Integration of Sensory Feedback into CPG Model for Locomotion Control of Caterpillar-like Robot

Guoyuan Li, Wei Li and Houxiang Zhang *Faculty of Maritime Technology and Operations*
Aalesund University College
Postboks 1517, N-6025, Aalesund, Norway
{guli, weli, hozh}@hials.no
Jianwei Zhang *Department of Computer Science*
University of Hamburg
Vogt-Koelln-Strasse 30, 22527, Hamburg, Germany
zhang@informatik.uni-hamburg.de

Abstract

This paper presents an approach of sensory feedback integration into a CPG model under a hierarchical control architecture for adaptive caterpillar-like locomotion. Motivated by the simplicity of CPG models in gait generation, a sinusoidal generator is employed as the low level controller of a caterpillar-like robot. To regulate the behavior of the robot with respect to the changes of environmental conditions, a behavior adaptor is designed to integrate sensory feedback into the sine-based model, so that sensory input can be transferred into the model and further affect the output of the model to some extent. Meanwhile, a policy gradient based reinforcement learning method is adopted at the high level, aiming to learn the mapping between sensory input and reasonable responses of the robot. The optimized policy could be obtained after episodic learning. Simulation results show the caterpillar-like robot can climb over uneven terrains with the help of the sinusoidal generator and the learned policy, which verifies the effectiveness of the proposed approach.

I. INTRODUCTION

Last century has verified that animal's locomotion is controlled by a special neural circuit called central pattern generators (CPG) in the spinal cord of vertebrates. Without any rhythmic inputs, CPG can produce coordinated patterns of rhythmic activity [1]. Although the underlying mechanisms of CPG are not yet fully understood, the concept has been widely used in control technology communities.

In robotics, CPG is usually applied on biologically inspired robots for gait generation. A lot of CPG models have been developed in recent years by combining biological features and mathematical properties. From [2], there are three types of CPG models based on the abstraction from neurobiology. The first type retains and follows most biological mechanisms. Herrero-Carrón et al. developed such a CPG model based on recently revealed strategies of living CPGs [3]. The second type is much simpler that it only keeps the concept of neurons and synapses. The connections between neurons play a key role in output generation. This type of CPG models is widely used in bio-inspired robots, such as the work by Ekeberg [4], Kimura [5] and Ma [6]. The last type belongs to pure mathematical models, such as phase oscillators [7] and sinusoidal generators [8][9].

This work is partly supported by a grant from German Research Foundation (DFG no. U-4604-DFG-1001)

To date, although CPG is considered an elegant solution for gait generation, there is no universal rules to directly use CPG models to achieve more complex tasks such as localization, path planning, navigation and adaptive locomotion. But in the literature, a good number of novel intelligent control methods such as fuzzy control, genetic algorithm and neural network control were proposed to bridge the gap. These methods can optimize control parameters in CPG models by maximizing a scalar evaluation via the interaction with the environment. Hasanzadeh proposed a fuzzy logic tuner to optimize the locomotion of a snake-like robot when it moving on terrains with different friction coefficients [10]. Kamimura et al. developed ALPG software to seek efficient locomotion patterns automatically for a modular robot, in which a genetic algorithm is used to optimize the CPG network [11]. Tomoyuki et al. proposed a CPG-based reinforcement learning (RL) to improve the energy efficiency of a biped robot [12]. They succeeded to find out the optimal torque-free period during the bipedal walking. Snel et al. used RNN-based CPG and hierarchical RL on a simulated hexapod robot to realize dynamic walking on terrains with varying complexity [13]. Although adaptive locomotion has been realized on variety of robots [14]–[18], it is seldom to see any applications for limbless robots, especially for caterpillar-like robots. Moreover, how to integrate sensory feedback into CPG models for caterpillar-like robots is also seldom addressed in the literature.

The research presented in this paper is related to our ongoing DFG project “Biologically Inspired Modular Climbing Caterpillar Robot Using Passive Adhesion” (BICCA). In the project, we combine CPG properties with the concept of a modular robot to create a novel climbing caterpillar-like robot [19]. A hierarchical control architecture including a CPG model with caterpillar locomotion features and a learning algorithm for sensor-servo-based behavior control is designed for the control system. In this paper, we concentrate the effort on integrating sensory feedback into a CPG model based on a hierarchical control architecture to realize adaptive caterpillar-like locomotion. The contribution of this paper is twofold. First, in the low control level, a solution for sensory feedback integration into a sinusoidal generator is proposed. Thus the low level controller not only keeps its simplicity but also provides interface for high level control. Second, simulation has been implemented to verify the effectiveness of the proposed control structure in realizing adaptive locomotion for caterpillar-like robots.

The paper is organized as follows. Section II describes the overall structure of the hierarchical control system. In Section III, all the control components including the sine-based model, the process of sensory information, the method to integrate sensory feedback and how the policy gradient based reinforcement learning works will be presented in detail. Section IV presents simulation results. Conclusion and future work are drawn in section V.

II. Hierarchical Control Structure

The prior knowledge shows that a CPG model is suitable for gait generation, while an intelligent learning method is able to deal with environmental changes. Therefore, to achieve adaptive caterpillar-like locomotion, an intuitive way is to develop hierarchical control architecture by combining the CPG model with the learning method.

Fig. 1 illustrates the whole control architecture for adaptive locomotion. It consists of two levels. At the low level, a CPG model is employed in the locomotion control component so as to generate gaits for the robot. Taking advantages of the onboard sensors of the robot, raw sensor data are gathered during locomotion. All the raw sensor data are passed to a sensor filter, from where they are converted into reasonable sensory information.

At the high level, a learning method is applied to learn the mapping between the sensory information and the sensory input to the CPG model. An optimized policy will be obtained after repeating learning process. Combining the ability of online modulation of the CPG model, a behavior adapter component is

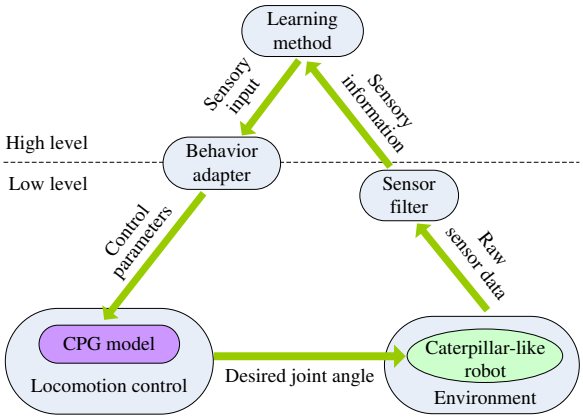

Fig. 1. The overall hierarchical control structure.

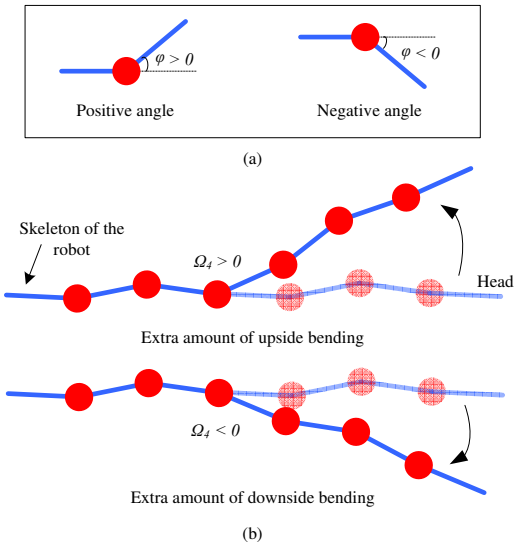

Fig. 2. The definition of angle position and the effect of offset setting.

used to bridge the CPG model and the learning method. It plays a role in changing the control parameters of the CPG model in a smooth manner. In this way, the learned policy will affect the CPG output, resulting in the adaptation of the robot to any environmental changes.

III. Implementation of Control System

The following will present the implementation of each component in the control system in detail.

A. Locomotion Control

To generate linear gait for a caterpillar-like robot with multiple degrees of freedom, we apply the sinusoidal generator as the controller of the robot [8]. The sinusoidal generator is employed for each module of the robot to generate linear gait. It describes the bending angle of the corresponding module in function of time. The dynamics of the *i*th sinusoidal generator are as follows:

$$\varphi_{(i,t)} = A_i \cdot sin(\frac{2\pi}{T_i}t + i\theta) + \Omega_{(i,t)} \tag{1}$$

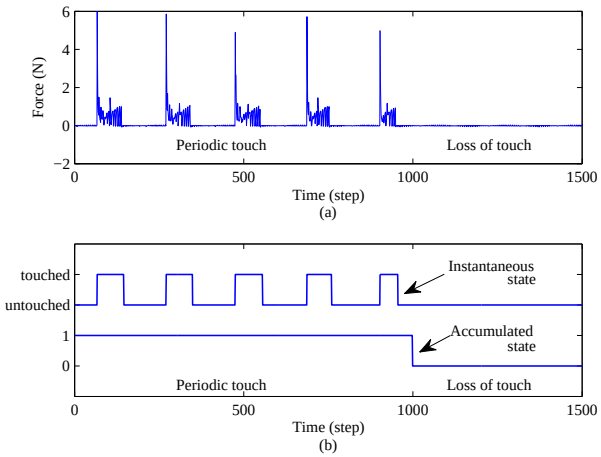

Fig. 3. Sensor data processing.

where φ is the desired angle of the ith module at the tth time step; A, T, θ and Ω donate the amplitude, the period, the phase difference and the offset of the sinusoidal generator, respectively.

Note that the offset parameter Ω for the sinusoidal generator is a function of time in (1). This indicates the parameter Ω can be considered as an adaptive variable modifying the caterpillar-like robot's body shape during locomotion. Fig. 2 illustrates how it affects the body shape of the robot: a positive value of Ω causes an extra amount of upside bending, while a negative value of Ω results in downside bending. According to the phenomenon, an adaptive controller is proposed to properly change the parameter Ω when the caterpillar-like robot moving on uneven terrains. The purpose is to increase the robot's contact areas and friction, promoting its stability and adaptability. The following sections will introduce how to combine this parameter with sensor information to achieve adaptive locomotion.

B. Sensor Filter

Assume touch sensors are installed at the bottom part of the caterpillar-like robot, which is similar to the function of prolegs of living caterpillars.

As the robot performs a linear gait during the locomotion, each module of the robot will move up and down periodically. Correspondingly, its touch sensor will tap on the terrain regularly. But this will become false when the robot moving on uneven terrains. Because some modules of the robot will lose contact against environmental changes. To sketch out the terrain via these touch sensors, it is necessary to identify whether a module is "periodic touch", or "loss of touch". The sensor filter component play such a role in handling the raw sensor data and identify the two module states.

Here we take an example to explain how the sensor filter component works. Considering a case when the caterpillar-like robot moving from a flat terrain to a slope, the robot will get stuck during the climbing process due to its rigid structure and the unchanged linear gait. Therefore, the internal modules of the robot will undergo the "periodic touch" state and the "loss of touch" state in succession, as shown in Fig. 3(a). The spikes of the force measured by the touch sensor indicate the contact of the corresponding module.

The sensor filter component takes two steps to identify the state changes of each module. First, it

converts the raw data to an instantaneous contact value by a threshold function:

$$s_{(i,t)} = \begin{cases} 1 & \text{if} \quad R_{(i,t)} > \Theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where s represents the instantaneous contact value of the ith module at the tth time step; R is the raw sensor data; and Θ is the threshold.

Then the sensor filter component accumulates the instantaneous contact values so as to form module states:

$$S_{(i,t)} = \begin{cases} 1 & \text{if} \sum_{t'=t-kT}^{t} s_{(i,t')} > 0 \quad (k \geq 1) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where S is the module state; T represents the number of time steps in one period of the linear gait; k is a constant that controls how long the passed instantaneous contact states are considered.

Fig. 3(b) shows the result after the process of the sensor filter component. Note that a value of 1 for S indicates that the module is touching the terrain periodically, whereas a value of 0 means the module loses contact with the terrain.

C. Behavior Adapter

The behavior adapter component connects the CPG model in the low level and the learning method in the high level (see details in Fig. 1).

As mentioned in Section III-A, the offset parameter Ω in (1) is responsible to modify the body shape of the caterpillar-like robot, so as to adapt it to environmental changes. However, abrupt change of Ω can cause the robot to generate jerk behaviors or even get damaged. To avoid such a result, the sensory input generated by the learning algorithm is required to transmit to the CPG model in a smooth manner. Therefore, the dynamics of Ω is designed as a leaky integrator:

$$\tau \dot{\Omega}_i = -\Omega_i + A \cdot \lambda_i \tag{4}$$

where λ is the sensory input from the high level; τ is a time constant that controls the afferent speed of λ; and A is the amplitude of the sinusoidal generator. To avoid the excess of maximum angle, λ is designed in range of [-1, 1].

The leaky integrator works in two opposite ways. On the one hand, when the sensory input λ is non-zero, the leaky integrator will leak a small amount of sensory input to Ω in the CPG model with a speed of τ, until the total amount of afferent sensory input approaching λ. On the other hand, if a zero value of sensory input is generated by the learning algorithm, the leaky integrator will gradually remove the effect of sensory input on Ω and finally recover the distorted linear gait back to normal state.

As a result, the offset parameter Ω enables the robot to smoothly shift any extra amount of bending between the zero value and the desired value, enabling the robot to modify its body shape according to the result from high level.

D. Reinforcement Learning

On the high level, a learning method is used to find the proper mapping from the current module states to the sensory inputs. Considering that the modules states have discrete values 0 and 1, while the sensory inputs are continuous and as defined bounded in the range of [-1, 1], it is preferred to choose direct

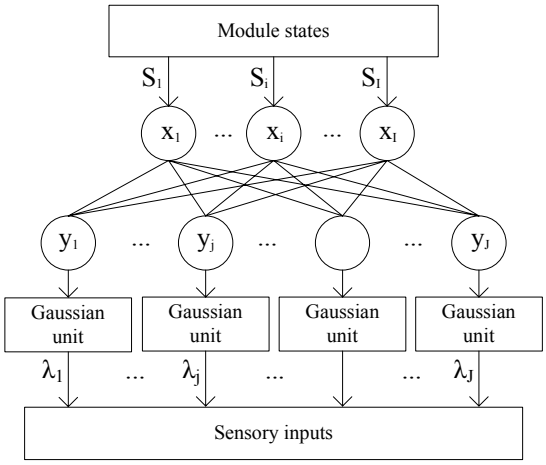

Fig. 4. A two-layer neural network for stochastic policy representation.

policy search algorithms rather than value function approaches. Thus we solve the mapping problem by using a policy gradient reinforcement learning method [20].

A two-layer neural network is constructed, as shown in Fig. 4. As a way of function approximation, a stochastic policy can be represented by the neural network.

Let x and y denote the units derived from the input layer and the output layer, respectively; let I and J be the number of units in the two layers; let w be the weight in the network; and let $net_j = \sum_{i=1}^{I} x_i w_{ij}$ be the linear representation of the units from the input layer.

For the output layer, the activation function for unit y_j is designed as a hyperbolic tangent type:

$$y_j = \frac{1 - e^{net_j}}{1 + e^{net_j}} \tag{5}$$

A Gaussian unit is appended to each output unit y_j in the neural network. Use of the stochastic output unit makes sense because their randomness allows any necessary exploration to take place. The Gaussian unit uses the output unit y as the mean and takes another variable σ as the standard deviation. Besides, a probability mass function is designed for the output unit y_j:

$$\begin{aligned} g_j &= Pr\{\lambda_j | y_j, \sigma\} \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-(\lambda_j - y_j)^2 / 2\sigma^2} \end{aligned} \tag{6}$$

The sensory output λ_j thus can be sampled from g_j:

$$\lambda_j = y_j + \sigma \cdot n \tag{7}$$

where $n \sim N(0,1)$. $N(0,1)$ is a Gaussian distribution which has a mean of 0 and a variance of 1.

Since the randomness of all the Gaussian units is independent and identically distributed, the overall probability mass function is dependent on the production of individual probability:

$$g = \prod_{j=1}^{J} g_j \tag{8}$$

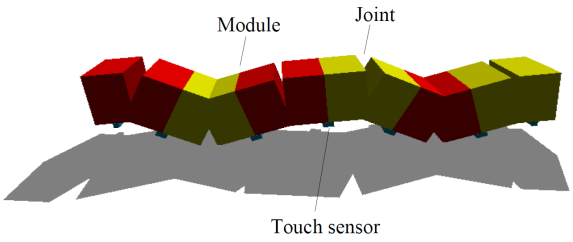

Fig. 5. Simulated caterpillar-like robot.

TABLE I

MODULE SPECIFICATION

Components	Module	Touch sensor
Length (mm)	72	10
Width (mm)	52	52
Height (mm)	52	10
Weight (g)	150	10

According to [20], The weight between the input unit x_i and the output unit y_j can be updated by:

$$\begin{aligned}\Delta w_{ij} &= \alpha(r-b)\frac{\partial ln(g)}{\partial w_{ij}} \\ &= \alpha'(r-b)(\lambda_j - y_j)(1-y_j^2)x_i \end{aligned} \tag{9}$$

where α' is the learning rate; r is the reward; b is the reinforcement baseline.

As a result of weight update, an optimized policy will be obtained.

IV. SIMULATION

A simulation has been carried out to verify the feasibility of the hierarchical control architecture. For simplicity, a simplified caterpillar-like robotic configuration was designed in Open Dynamics Engine (ODE) [21]. In the simulation, to comply with the rules in the real world, we set the gravity to $-9.81m/s^2$ and the friction coefficient to 0.6. The robot has 6 modules with 7 touch sensors mounted at its bottom. Each module is a simple rigid box, as shown in Fig. 5. Between each modules, a joint is utilized to connect and enable the module to rotate in a vertical plane with a range of $\pm 90^\circ$. Table I lists the specification of the robot's module.

TABLE II

CPG CONTROL PARAMETERS IN SIMULATION

Parameters	Value	Description
A	20	Amplitude
T	20	Time steps in one period
θ	$2\pi/3$	Phase difference
τ	5	Afferent speed of sensory input

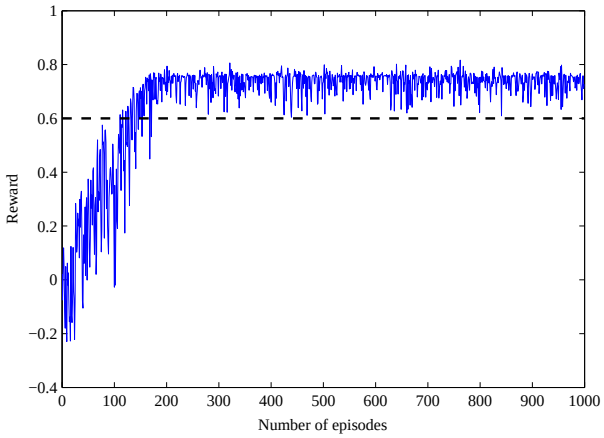

Fig. 6. The reward of the episodic learning. The dash line indicates the least reward for crossing the terrain.

A linear gait with an amplitude of 20° and a phase difference of 120° is applied on the robot as the basic locomotive pattern. Table II shows the control parameters of the CPG model that are used in the simulation. A simulation scenario is also constructed in ODE, consisting of two complete sinusoidal surfaces, as shown in Fig. 7. It has a length of $3m$, a width of $0.8m$, and the maximum height of $0.5m$.

The goal of this experiment is to obtain an optimized policy so that the robot can climb over the uneven terrain autonomously and adaptively. In the learning task, a two-layer neural network with 7 input units and 6 output units is generated for policy search. The weights in the neural network are initialized to random values and the standard deviation σ for each Gaussian unit is unified to 0.1. The learning rate and the reinforcement baseline are set to 0.1 and 0, respectively. A reward is defined as weighted average between normalized forward speed and normalized touch ratio:

$$Reward = \eta \cdot \frac{v}{v_0} + (1-\eta) \cdot \sum_{i=1}^{m} \sum_{t=1}^{n} \frac{S_{(i,t)}}{mn} \tag{10}$$

where η is the weight ($\eta = 0.8$ in the simulation for faster climbing speed but inferior touch ratio); v and v_0 are the forward speed for the robot to climb over uneven terrain and flat terrain, respectively; $S_{(i,t)}$ represents the ith module state at the tth time step; m is total module number of the robot; and n is the total time step.

The policy is trained in an episodic manner. First, for each episode, the simulation runs under the current policy with a fixed amount of time steps. A reward is obtained at the end of each episode. Then, the current policy is updated along the gradient direction with respect to the expected reward. This process is repeated until the predefined number of episodes is reached. Last, the Gaussian units in Fig. 4 are removed after training. In this experiment, the number of episodes to be done is set to 1000. Fig. 6 shows the variation of the reward with respect to the number of episodes. The result reveals that the robot managed to climb over the terrain after about 110 episodes and the performance of adaptive locomotion was gradually promoted to steady state with a reward of 0.75 after 200 episodes.

Fig. 7 illustrates how the robot climbs over the sinusoidal terrain using the final obtained policy. When the robot climbs up slopes, the policy generates positive offset on the internal modules of the robot, so that most modules of the robot can contact with the terrain to propel the robot forward. Whereas during the downhill movement, the policy produces positive offset on both ends of the robot. The robot thus bends up its head and tail to speed up the downhill motion. Note that after the downhill motion, the

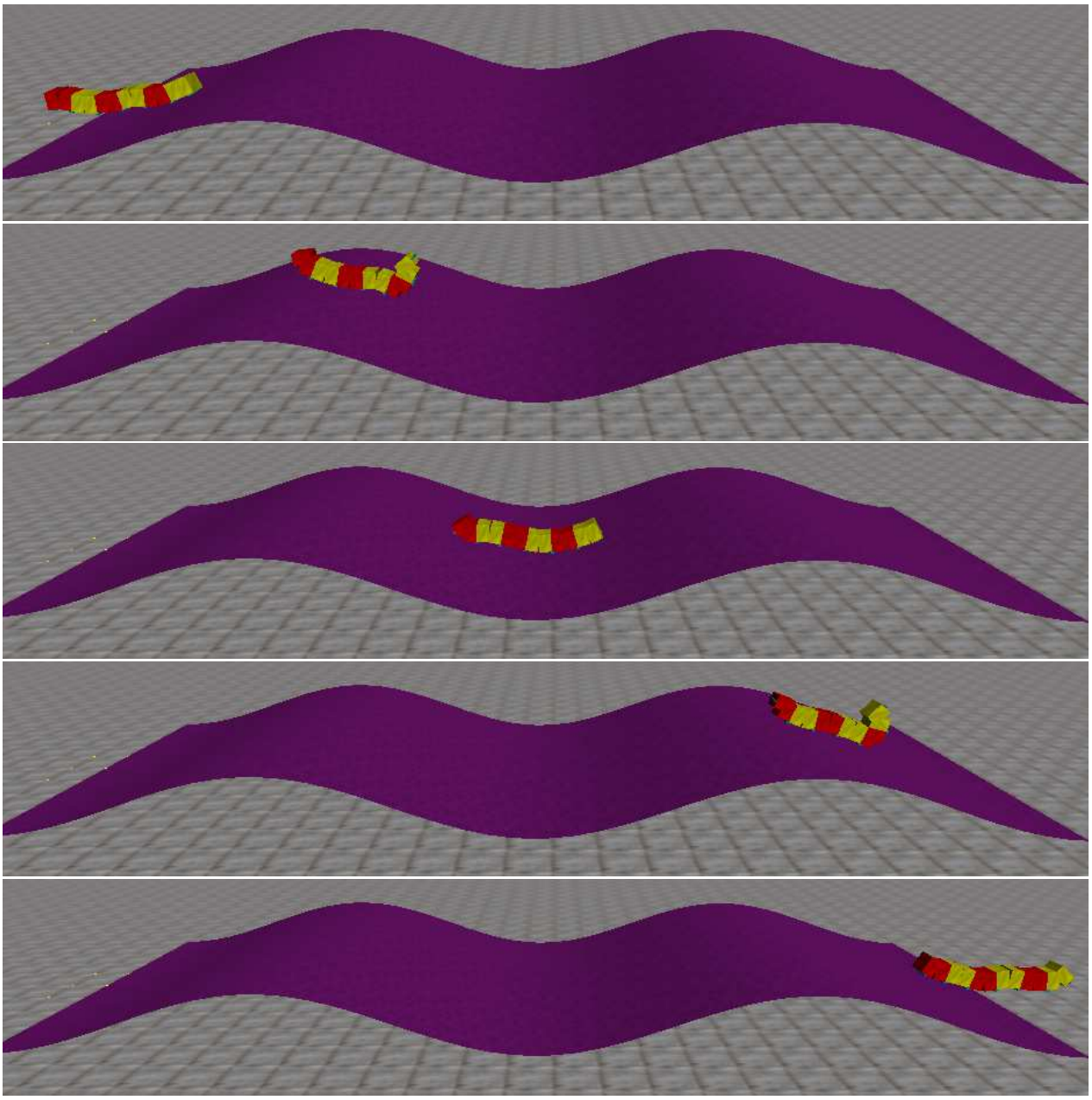

Fig. 7. Scenes of adaptive caterpillar-like locomotion. Adaptive locomotion starts from left to right.

policy removes the effects of offset on the modules at both ends. Therefore, the robot gradually recovers to the normal linear motion.

The whole climbing process lasts about 3200 time steps. For simplicity, here we only track the middle module of the robot (module 4). Fig. 8 shows the the variation of the sensor input and the module output for module 4 during the simulation. From Fig. 8(a), the sensory input is modified dramatically to the positive extreme around the $100th$, $1500th$ and $2900th$ time steps, when the robot is climbing at the left end, the middle and the right end of the terrain, respectively. Fig. 8(b) shows the module bends upside correspondingly at these time steps. It is worth noting that recovery of offset bending always

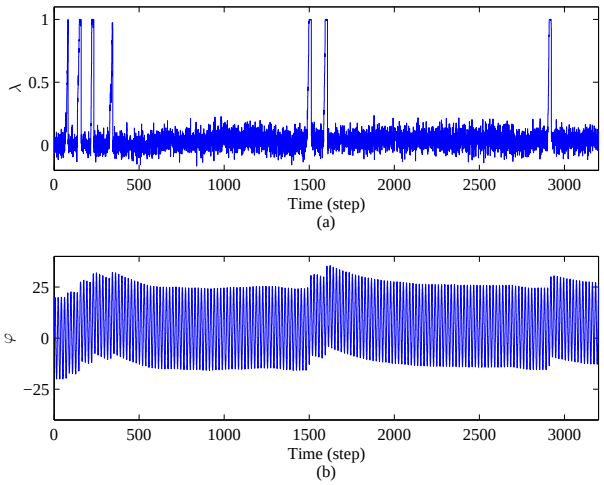

Fig. 8. Tracked data for module 4 in the simulation. (a) The sensory input. (b) The module output.

happens after these extreme spikes of sensory input. The module can bend back to normal state in a short period. The angle variation for module 4 is smooth and reasonable, which helps the robot to adapt to the environmental changes.

From the simulation result, we conclude that the effectiveness of our proposed hierarchical control system.

V. Conclusion

In this paper, how to combine sensory feedback with a CPG model in a hierarchical control architecture to realize adaptive caterpillar-like locomotion is presented. First, as a simple CPG model, sinusoidal generators are utilized as the locomotion controller of the robot. Taking advantages of online modulation of CPG models, sensory feedback is designed to regulate the offset of the CPG output. Then, sensor data filtering is used for simplifying sensory information from the environment. Next, a mapping between the sensory information and the sensory input to the CPG model is studied by means of policy gradient reinforcement learning. Through episodic learning, an optimized policy can be obtained for specific applications. Last, simulation results show the proposed approach is effective in realizing adaptive locomotion for caterpillar-like robots.

Future work will focus on two aspects: (1) Refine the mechanism for sensory integration, in particular for developing more powerful sensory feedback control for not only affecting the offset, but also the phase difference and the amplitude of the CPG model. (2) Combine multiple channels of information from sensors such as accelerometer and camera to improve the perception of environmental changes.

References

[1] S. Grillner, “Neurobiological bases of rhythmic motor acts in vertebrates”, *Science*, vol. 228, pp. 143-149, 1985.
[2] A. J. Ijspeert, “Central pattern generators for locomotion control in animals and robots: a review”, *Neural Networks*, vol. 21, no. 4, pp. 642-653, 2008.
[3] F. Herrero-Carrón, F. B. Rodríguez and P. Varona, “Bio-inspired design strategies for central pattern generator control in modular robotics”, *Bioinspiration & Biomimetics*, vol. 6, no. 1, pp. 1-16, 2011.
[4] Ö. Ekeberg, “A combined neuronal and mechanical model of fish swimming”, *Biol. Cybern.*, vol. 69, pp. 363-374, 1993.
[5] H. Kimura, Y. Fukuoka, A. H. Cohen, “Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts”, *Int. J. Robot. Res.*, vol 26, no. 5, pp. 475-490, 2007.

[6] X. Wu and S. Ma, “Adaptive creeping locomotion of a CPG-controlled snake-like robot to environment change”, *Auton. Robot*, vol. 28, pp. 283-294, 2010.
[7] A.J. Ijspeert and A. Crespi, “Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model”, in *proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, Roma, pp. 262-268, 2007.
[8] J. Gonzalez-Gomez, H. Zhang and E. Boemo, “Locomotion principles of 1D topology pitch and pitch-yaw-connecting modular robots”, in *Bioinspiration and Robotics: Walking and Climbing Robots*, Vienna, pp. 403-428, September 2007.
[9] M. Tesch, K. Lipkin, I. Brown, R.Hatton, A. Peck, J. Rembisz and H. Choset, “Parameterized and scripted gaits for modular snake robots”, *Adv. Robot.*, vol. 23, pp. 1131-1158, 2009.
[10] S. Hasanzadeh and A. A. Tootoonchi, “Adaptive optimal locomotion of snake robot based on CPG-network using fuzzy logic tuner”, in *proc. of IEEE Int. Conf. on Robotics, Automation and Mechatronics*, Chengdu, China, pp. 187-192, 2008.
[11] A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji and S. Murata, “Distributed adaptive locomotion by a modular robotic system, M-TRAN II”, in *proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan, pp. 2370-2337, 2004.
[12] T. Tomoyuki, Y. Azuma and T. Shibata, “Acquisition of energy-efficient bipedal walking using CPG-based reinforcement learning”, in *proc. of IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, USA, pp. 827-832, 2009.
[13] M. Snel, S. Whiteson and Y. Kuniyoshi, “Robust central pattern generators for embodied hierarchical reinforcement learning”, in *proc. of the First Joint IEEE International Conference on Development and Learning and Epigenetic Robotics*, pp. 1-6, August 2011.
[14] S. Aoi and K. Tsuchiya, “Stability analysis of a simple walking model driven by an oscillator with a phase reset using sensory feedback”, *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 391- 397, 2006.
[15] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi and G. Cheng, “Learning CPG-based biped locomotion with a policy gradient method: application to a humanoid robot”, *Int. J. Robot. Res.*, vol. 27, no. 2, pp. 213-228, 2008.
[16] C. Liu, Q. Chen and D. Wang, “CPG-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots”, *IEEE Trans. Systems, Man and Cybernetics*, Part B, vol. 41, no. 3, pp. 867-880, 2011.
[17] L. Wang, S. Wang, Z. Cao, M. Tan, C. Zhou, H. Sang and Z. Shen, “Motion control of a robot fish based on CPG”, in *proc. of IEEE Int. Conf. on Industrial Technology (ICIT)*, HongKong, China, pp. 1263-1268, 2005.
[18] A. El-Fakdi and M. Carreras, “Policy gradient based reinforcement learning for real autonomous underwater cable tracking”, in *proc. of IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, pp. 3635-3640, 2008.
[19] H. Zhang, J. Zhang and W. Wang, “From the biological model to a small climbing caterpillar robot”, *Int. J. Advanced Mechatronic Systems*, vol. 2, no. 1/2, pp. 90-98, 2010.
[20] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning”, *Machine Learning*, vol. 8, no. 3, pp. 229-256, 1992.
[21] Open dynamics engine, http://www.ode.org/.