the description of the CPG model suggested in [14], and the right side is the robotic salamander of Gazebo simulation used as an agent for reinforcement learning in the paper

first terrestrial vertebrates [15,16]. The effort to experiment with actual salamander-like robot can also be seen in [17].

In this paper, our major interest is to use the CPG model of robotic salamander designed for walking on firm ground surface [14] in a new environment that the robot has never experienced, such as mud. In order to adapt the CPG model parameters for robotic salamander to the new environment while it continues to walk toward the target, deep reinforcement learning is used. In particular, the drive signal used to determine both intrinsic frequency and amplitude of the CPG model is considered as the most important parameter for interaction of the robot with inexperienced ground environment.

The reinforcement learning is one of the learning schemes in the category of machine learning [18,19]. It makes the learner, called as an agent, learn from the interaction with environments in order to achieve any goal. Its general purpose is to design the controller, called as policy, for nonlinear systems. The reinforcement learning is becoming more important thanks to its scalability according to various learning objects and tasks. In the past, the reinforcement learning was applied only to the discrete and low-dimensional action spaces. According to the deep deterministic policy gradient (DDPG) algorithm developed in [20], it is recently extended to the continuous and high-dimensional action spaces, a category which many robotic researchers want to resolve. There have been a few surprising results, e.g., the references in [21–24] dealt with the control of high-dimensional systems using deep neural network as functional approximation of the reinforcement learning, namely deep reinforcement learning algorithm.

One of the critical issues that make control of robotic salamander more difficult is the uncertainties in real environment and its modeling. Due to the continuous environmental changes, it is hard to make an exact environment model itself. In order to resolve this issue, the deep reinforcement learning method is applied in this paper to generate parameters for mathematical model with uncertainties by a trial-and-error method. It is expected that the robot is able to learn how to overcome the uncertainties using a deep reinforcement learning method. Also the DDPG is applied to the robotic salamander in order to implement agent's adaptation for the gaits in new environment. For this purpose, the supervised learning algorithm is utilized to make the motion smooth and easy. It is intended not to make the robot (agent) learn from scratch by learning the CPG parameters of the already known environment in advance. The robotic salamander and the environments are implemented using the dynamic simulation tool called as Gazebo inter-lockable [25] with robot operating system (ROS), as shown in Fig. 1.

The remainders of the paper are organized in the following order; Sect. 2 introduces the related fundamentals regarding deep reinforcement learning and rhythmic motion generation. In Sect. 3, network structures, reward function and exploration strategy are described to explain the learning processes for rhythmic motion of robot toward a target. In Sect. 4, simulation results are given to reveal the advantages of deep reinforcement learning for adaptation to the changing environment. Finally, in Sect. 5, conclusions are drawn for the paper.

2 Backgrounds

In this section, we provide background materials required to implement deep reinforcement learning process using deep deterministic policy gradient. By applying to this method on the CPG model, parameters for the adaptation of rhythmic motion of the salamander robot can be generated for changing environment.

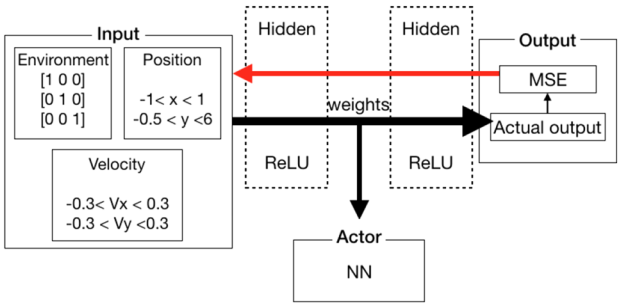


Fig. 3 Structure of the supervised pre-training network, where the black right arrow indicates the weight parameters in forward propagation phase and the red left arrow represents the back-propagation. After training, the weight parameters are cloned to the actor network

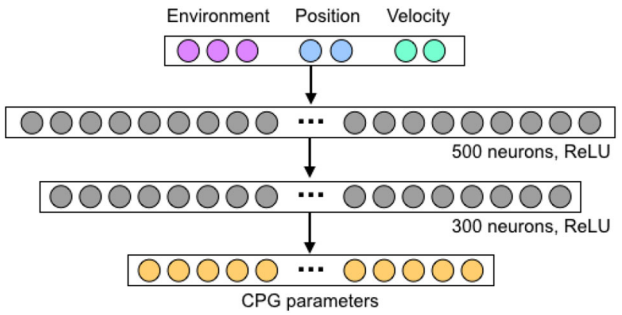


Fig. 4 Structure of actor network to determine the CPG parameters, where the inputs are one-hot vectors to denote environments, position and velocity of agent. The weights of actor network are cloned from those of the supervised pre-training network to produce the CPG parameters

2.1 Reinforcement learning

In reinforcement learning, the behavior that an agent performs in interaction with an environment E is referred to as a policy, $\pi : S \rightarrow A$. As the environment is stochastic, it is modeled as a Markov decision process with a state space S, action space A and reward function $r_{r(t)}$. A Markov decision process is a decision-making process where an agent interacting with an environment E changes its state s_t through the action a_t at each time step t. The agent receives the reward $r_{r(t)}$ according to the changed state s_{t+1} and its action a_t. The return, denoted by R_t, is defined to represent the sum of discounted future rewards with a discounting factor $\gamma \in [0, 1]$ as suggested in [18]:

$$R_t = r_{r(t+1)} + \gamma r_{r(t+2)} + \gamma^2 r_{r(t+3)} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{r(t+k+1)}. \tag{1}$$

The discount factor is introduced to prevent the expected return from going infinity. The value of action a that the agent takes with the policy π can be defined as the expected discounted return (cumulative reward) which begins with state s. It is termed as the action-value function, denoted by $Q^{\pi}(s, a)$, and it is given as [18]:

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= \mathbb{E}_{\pi}[R_t | s_t = s, a_t = a], \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{r(t+k+1)} | s_t = s, a_t = a\right]. \end{aligned} \tag{2}$$

In many approaches, the Bellman equation describing the optimal action-value function is used. It can be described as a function $\mu : S \rightarrow A$ if the target policy is deterministic:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{\mu}\left[r_{r(t)} + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))\right]. \tag{3}$$

The goal of reinforcement learning is to find the optimal policy which maximizes a cumulative reward. One of the methods for reinforcement learning in order to find the optimal policy is Q-learning[19] which makes use of the greedy policy $\mu(s) = \text{argmax}_a Q(s, a)$. Mnih et al. in [22,23] used the Q-learning algorithm with a deep neural network as function approximators. Moreover, they proposed a deep Q-network (DQN) by introducing two major changes such as a replay buffer and a target network.

2.2 Deep deterministic policy gradient (DDPG)

It is hard to apply Q-learning to continuous action spaces because the optimization is too slow to use in high-dimensional action spaces. The DDPG algorithm resolves these limitations by applying the actor-critic method to the deterministic policy gradient (DPG) algorithm [26]. The deterministic policy is formally considered such that $\mu_{\vartheta} : S \rightarrow A$ with parameters ϑ. The objective function for updating the actor policy network is defined as follows:

$$J(\mu_{\vartheta}) = \mathbb{E}_{\mu_{\vartheta}}[Q^{\mu}(s, a)|_{a=\mu_{\vartheta}(s)}]. \tag{4}$$

As a result, the weights of the actor network ϑ are updated by the following rule:

$$\begin{aligned} \vartheta_{t+1} &= \vartheta_t + \alpha \mathbb{E}_{\mu_{\vartheta}}[\nabla_{\vartheta} Q^{\mu}(s, a)|_{a=\mu_{\vartheta}(s)}], \\ &= \vartheta_t + \alpha \mathbb{E}_{\mu_{\vartheta}}[\nabla_a Q^{\mu}(s, a)|_{a=\mu_{\vartheta}(s)} \nabla_{\vartheta} \mu(s|\vartheta)], \end{aligned} \tag{5}$$

where α is the learning rate. The gradient of the objective function was obtained by applying the chain rule. The gradient of actor policy $\nabla_{\vartheta} \mu(s|\vartheta)$ moves according to the criteria given by the term $\nabla_a Q^{\mu}(s, a)|_{a=\mu_{\vartheta}(s)}$. This direction of the actor policy gradient is provided by the critic network. The critic network serves to evaluate the current policy through the action-value function. It is referred to as DDPG algorithm in [20] that combines the innovations of the DQN which recently performs successfully and the actor-critic method in [27] with deep neural approximators. The details on the

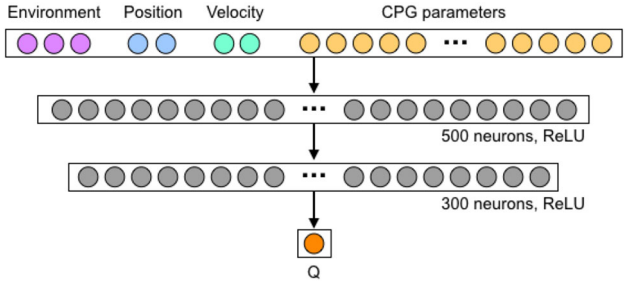


Fig. 5 Critic network yields one action value Q with linear activation function. For this, the inputs are taken as both the state s and the action $a = \mu_\vartheta(s)$, in which the state is composed of the one-hot vectors corresponding to environments, position and velocity, and the action is 26 CPG parameters corresponding to the output of actor network

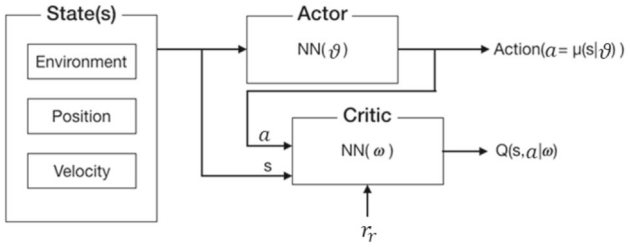


Fig. 6 Actor-critic policy gradient diagram for robotic salamander, where r_r denotes the reward, the output of actor network is a policy $a = \mu(s|\vartheta)$ and the output of critic network is an action-value function $Q(s, a|\omega)$. It is noted that actions a (here the CPG parameters) that are the outputs of actor network are selected to be the current policy $\mu(s|\vartheta)$

network structure and the learning process for salamander robot are explained in Sect. 3.

2.3 Central pattern generator model

CPG is a network of neural oscillators which generate rhythmic muscular activities that can be used by animals for periodic movements such as walking, breathing and cardiac activities [28]. A spinal cord model of CPG was presented in [14] which could be applied to the salamander robot in order to represent the evolutionary changes in animals from aquatic to terrestrial locomotion. Rhythmic outputs generated by the CPG are used for generating gaits in the salamander robot. The numerical CPG model for ith neural oscillator was presented in order to explain how the oscillators interact with adjacent ones in [14], and it is given by:

$$\begin{aligned} \dot{\theta}_i &= 2\pi v_i + \sum_j r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}), \\ \ddot{r}_i &= \rho_i \left(\frac{\rho_i}{4}(R_i - r_i) - \dot{r}_i \right), \\ x_i &= r_i(1 + \cos\theta_i), \end{aligned} \tag{6}$$

where θ_i and r_i are receptive phase and amplitude of ith oscillator, respectively, and three variables θ_i, r_i and $\dot{r}_i$ denote states of ith neural oscillator. Parameters v_i and R_i are the desired natural frequency and amplitude, respectively; $v_i = f(c_{v,1}, d, c_{v,0})$ and $R_i = f(c_{R,1}, d, c_{R,0})$ can be modulated by the drive signal d. The parameter w_{ij} denotes the coupling strength between oscillators i and j, while ϕ_{ij} is a phase difference between them. The parameter ρ_i is a positive time constant, and x_i is the nonnegative output of ith oscillator.

Figure 2 shows the network of CPG model proposed in [14] to be applied to the salamander robot. The CPG network is formed by the coupled oscillators for spinal cord, and a pair of the coupled oscillators of CPG represent excitatory and inhibitory neurons. The CPG in body consists of 16 oscillators divided into eight sets of the coupled oscillators, and the CPG for limb is composed of four oscillators, one for each limb. The oscillators are only coupled with the neighboring oscillators, e.g., the body oscillator 1 is coupled with the body oscillators 2 and 9 and the limb oscillator 17 as shown in Fig. 2. The difference between the outputs of excitatory and inhibitory oscillators determines the desired angle Θ_k for the robot joints in [14], and it is given as follows:

$$\Theta_k = \epsilon_k(x_{\text{left}} - x_{\text{right}}) \quad \text{for body motors,} \tag{7}$$

where ϵ_k implies a gain that increases linearly from head to tail, with $\epsilon_1 = 0.5$ for the head spine actuator and $\epsilon_6 = 1.0$ for the tail. On the other hand, the desired angles of the limb Θ_{k_L} motors directly depend on the phase of the corresponding oscillator given as:

$$\Theta_{k_L} = g\theta_i \quad \text{for limb motors,} \tag{8}$$

where $k_L = i - 16$ and g implies a gain that adjusts swing and stance time, which is taken as 1 for simplicity.

3 Methods

The DDPG algorithm that is very useful while dealing with continuous action space is combined with actor-critic method based on experience replay and target network method. The learning process of robot model is introduced by using DDPG in [20]; firstly, the actor-critic neural network is explained focusing on how the agent (robot) learns to achieve a specific goal, and then, the details of whole learning process are presented in the following sections.

3.1 Supervised pre-training

Reinforcement learning algorithm in the paper is a learning method for the agents to learn by itself such that it can arrive at specific goal. The goal of our agent, *i.e.*, robotic salamander, is to generate rhythmic motions in the body to reach a target

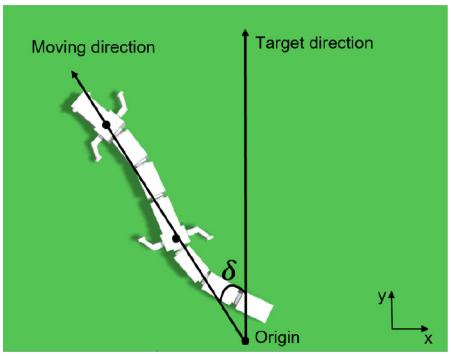
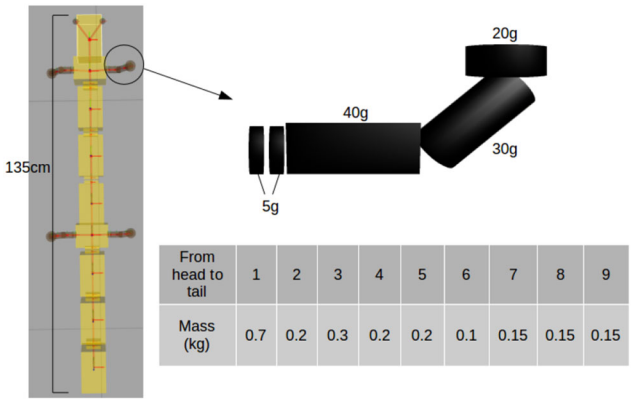


Fig. 7 Reward signal is defined as how close the agent moves to the target direction, where δ denotes the angle between the moving direction of agent and the direction from origin to target position



From head to tail	1	2	3	4	5	6	7	8	9
Mass (kg)	0.7	0.2	0.3	0.2	0.2	0.1	0.15	0.15	0.15

Fig. 8 Specifications of robotic salamander model used in Gazebo environment

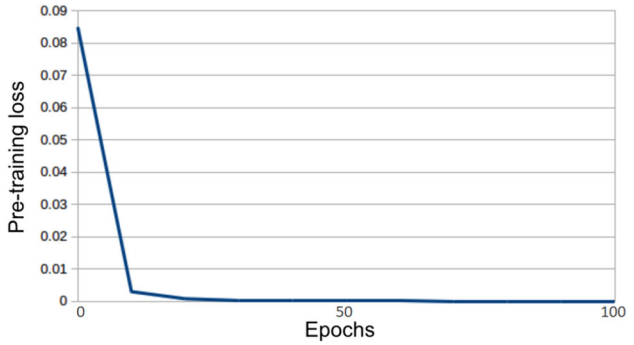


Fig. 9 Supervised pre-training loss is monotonically decreased for 100 training epochs

position despite of environmental change (from firm ground surface to the mud) including a new environment which the robotic salamander has never experienced. It means that the robotic salamander does not know the CPG parameters for walking in the new environment. Supervised learning algorithm [29] is used for giving an aid to the reinforcement learning algorithm. In order to apply the supervised learning algorithm, the dataset composed of the input and output pairs (X_i, Y_i) is required. The information about each environment is represented as one-hot vector. For example, the firm ground surface is represented as vector $[1, 0, 0]^{\mathrm{T}}$, while $[0, 1, 0]^{\mathrm{T}}$ and $[0, 0, 1]^{\mathrm{T}}$ represent new environments. In particular, $[0, 1, 0]^{\mathrm{T}}$ corresponds to the mud environment used in simulation. The position information of the agent is sampled in Cartesian $x - y$ coordinates. In the ranges of $x \in [-1, 1]$ and $y \in [-0.5, 6]$, whose unit is meter, 2000 data are randomly chosen. The planar velocities of agent denoted by v_x and v_y are also used as inputs to the network. The target position is set to a point 4 meters away from the initial position on the y-axis. The parameters of CPG model provided in [14] are used as labels of the supervised learning. After training the supervised learning neural network, the weight parameters of the network are cloned to the actor policy network of reinforcement learning algorithm. Thus, the structure of pre-training network becomes equal to that of the actor network. The configuration of pre-training network is shown in Fig. 3.

3.2 Actor-critic network

Just like the pre-training network, the one-hot vectors corresponding to environment, position and velocity of the robotic salamander are used as network inputs. Also, the network yields 26 CPG parameters to be used in the CPG model. The structure of actor network is shown in Fig. 4. It is assumed that there is a classifier which converts the information of environments into one-hot vector. The CPG parameters as the outputs of network correspond to the action since it makes the action at the end. These normalized CPG parameters are for the network training, while the range of the parameters is shifted when making actions. The experiences of model are stored in replay buffer for the experience replay. The weights of actor network are initialized as those of the supervised pre-training network.

The critic network determines the direction that the actor policy is updated. The critic network has the same structure of actor network except the input and output layers as shown in Fig. 5. The inputs of critic network are taken as both the state s composed of the one-hot vectors corresponding to environments, position and velocity of agent, and the action $a = \mu_\vartheta(s)$ of the CPG model parameters corresponding to the output of actor network. The critic network has one output neuron with linear activation function for the action value Q. The critic network is updated from the gradients obtained from the temporal difference (TD) error signal [18]. The TD error signal is calculated from the outputs of critic network when the current and next states taken out from the replay buffer are used as the inputs. However, directly updating the actor and critic networks may cause them to be unstable as well as the learning algorithms to be diverged. It can be resolved by using the separate target network in

Table 1 Trained 26 CPG model parameters on firm ground and in mud, respectively

	On the firm ground surface	In the mud
$[\epsilon_1, \epsilon_2, \epsilon_2, \epsilon_4, \epsilon_5, \epsilon_6]$ for body	[0.500, 0.600, 0.700, 0.799, 0.899, 1.00]	[0.524, 0.627, 0.732, 0.837, 0.942, 1.04]
ρ_i	19.995	20.96
d	3.01	2.49
$[c_{v,1}, c_{v,0}]$ for body	[0.199, 0.301]	[0.21, 0.314]
$[c_{v,1}, c_{v,0}]$ for limb	[0.199, 0.0]	[0.19, 0.0]
$[c_{R,1}, c_{R,0}]$ for body	[0.065, 0.196]	[0.068, 0.205]
$[c_{R,1}, c_{R,0}]$ for limb	[0.1305, 0.1306]	[0.128, 0.128]
$[w_{ij}, \phi_{ij}]$ downwards in body CPG	[10.00, −0.7833]	[10.47, −0.675]
$[w_{ij}, \phi_{ij}]$ upwards in body CPG	[9.998, 0.784]	[10.3, 0.7]
$[w_{ij}, \phi_{ij}]$ contralateral in body CPG	[10.001, 3.137]	[10.3, 3.137]
$[w_{ij}, \phi_{ij}]$ from limb to body CPG	[30.006, 3.135]	[31.33, 3.10]
$[w_{ij}, \phi_{ij}]$ within the limb CPG	[10.001, 3.136]	[10.461, 2.76]

order to generate the TD target y_i. The TD target and the loss function for the critic network have the following form:

$$y_i = r_{r(i)} + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\vartheta')|\omega'), \tag{9}$$

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\omega))^2, \tag{10}$$

respectively, where N is the minibatch size. From the structure of critic network, we can get the direction of the actor policy by calculating the gradient of the output Q of critic network with respect to the action, namely $\nabla_a Q^{\mu}(s, a)|_{a=\mu_\vartheta(s)}$. In the DDPG algorithm, the update of target network is similar to that of deep Q-network, but it is modified to a small amount for actor-critic method [20]. The target network is updated not directly by copying the weights, but by using soft update like $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$. This improves the stability of learning algorithm by changing the target value slowly. After updating the critic network by minimizing the loss function, the actor network is updated according to the direction suggested by the critic network. This actor-critic method is used with the experience replay and the separate target network.

The Adam descent[24] is used with the learning rate of 10^{-4} and 10^{-3} for the actor and the critic, respectively. For the critic network, a discount factor of $\gamma = 0.99$ is applied and $\tau = 0.001$ is used for the soft target updates. The activation functions of hidden layers are the rectifier linear unit(ReLU) functions [30]. On the other hand, the activation function of each output node of the actor is selectively used either the sigmoid function or the hyperbolic tangent function. For instance, the hyperbolic tangent functions are used to determine CPG parameters related to the angle scaling between $[-1, 1]$, and the sigmoid functions are used for the remainders of the parameters. The networks are composed of two hidden layers with 500 and 300 neurons, respectively. The networks are trained with minibatch size of 64, and a replay buffer size is 10^6.

3.3 Reward function

Figure 6 illustrates whole learning processes for the locomotion of robotic salamander in the inexperienced environment. To begin with, the CPG parameters which are the outputs of actor network are chosen as the current policy. The reward $r_{r(t)}$ and the next state s_{t+1} are observed after executing actions, and these one-step dynamics of the agent $(s_t, a_t, r_{r(t)}, s_{t+1})$ are stored in replay buffer. The actor network uses this replay buffer in order to select the action following the policy, and the critic network uses it for calculating TD error and policy gradient. The samples stored in replay buffer are taken out by minibatch size. In particular, the reward signal $r_{r(t)}$ is most important part of the reinforcement learning algorithm. It determines the performance to arrive at the goal of agent and the convergence rate of learning algorithm.

The goal of agent is to reach the target position in given map including inexperienced ground condition. Let us consider the motion of agent shown in Fig. 7. At every time step, the positions of agent used as inputs of networks are calculated from two girdle positions. The girdle is a sort of virtual link between body segments having limbs as shown in Fig. 7. It is expected that the girdle (or moving) direction is aligned with the target direction. In order to make the agent follow the target direction, the reward signal $r_{r(t)}$ is defined as follows:

$$r_{r(t)} = V\cos\delta - V|\sin\delta|, \tag{11}$$

where V is the speed of robotic salamander and δ is an angle between the moving direction and target direction. The

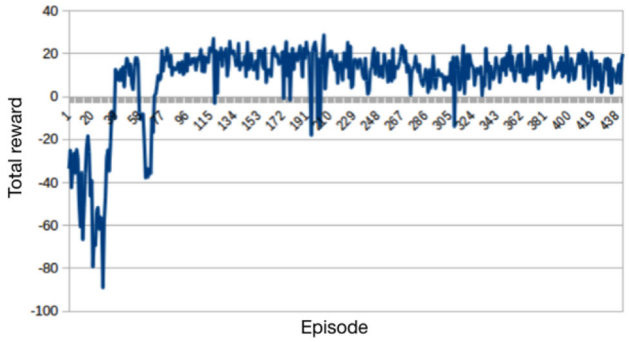


Fig. 10 Sum of total rewards per each episode while training, where it is seen that the agent finds a reasonable policy within 100 episodes. Although the initial rewards were noisy and low due to the exploration effect, the constant rewards mostly maintained after conducting 100 episodes

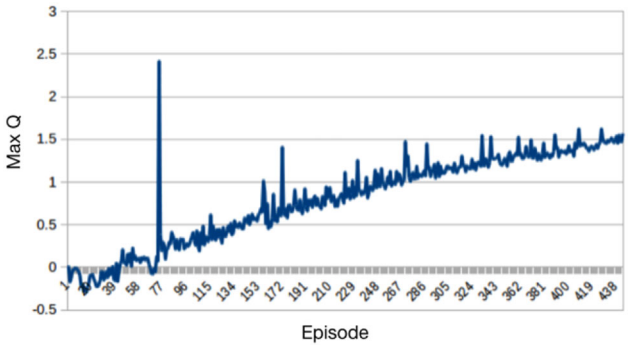


Fig. 11 Average of maximal Q values per episode, where it is confirmed that the actor policy is updated in the right direction because Q value increases according to the episode progress. Also its gradual increase implies the improvement in the learning algorithm stability [23]

reward function given above implies how close the agent moves to the target direction. The first term $V \cos\delta$ means that the closer both moving and target directions each other, the bigger reward it gets. The second term $V|\sin\delta|$ reduces the reward as the moving direction is deviated from the target. This reward signal results in the agent moving toward the target position and finding the suitable CPG parameters when the agent encounters the new environment.

3.4 Action noise

There always exists the exploration *vs* exploitation dilemma in the reinforcement learning problem. Exploitation means that the agent makes the best decision if current information is known, and exploration helps in gathering much information. Proper exploration should be provided for the agent in order to improve the performance of reinforcement learning algorithm. Here, Ornstein–Uhlenbeck process in [20] is used as an action noise to conduct the exploration. It is a stochastic process which has mean-reverting property, and it is given by the differential equation:

$$\mathrm{d}x_t = \beta(\mu - x_t)\mathrm{d}t + \sigma \mathrm{d}W_t, \tag{12}$$

where β denotes the rate at which the variables revert toward the mean, μ is the mean, σ is the degree of volatility of the process and $\mathrm{d}W_t \sim \mathcal{N}(0, \mathrm{d}t)$ follows standard Wiener process. In this paper, small exploration regarding the drive signal is chosen, and thus, the action noise of drive signal is set such that $\mu = 0.3$, $\beta = 0.5$, and $\sigma = 0.1$.

4 Simulation

In order to implement an agent, dynamic simulator named Gazebo [25] was used. Figure 8 illustrates 135-cm-long robotic salamander model used as the agent. It is composed of simple polytopes like boxes, spheres and cylinders provided in the Gazebo. The body spine was made by connecting the box shapes in a line, and six hinge joints were located between body links. The continuous rotating joints unlike real salamanders connect the body and the limb. The data regarding each link mass are given in Fig. 8 as well.

The robotic salamander implemented in Gazebo was controlled by using well-known proportional-derivative (PD) controller [31] as suggested in [14], where the errors between the desired joint configurations determined by neural oscillators based on the CPG model and the actual joint configurations are applied to the PD controllers. The ROS enables connecting between Gazebo simulation and CPG model. The robotic joint configuration messages are published to ROS topic, and then, each joint of robotic salamander in Gazebo simulation subscribes the messages on ROS topic in order to drive the joint PD controller. Since the Gazebo is also able to publish the position and velocity of robotic salamander, these information could be used as the states of agent for actor-critic networks. The task of an agent in this simulation is to pass through the firm ground surface, subsequently to enter the mud and then to reach the final target position located inside the mud. The mud environment was also provided by the Gazebo simulation.

4.1 Results of supervised pre-training

The purpose of the supervised pre-training is to provide the trained weight parameters with the actor network of deep deterministic policy gradient algorithm. The pre-training network yields the CPG model parameters. Since the structure of the supervised pre-training network is same as that of the actor network, the input layer has seven neurons including three neurons for the information regarding environments and four neurons for the sampled position and velocity of the agent, and there are 26 neurons in the output layer for the

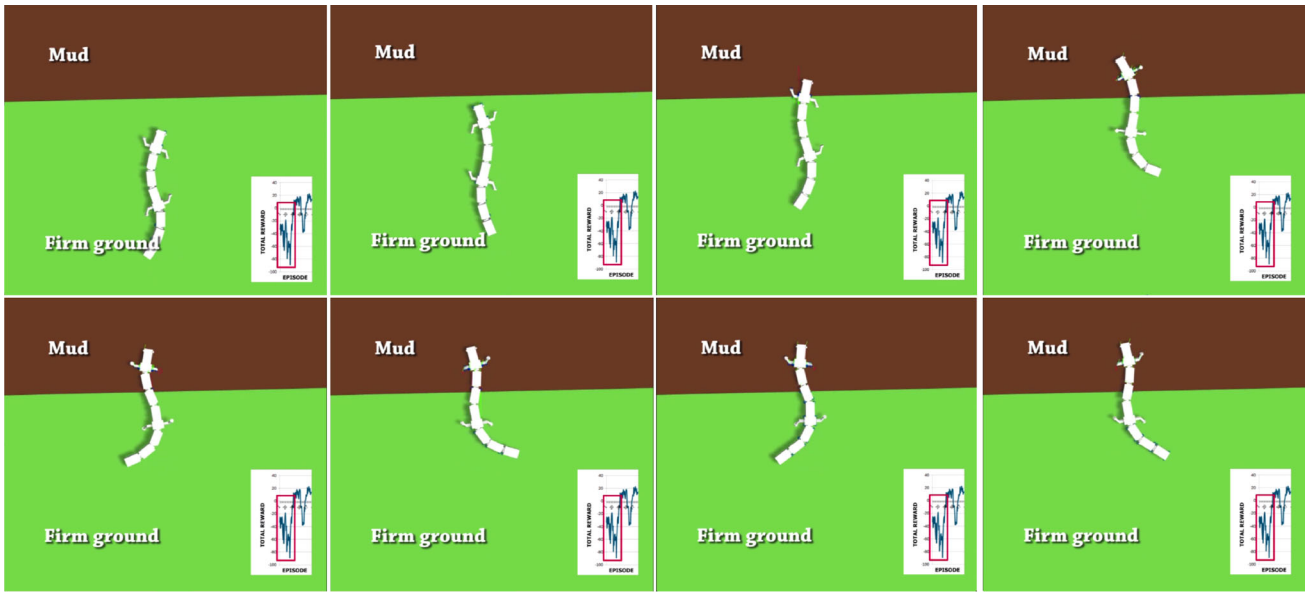


Fig. 12 Sequential snapshots of robotic salamander motion without reinforcement learning, from on the firm ground to the mud using Gazebo simulator

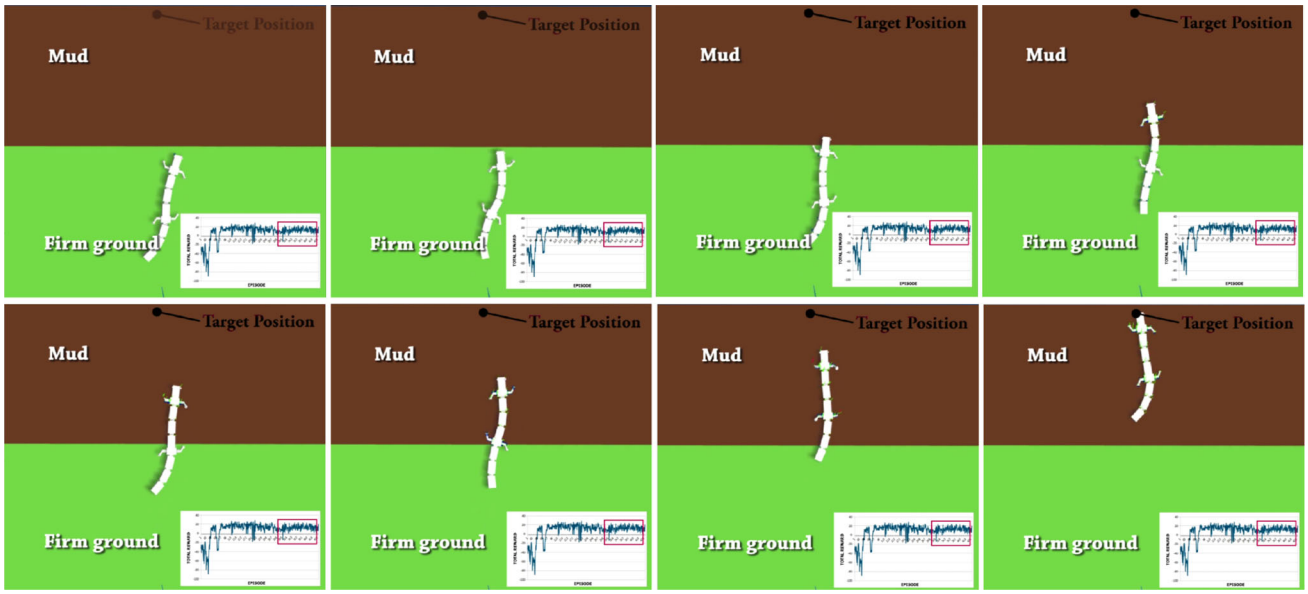


Fig. 13 Sequential snapshots of robotic salamander motion with reinforcement learning, from on the firm ground to the mud using Gazebo simulator

CPG parameters. The loss defined as the mean squared error function is minimized while training. The pre-training loss in 100 training epochs is shown in Fig. 9. The loss function gets closer to zero as the epoch step increases. By applying the supervised learning, the agent can move forward on the firm ground surface with its suitable CPG parameters while the reinforcement learning is conducted. The only thing left for the agent is to learn about the new environment never experienced by it while moving.

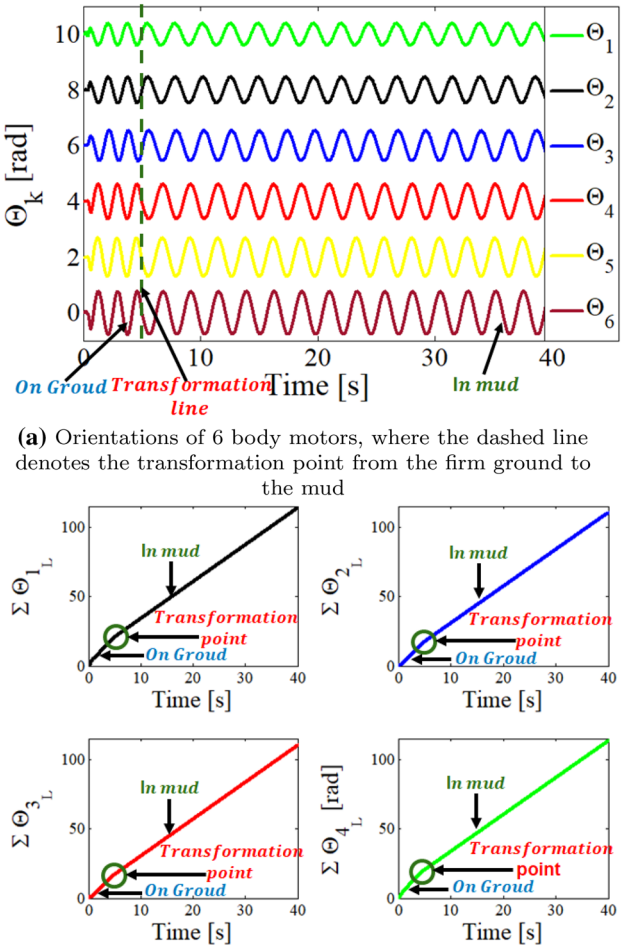
4.2 Learning result

One of the research aims of this paper is to analyze how the CPG parameters are varied when the agent interacts with new environment. The average CPG parameters produced by the trained actor network in the mud as well as on the firm ground surface are listed in Table 1. These parameters were used by the agent to perform the task successfully. The CPG parameters on the firm ground surface were similar to those provided in [14]. For the inexperienced mud environment, all

the parameters suggested in Table 1 were changed to some extent; however, we are to focus on the drive signal d. The trained actor network yielded the average drive signal as 2.49 in the mud as given in Table 1. It is seen that the agent has chosen a policy of moving slowly by lowering the drive signal due to the increases in friction and resistive force caused by the environmental change from the firm ground to the mud. As a matter of fact, it was quite different from our expectation that the robotic salamander would walk quickly with higher drive signal because it receives negative rewards over time it spends. Also the phase differences within limb CPG in the mud ϕ_{ij} were significantly lower than those on the firm ground surface in Table 1. Both the drive signal and the phase differences within limb CPG were the important parameters in adapting to the mud environment.

The robotic salamander learns the policy updated in the direction that the critic network suggests. As mentioned before, the output of critic network is the action-value function Q which represents the expected sum of future rewards. Also, the critic network is updated by the reward per time step. In order to show the progress of agent, the results of learning can be seen by checking both the total reward that the agent receives in each episode and the maximum Q value. It is noted that the episode is terminated when the robot moves for 1 meter or the total reward arrives at 40. Figure 10 shows the total reward for each episode. It can be seen that the agent finds a reasonable policy within 100 episodes. Although the initial total rewards were very noisy and low due to the exploration effect, the total rewards were mostly maintained at near 20 after conducting 100 episodes. As more substantial metric to assess the learning capability, Fig. 11 shows the average of maximal Q values in each episode. Since the Q value increases according to the episode progress, it is confirmed that the actor policy is updated in the right direction; furthermore, its gradual increase implies that the stability of learning algorithm has been improved [23].

Figure 12 illustrates the sequential snapshots for robotic salamander when the reinforcement learning is not applied. It can be seen that, by using same CPG parameters given in [14], the robot moves smoothly toward the target on the firm ground surface by using sinusoidal waveform-like motion of body and periodic motions of limbs. When the robot enters into the mud, its forward motion is seized, although the body and limbs keep on moving with same rhythmic behavior. The sequential snapshots regarding the motions of robotic salamander using the reinforcement learning are shown in Fig. 13. The robotic salamander starts its rhythmic motion on the firm ground surface, and then, it enters in the mud. It can be seen that the robot moves smoothly in both the environments. Another observation is that the robot utilizes its body as well as the limbs in both the environments. This is quite different from the experiments conducted in [14] in that only the body was used while swimming in [14], but both



(a) Orientations of 6 body motors, where the dashed line denotes the transformation point from the firm ground to the mud

(b) Cumulative orientations of 4 limb motors, where big circles denote the transformation points from the firm ground to the mud

Fig. 14 Orientations of six body motors and four limb motors when the CPG parameters obtained by applying the DDPG are used, where we can observe that **a** rhythmic movements of the body motors become slow accordingly as it enters into the mud from the firm ground and **b** the slopes of the cumulative orientations of all the limbs become small and, in other words, the speeds of all the limbs become slow accordingly as it enters into the mud from the firm ground

body and limbs are all used while moving in the mud. Thus, it can be concluded that the reinforcement learning was able to help in adapting to the inexperienced new environment in such a way that both body and limbs remain in action.

Figure 14 shows the orientations of all the body and limb motors when the robotic salamander enters into the mud from the ground with the CPG parameters obtained by using DDPG. Figure 14a shows that the rhythmic motions of body motors become slow entering into the mud from the ground as we can see in the frequency changes in waveforms. On the other hand, since the limb motors use continuous rotational motions instead of to-and-fro motions during the gait, the cumulative orientations of limbs are suggested in Fig. 14b. Looking at the slopes of the cumulative orientations of

the limbs, it can be seen that the speeds of limbs become slow accordingly as it enters into the mud from the ground, because the slopes for the accumulated orientations are low as shown in Fig. 14b. The detailed explanations are also given for clarity in Fig. 14 in the paper.

5 Conclusions

In this paper, the CPG motion of salamander was implemented on a robot such that it could adapt to new environment. The robot, which is also called agent, learns information about specific environment through the supervised pre-training. The firm ground surface was taken as the known environment for locomotion, while the mud was later provided as the inexperienced new environment. In Gazebo simulation, the robotic salamander has moved adapting and interacting with the environment by changing the CPG model parameters. The robot learned by trial and error with a deep reinforcement learning algorithm and overcame the environmental uncertainties while moving from one environment to another.

It was shown that the robot smoothly reached the target position using the CPG model parameters learned by the reinforcement learning method. Although the robot has continuous and complex action spaces, it could learn a policy well in the direction that the critic network suggests, and it was able to carry out the task as well. From the results, it was also concluded that the DDPG was a suitable learning algorithm to control the rhythmic locomotion and its scalability to the robot in changing environment was very large. Furthermore, it could be seen that the drive signal to modulate whole motions was decreased when the robot entered into the mud environment from the firm ground.

References

1. Zhou X, Bi S (2012) A survey of bio-inspired compliant legged robot designs. Bioinspir Biomim 7(4):041001
2. Raj A, Thakur A (2016) Fish-inspired robots: design, sensing, actuation, and autonomy—a review of research. Bioinspir Biomim 11(3):031001
3. Hirose S, Yamada H (2009) Snake-like robots [tutorial]. IEEE Robot Autom Mag 16(1):88–98
4. Koh J-S, Cho K-J (2013) Omega-shaped inchworm-inspired crawling robot with large-index-and-pitch (LIP) SMA spring actuators. IEEE/ASME Trans Mechatron 18(2):419–429
5. Paranjape AA, Chung S-J, Kim J (2013) Novel dihedral-based control of flapping-wing aircraft with application to perching. IEEE Trans Robot 29(5):1071–1084
6. Chen Y et al (2015) Hybrid aerial and aquatic locomotion in an at-scale robotic insect. In: IEEE/RSJ international conference on intelligent robots and systems (IROS)
7. Marder E, Bucher D (2001) Central pattern generators and the control of rhythmic movements. Curr Biol 11(23):R986–R996
8. Ijspeert AJ (2008) Central pattern generators for locomotion control in animals and robots: a review. Neural Netw 21(4):642–653
9. Yu J et al (2014) A survey on CPG-inspired control models and system implementation. IEEE Trans Neural Netw Learn Syst 25(3):441–456
10. Manzoor S, Choi Y (2016) A unified neural oscillator model for various rhythmic locomotions of snake-like robot. Neurocomputing 173(3):1112–1123
11. Frolich LM, Biewener AA (1992) Kinematic and electromyographic analysis of the functional role of the body axis during terrestrial and aquatic locomotion in the salamander Ambystoma tigrinum. J Exp Biol 162(1):107–130
12. Ashley-Ross MA, Bechtel BF (2004) Kinematics of the transition between aquatic and terrestrial locomotion in the newt Taricha torosa. J Exp Biol 207(3):461–474
13. Delvolve I, Bem T, Cabelguen J-M (1997) Epaxial and limb muscle activity during swimming and terrestrial stepping in the Adult Newt, *Pleurodeles waltl*. J Neurophysiol 78(2):638–650
14. Ijspeert AJ et al (2007) From swimming to walking with a salamander robot driven by a spinal cord model. Science 315(5817):1416–1420
15. Cohen AH (1988) Evolution of the vertebrate central pattern generator for locomotion. In: Cohen AH, Rossignol S, Grillner S (eds) Neural control of rhythmic movements in vertebrates. Wiley
16. Gao K-Q, Shubin H (2001) Late Jurassic salamanders from northern China. Nature 410(6828):574
17. Crespi A et al (2013) Salamandra robotica II: an amphibious robot to study salamander-like swimming and walking gaits. IEEE Tran Robot 29(2):308–320
18. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
19. Watkins C, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292
20. Lillicrap TP et al (2015) Continuous control with deep reinforcement learning. arXiv preprint arXiv: 1509.02971
21. Silver D et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489
22. Mnih V et al (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602
23. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nature. https://doi.org/10.1038/nature14236
24. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
25. Koenig N, Howard A (2004) Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Proceedings of 2004 IEEE/RSJ international conference on intelligent robots and systems
26. Silver D et al. (2014) Deterministic policy gradient algorithms. In: Proceedings of the international conference on machine learning
27. Konda VR, Tsitsiklis JN (2000) Actor-critic algorithms. In: Conference on neural information processing systems (NIPS), pp 1008–1014
28. Hooper SL (2000) Central pattern generators. Curr Biol 10(5):R176–R179
29. Bishop CM (2006) Pattern recognition and machine learning (information science and statistics). Springer, Berlin
30. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the international conference on machine learning
31. Choi Y, Chung WK (2004) PID trajectory tracking control for mechanical systems. Springer, Berlin