ELSEVIER

Artificial Intelligence in Engineering 15 (2001) 353–363

ARTIFICIAL INTELLIGENCE in ENGINEERING

www.elsevier.com/locate/aieng

Emergent synthesis of motion patterns for locomotion robots

M.M. Svinin[a,*], K. Yamada[b], K. Ueda[b]

[a]*Bio-Mimetic Control Research Center, RIKEN, Anagahora, Shimoshidami, Moriyama-ku, 463-0003 Nagoya, Japan*
[b]*Department of Mechanical Engineering, Kobe University, 1-1, Rokkodai, Nada-ku, Kobe 657-8501, Japan*

Received 7 May 2001; accepted 17 September 2001

Abstract

Emergence of stable gaits in locomotion robots is studied in this paper. A classifier system, implementing an instance-based reinforcement-learning scheme, is used for the sensory-motor control of an eight-legged mobile robot and for the synthesis of the robot gaits. The robot does not have a priori knowledge of the environment and its own internal model. It is only assumed that the robot can acquire stable gaits by learning how to reach a goal area. During the learning process the control system is self-organized by reinforcement signals. Reaching the goal area defines a global reward. Forward motion gets a local reward, while stepping back and falling down get a local punishment. As learning progresses, the number of the action rules in the classifier systems is stabilized to a certain level, corresponding to the acquired gait patterns. Feasibility of the proposed self-organized system is tested under simulation and experiment. A minimal simulation model that does not require sophisticated computational schemes is constructed and used in simulations. The simulation data, evolved on the minimal model of the robot, is downloaded to the control system of the real robot. Overall, of 10 simulation data seven are successful in running the real robot.

Keywords: Locomotion robots; Minimal simulation; Gait patterns; Emergence; Reinforcement learning; Classifier systems

1. Introduction

The problem of understanding of sensible motions in living organisms and reproducing them in mechanical systems is challenging researchers and engineers working in the field of robotics. The problem is very complex, especially for locomotion where legs interact with the external environment and the structure of the system is varied from step to step. Classical, model-based control approaches gives us a basic framework for studying mechanics of locomotion [1]. They can be used to synthesize a very limited subset of possible gaits. At the same time, they are too sensitive and may fail if autonomy of the robot is required to function in unknown environments. Thus, new paradigms for synthesis of sensorimotor coordination in complex locomotion system should be developed and evaluated [2].

Biological approaches are an alternative for studying locomotion [3] as the living creatures offer proofs-of-concept. In living organisms, the locomotion gaits are the product of interaction between nervous and muscular systems. While the muscular system can be well formalized within theoretical mechanics, modeling of the nervous system is still an open field of research. Interesting biological considerations on the design of controllers for locomotion robots are given by Cruse [4,5] who developed a biologically inspired network for controlling six-legged walking.

One of the most popular approaches to self-organized control of locomotion robots and evolving periodical gaits is based on the use of a network of neural oscillators. This approach, associated with the central pattern generation, has received considerable attention in literature. The idea is that the neural oscillators can gradually excite a rhythmic motion of the robot [6–8]. It has been applied to control of bipeds [9–11], quadruped [12,13], and hexapod [14–17] robots. As the number of legs is increasing, the structure of the neural network becomes more complex. To cope with the difficulties in synthesis of the leg controllers, the authors of the last four cited papers used genetic algorithms to develop dynamical neural networks.

In living organisms, stable periodical gaits are essentially an emergent phenomenon. One possible approach to evolving stable periodical gaits is to exploit the idea of self-organization by learning the perceived information. From this point of view the ideas of reinforcement learning [18] and learning classifier systems are especially attractive since they can be applied to a wide variety of complex adaptive systems. Learning classifier systems are a class of inductive models, capable of adaptation and learning in reinforcement

* Corresponding author.
E-mail address: svinin@bmc.riken.go.jp (M.M. Svinin).

0954-1810/01/$ - see front matter
PII: S0954-1810(01)00027-9

mode [19–22]. In a sense, they emulate the mental processes in animals and humans.

The concept of learning classifier systems defines a very inspiring paradigm of evolutionary machine learning. Dorigo and Colombetti used classifier systems to control wheeled mobile robots [23]. However, similar principles can be used in sensorimotor control and coordination of leg movements. In our earlier work we designed a classifier system implementing the instance-based reinforcement learning [24], and applied it to behavior acquisition of autonomous mobile robots [25]. This classifier system can also be used for studying emergence of motion patterns in locomotion mechanisms. In this paper we use it to synthesize the robot gaits.

Evolving the robot controllers by learning requires a considerable amount of repetitive trials. The learning procedure can be tested on real robots but it may take a long time on the traditional hardware. An alternative way is to implement the learning procedure on a simulator and, after successful learning, download it into the robot hardware. In this case the output of the simulation can be compared with the genetic information a child receives from his/her parents as shown in Fig. 1.

It should be pointed out that evolving the robot controllers under simulation is one of the central problems facing the development of evolutionary robotics. Unless the problem is solved, the application of evolutionary robotics to evolving behavior of such complex systems as walking machines or multi-fingered robot hands is at least unrealistic. Therefore, the development of carefully constructed and validated empirical models, computationally suitable for inclusion into the evolution loop, is of crucial importance.

The development of such empirical models, closely linked to mechanics, artificial intelligence and computational psychology, requires establishing some basic principles. One of them, the minimal simulation principle, was proposed by Jakobi, who applied it to different robotic systems [26]. The need of minimal simulation models was also pointed out by Pin, who studied a fuzzy behaviorist approach for controlling autonomous mobile robots [27].

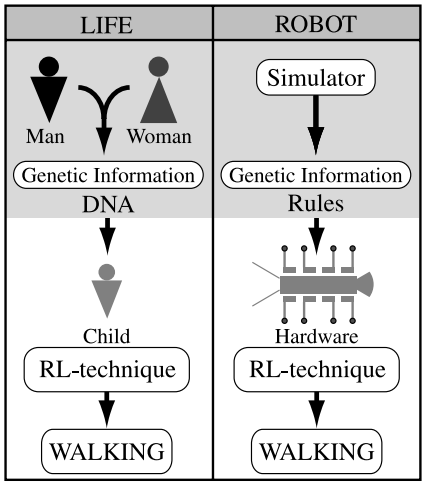

Fig. 1. Role of simulation in evolving controllers.

In this paper we, first, develop a minimal simulation model, and then use it in the synthesis of stable gait patterns for an eight-legged walking machine. The synthesis of motion patterns is done by a learning classifier system that can generate a compact set of rules corresponding to a specific locomotion problem. In general, the rules are hard to predict in advance especially in case if the number of legs is large. In our approach, under proper reinforcement, the rules and the corresponding motion patterns emerge during the learning process. In this sense the synthesis of feasible robot gaits can be called emergent.

This paper is organized as follows. The classifier system is briefly described in Section 2. Organization of the control scheme is presented in Section 3. A minimal simulation model is devised in Section 4. Simulation results and experimental verification of the controller evolved on the minimal simulation model are discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. Classifier system

In our approach, the robot control system is modeled by a classifier system that outputs control command in response to a sensory input. In the classifier system, the actually observable sensor space is populated with action rules mapping certain state space to certain actions. The sensor state space evolves and, as learning progresses, its structure is self-organized (Fig. 2). The structure of the classifier systems is similar to the basic one proposed by Wilson [28]. The essential difference is that we formulate it in the continuous state and action spaces.

2.1. Action rule

Let n_s be the number of sensors and $X = \{x_1, \ldots, x_{n_s}\}^T$ be the sensory input of the robot. The system operates on a set

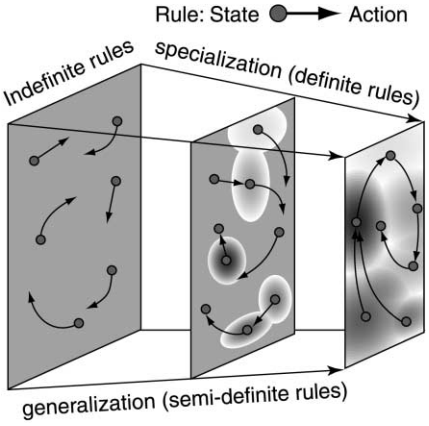

Fig. 2. Dynamics of the state space.

of action rules, R. The rule $r \in R$ is defined as follows:

$$r := \langle V, W, u, a \rangle, \tag{1}$$

where $V = \{v_1, \ldots, v_{n_s}\}^T$ is the state vector associated with and memorized in the rule r, $W = \{w_1, \ldots, w_{n_s}\}^T$ is the weight vector, u is the utility of the rule, a is the action corresponding to the rule r.

The utility u is a real number indicating the relative value of the rule in the learning process. It does not have any direct physical meaning and may be only associated with the rule's strength or internal energy. The utility may have a biological meaning since it is inherited from the rule's parent and is changing later during the evolution process.

If V matches in some sense the current sensory input X, the rule r becomes active and can trigger its action a. The weight vector W is used for comparing V and X. $w_i \in [0, 1]$ is the continuous analog of the Holland's 'don't care' symbol #. The closer w_i to zero, the less important the measurement of the ith sensor. The rules for which $W = 0$ are called indefinite. They can be activated anywhere regardless of the current state X the robot is in. All the other rules are definite. They can be activated in a vicinity of V, and the vicinity is defined using the weight W. The rule's specificity:

$$\lambda = \frac{1}{n_s} \sum_{i=1}^{n_s} w_i \tag{2}$$

serves as a measure of definiteness of the rule. When λ is closer to 1, the behavior specified by the rule is more reactive. On the other hand, as λ gets closer to 0 the behavior becomes more proactive (that is more 'unrestricted' in the way of exploration of the environment).

Actually, in our implementation it is enough to keep only one indefinite rule. All the other rules in R are definite. In the beginning R consists of the indefinite rule with initially assigned utility u_0. As learning progresses, the total number of rules in R, n_r, varies by reproduction and extinction.

2.2. Action selection

The rules in R compete with each other for the right to trigger their actions. For all the rules $r_j \in R$, the normalized weighted distance between the current sensory state X and the rule's state vector V^j is defined as:

$$\sigma_j^2 = \frac{1}{n_s} \sum_{k=1}^{n_s} \left(\frac{x_k - v_k^j}{d_k} w_k^j \right)^2, \tag{3}$$

where d_k is the time-dependent scaling parameter. It is defined as the maximum difference between the maximum and the minimum values of the k-th sensor observed during the learning process.

Next, we define the matching rate:

$$m_j = \exp(-T_m \sigma_j^2), \tag{4}$$

where T_m is a constant. Note that even if no definite rules match the sensory input X, the indefinite rule will always do. Indeed, $W = 0$ and the matching rate $m = 1$ regardless of the encountered state X. It makes the indefinite rule a likely candidate for selection. This is especially important in the beginning of the learning process when the indefinite rule is often taken for execution and generation of new rules.

It is also to be noted that whenever the indefinite rule is selected for execution, the action associated with this rule is generated randomly in accordance with the uniform distribution within $[a_{\min}, a_{\max}]$.

The winner rule is selected with probability given by the weighted Boltzmann distribution:

$$P(r_j) = \frac{m_j \lambda_j \exp(u_j/T)}{\sum_{k=1}^{n_r} m_k \lambda_k \exp(u_k/T)}, \tag{5}$$

where the parameter T has the meaning of temperature, setting the balance between exploitation and exploration of the state space.

2.3. Credit assignment

The utilities of the rules are updated every time after the winner executes its action. The utility adjustment mechanism consists of following parts.

Direct payoff distribution. The direct payoff P is given to the winner rules only in specific states. There are two types of payoff: reward ($P > 0$) and punishment ($P < 0$). The payoff is spreading back along the sequence of the rules triggered their actions (i.e. to the current and previous winners) with the discount rate γ:

$$u_w^{(k)} \leftarrow u_w^{(k)} + \gamma^k P, \qquad k = 0 \ldots N, \tag{6}$$

where N is the depth of the winners chain, and $0 < \gamma < 1$. This corresponds to the profit sharing strategy where the payoff is gradually discounted as the time step passes backward. Here, the parent of r_w is $r_w^{(1)}$, the parent of $r_w^{(1)}$ is $r_w^{(2)}$ and so on.

Bucket brigade strategy. The current winner r_w hands over a part of its utility, Δu, back to the previous winner, $r_w^{(1)}$:

$$u_w^{(1)} \leftarrow u_w^{(1)} + \Delta u, \tag{7}$$

where

$$\Delta u = \begin{cases} \kappa \lambda_w (u_w - u_w^{(1)}), & \text{if } u_w > u_w^{(1)} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

and $\kappa \in [0, 1]$. Note that the rule $r_w^{(1)}$ increases its utility. However, we do not decrease the utility of the rule r_w and this is the main distinction between our strategy and the conventional ones.

If actions are triggered only by a limited number of rules (subset of R) and they hand over Δu to one another, the utility of each rule is expected to gradually converge to the highest utility among these rules. Therefore, action rules, cooperating in such a manner, can survive a period

during which rewards are rare. This can be associated with self-organization of the rules looking for an eventual reward.

Taxation. Whenever a definite rule r_w triggers its action, its utility is updated as:

$$u_w \leftarrow (1 - c_f)u_w. \tag{9}$$

The rule r_w pays a cost at the activation rate c_f to prevent dead lock or loop behavior. In a sense, the rule is taxed for the right to perform. Note that the indefinite rule is tax-free since its main function is to generate new rules.

Evaporation. All rules reduce their utilities at the evaporation rate $\eta < 1$ when the robot reaches the goal state:

$$u_w \leftarrow \eta u_w. \tag{10}$$

In a sense, it corresponds to the 'currency inflation'. The rules decreasing their utility below the threshold u^{min} are removed.

2.4. Reproduction

Having selected the winner rule r_w, we execute the action a_w associated with this rule. Next thing, we should take care of after the action execution and utility adjustment is the reproduction process. In our system, the winner rule r_w always generates a new rule r_c (child rule) except for the case when the action triggered by r_w has led to stepping back or falling down. Details of the reproduction process are formalized as follows.

If the winner is the indefinite rule, the reproduced rule parameters are set as:

$$v_i^c = x_i, \qquad w_i^c = 1,\ i = 1,\ldots,n_s. \tag{11}$$

We call it 'memorization of experience'. The utility of the new rule and its action code are passed from the parent:

$$a_c = a_w, \qquad u_c = u_w. \tag{12}$$

On the other hand, if the winner is a definite rule we try to 'generalize the experience'. In this case, the newly produced rules are called generalized.

Note that the rule r_w with high utility u_w can win the competition even if its matching rate $m_w < 1$. In our system, the winner reproduces a generalized rule r_c provided that its matching rate m_w is within a certain reproduction threshold θ_r, i.e. $m_w < \theta_r$. It is reasonable to relate θ_r with the rule's utility u_w by the following expression:

$$\theta_r = \exp(-T_r u_w) \tag{13}$$

where T_r is a constant. The implication is that for the rules with higher matching rate we allow lesser utility to pass through the reproduction barrier, and vice verse.

The vectors V^c and W^c for the new generalized rule are set as follows:

$$v_i^c = x_i, \qquad w_i^c = 1 - \frac{|x_i - v_i^c|}{d_i},\ \ i = 1,\ldots,n_s. \tag{14}$$

The utility and the action code for the new generalized rule are set as:

$$a_c = a_w, \qquad u_c = \lambda_c u_w. \tag{15}$$

Note that the new generalized rule built as above can match a wider state space, which includes the state of its parent.

3. Walking machine

An eight-leg locomotion robot, OCT1-b, produced by AAI, is used in the experiments. The robot is shown in Fig. 3. Every leg has two degree of mobility (β for the rotation in the body plane, and α for lifting up and down as shown in Fig. 4), and is equipped with two angle sensors. In addition, there are two over-current sensors for each leg; 15 light sensors, eight infra-red sensors, and 11 whisker switches. The robot is controlled by and monitored from a PC (Sharp Mebius, Pentium MMX-166) through a serial communication cable.

It is assumed that the robot can acquire stable periodical gaits by learning how to reach a light source. The learning trials are called episodes. The episodes are updated when the robot reaches the light source, or when the number of produced actions reaches some limit, N_{amax}. In general, the learning task can be decomposed into consequent acquisition of stable walking and acquisition of the straight forward motions. However, in our teaching strategy we combine these sub-tasks into one, keeping the separate learning for the future research.

One classifier system is used for controlling the robot. The robot does not have a priori knowledge of the environment, its own internal model, and the goal coordinates. Based on the current sensory input, the control system outputs the commanded action for each leg. There are four actions for every leg. The actions for the first four

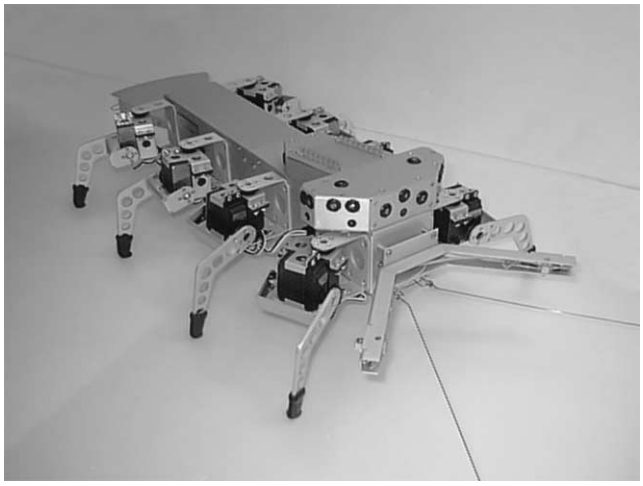

Fig. 3. OCT1-b robot.

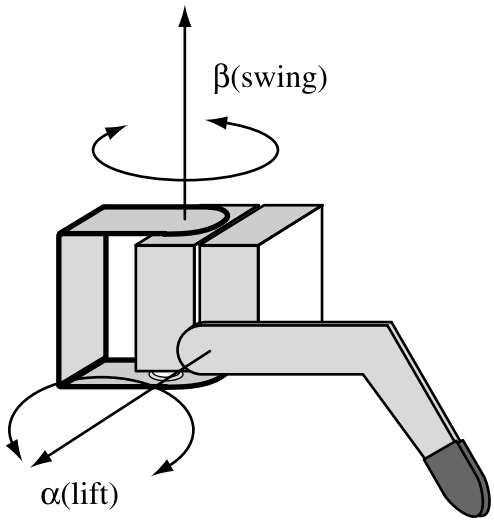

Fig. 4. Degrees of freedom.

front and rear legs are listed in Table 1. The actions and the values of the commanded angles are the factory settings for the sample motions of the robot.

The robot configuration is defined by the action that has been commanded. The total dimension of the action space ($4^8 = 65,536$) is rather large. In a very rough approximation half of the action space (125×2^8) is formed by the unstable configurations. While it is possible to enumerate the stable configurations and organize the learning process in the space of the 'stable' actions, this approach would not be general enough as in reality the explicit description of the stable configurations is not always possible. Therefore, we organized the learning process in the non-decomposed configuration space.

During the learning process the control system is self-organized by reinforcement signals. Reaching the light source defines a global reward. Forward motion of the robot gets a local reward, while stepping back gets a local punishment. The forward and backward motions can be detected from the intensity of the light source measured by the light sensors at the front panel of the robot.

The actions generated by the control system can lead to stable or unstable configurations of the robot as shown in Fig. 5. Here, the stability is understood in the static sense, i.e. it is preserved if the gravity center of the body, projected onto the ground, lies within the contact leg polygon. In the

Table 1
List of actions for the four front legs and the four rear legs

Number	Description	Front legs		Rear legs	
		β (°)	α (°)	β (°)	α (°)
1	Move forward and lift down	+39	−10	+7	−10
2	Move back and lift down	−7	−10	−39	−10
3	Move back and lift up	−7	+35	−39	+35
4	Move forward and lift up	+35		+7	+35

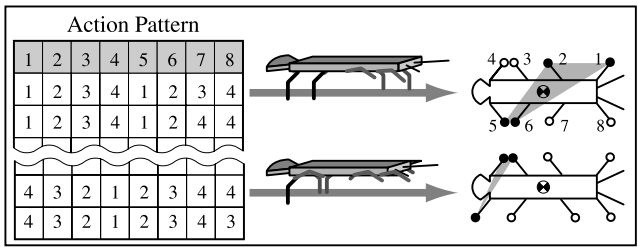

Fig. 5. Actions and configurations.

credit assignment scheme, whenever the robot comes into an unstable configuration, the rule that led to this configuration gets a local punishment.

4. Minimal simulation model

The simulation based on the exact dynamic model of a locomotion robot, including unilateral constraints imposed on the legs in the contact mode and taking into account the variable structure of the system, is still computationally expensive. As a result, the learning process built on the full dynamic model can be prohibitively slow, and all speed advantages over the 'real world' evolution can be lost [26].

In literature on locomotion systems, Jakobi constructed a very simple minimal model of an eight-legged robot (similar to OCT1-b), which he explained only diagrammatically [26]. Cymbalyuk developed a simple kinematic model of a six-legged robot [29]. In our previous study [30] we tested a minimal model of a four-legged robot. The model was based on psychological considerations and was associated with a virtual jump. In this section, we construct a relatively simple minimal model for the OCT1-b robot. The model takes into account, generalized parameters (friction and compliance) of the contact between the robot feet and the ground.

For the robot under consideration the minimal simulation model is a means to define the robot center position $x(t+1)$, $y(t+1)$ and orientation $\theta(t+1)$ based on the current configuration and the current control command. Let n_{ij}^{r} be the number of legs on the right side changing their configuration from the state i at the moment t to the state j at the moment $t+1$. Similarly define n_{ij}^{l} for the left side. It is assumed that the current control command leads to a stable configuration.

In defining the next configuration of the robot the legs that do not contact with the ground are ignored. Moreover, we will also ignore the legs that, at the instances t and $t+1$, are in transition to or from contact. Thus, the legs coming from state i to j do not contribute to motion of the body if i or j is 3 or 4. Physically, it corresponds to ignoring inertia of the legs.

Let $p > 0$ be the elementary propulsion force when it moves while keeping contact with the ground, and $q > 0$ be the elementary resisting force of the leg when it is fixed.

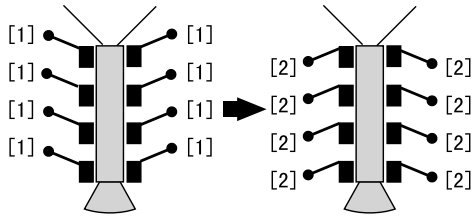

Fig. 6. Pure translational motion.

Define the driving force for the right side as:

$$f^{\mathrm{r}}_{\mathrm{drv}} = p\left(n^{\mathrm{r}}_{12} - n^{\mathrm{r}}_{21}\right), \tag{16}$$

and the resisting force as:

$$f^{\mathrm{r}}_{\mathrm{res}} = q\left(n^{\mathrm{r}}_{11} + n^{\mathrm{r}}_{22}\right). \tag{17}$$

In defining the total driving force F^{r} for the right side we employ the reasoning similar to that of the model of Coulomb friction. Namely, we assume that there is no motion and $F^{\mathrm{r}} = 0$ if $|f^{\mathrm{r}}_{\mathrm{drv}}| \leq f^{\mathrm{r}}_{\mathrm{res}}$. Otherwise, we define:

$$F^{\mathrm{r}} = \begin{cases} f^{\mathrm{r}}_{\mathrm{drv}} - f^{\mathrm{r}}_{\mathrm{res}} & \text{if } f^{\mathrm{r}}_{\mathrm{drv}} > f^{\mathrm{r}}_{\mathrm{res}}, \\ f^{\mathrm{r}}_{\mathrm{drv}} + f^{\mathrm{r}}_{\mathrm{res}} & \text{if } f^{\mathrm{r}}_{\mathrm{drv}} < -f^{\mathrm{r}}_{\mathrm{res}}. \end{cases} \tag{18}$$

In a similar way, changing the superscript r to l in Eqs. (16)–(18), we define F_{l}, the propulsion force for the left side of the robot. Then, we are in a position to define the total propulsion force F and moment M:

$$F = F^{\mathrm{l}} + F^{\mathrm{r}}, \tag{19}$$

$$M = F^{\mathrm{l}} - F^{\mathrm{r}}. \tag{20}$$

Having constructed F and M, we can define the linear displacement of the robot along the body axes:

$$\Delta v = c_v F, \tag{21}$$

and the rotational displacement of the body:

$$\Delta\theta = c_\theta M. \tag{22}$$

Here in the equations, c_v and c_θ stand, respectively, for the linear and rotational compliance.

Finally, the position and orientation of the robot at the moment $t + 1$ are defined:

$$x(t+1) = x(t) + \Delta v \cos\theta(t), \tag{23}$$

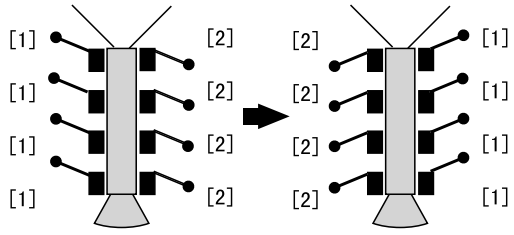

Fig. 7. Pure rotational motion.

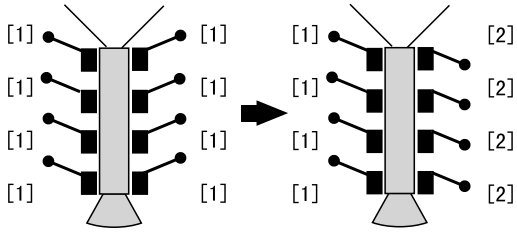

Fig. 8. Translational and rotational motions.

$$y(t+1) = y(t) + \Delta v \sin\theta(t), \tag{24}$$

$$\theta(t+1) = \theta(t) + \Delta\theta. \tag{25}$$

Of course, the minimal simulation model built as above is very rough. In this model, the robot motion is similar to that of a rowing boat. In a sense, it is considered as an intuitive model of prediction of the next possible configuration of the robot. The 'virtual boat' model resembles a simple non-holonomic system (unicycle mobile robot) as it excludes the lateral displacements of the body. Also excluded are any dynamical effects related to the balancing motions of the body.

Example 1. Consider the example shown in Fig. 6, where the current configuration of the legs is on the left side and the configuration required by the control command is on the right side. Here, we have $n^{\mathrm{r}}_{12} = 4$, $n^{\mathrm{r}}_{21} = 0$, $n^{\mathrm{r}}_{11} = 0$, $n^{\mathrm{r}}_{22} = 0$, $n^{\mathrm{l}}_{12} = 4$, $n^{\mathrm{l}}_{21} = 0$, $n^{\mathrm{l}}_{11} = 0$, $n^{\mathrm{l}}_{22} = 0$. Calculations by Eqs. (16)–(18) give $F = 8p$ and $M = 0$, which leads to the pure translational forward motion of the robot body.

Example 2. Consider the example shown in Fig. 7. Here, we have $n^{\mathrm{r}}_{12} = 0$, $n^{\mathrm{r}}_{21} = 4$, $n^{\mathrm{r}}_{11} = 0$, $n^{\mathrm{r}}_{22} = 0$, $n^{\mathrm{l}}_{12} = 4$, $n^{\mathrm{l}}_{21} = 0$, $n^{\mathrm{l}}_{11} = 0$, $n^{\mathrm{l}}_{22} = 0$. Calculations by Eqs. (16)–(18) give $F = 0$ and $M = 8p$, which leads to the pure rotational motion of the robot body to the left side.

Example 3. Consider the example shown in Fig. 8. Here, we have $n^{\mathrm{r}}_{12} = 4$, $n^{\mathrm{r}}_{21} = 0$, $n^{\mathrm{r}}_{11} = 0$, $n^{\mathrm{r}}_{22} = 0$, $n^{\mathrm{l}}_{12} = 0$, $n^{\mathrm{l}}_{21} = 0$, $n^{\mathrm{l}}_{11} = 4$, $n^{\mathrm{l}}_{22} = 0$. Calculations by Eqs. (16)–(18) give $F = 4p$ and $M = -4p$, which leads to the translational forward motion of the body accompanied with the turn to the right side.

Example 4. Consider the example shown in Fig. 9. Here,

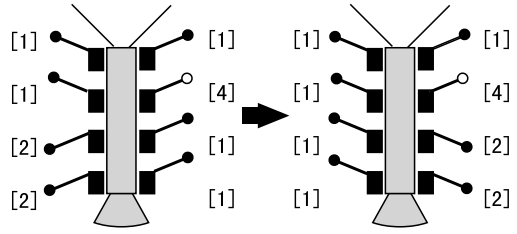

Fig. 9. Translational and rotational motions.

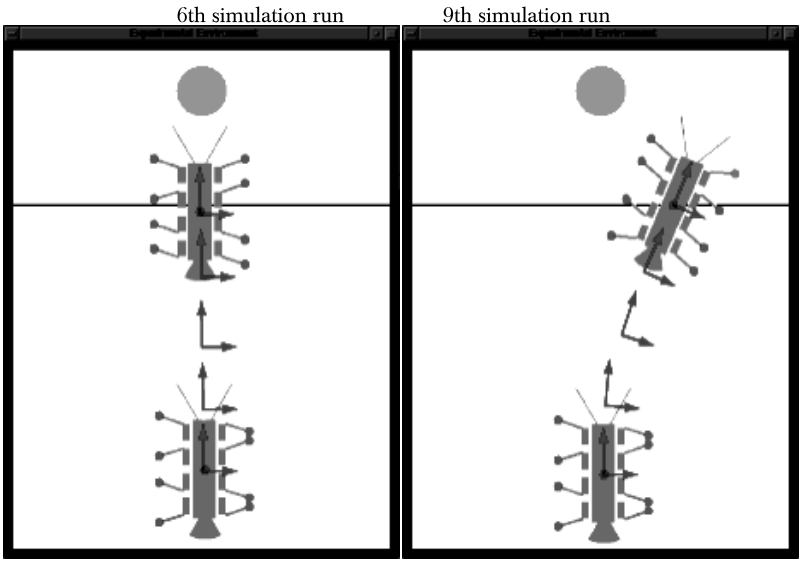

Fig. 10. Trajectory of the robot at the 90th episode.

we have $n^{\mathrm{r}}_{12} = 2$, $n^{\mathrm{r}}_{21} = 0$, $n^{\mathrm{r}}_{11} = 1$, $n^{\mathrm{r}}_{22} = 0$, $n^{\mathrm{l}}_{12} = 0$, $n^{\mathrm{l}}_{21} = 2$, $n^{\mathrm{l}}_{11} = 2$, $n^{\mathrm{l}}_{22} = 0$. Calculations by Eqs. (16)–(18) show that $F = 0$ and $M = 0$ (no motion) if $p \le q/2$, $F = 2p - q$ and $M = q - 2p$ (forward translation and rotation to the right) if $q/2 < p \le q$, $F = q$ and $M = 3q - 4p$ (forward translation and rotation to the right) if $p > q$.

The simplicity of the minimal simulation model is attractive since it allows to conduct the learning process with a relatively low computational cost. However, the model parameters p, q, c_v and c_θ, depending on the friction and the compliance of the contact between the leg foot and the ground, must be established experimentally in advance. In our experiments (rubber foot on the linoleum) we roughly defined $p = 2.0$, $q = 1.0$, $c_v = 0.5$, $c_\theta = 0.87$.

5. Simulation and experimental results

First, feasibility of the learning scheme is tested under simulation. The episodes are updated when the robot reaches the goal area source, or when the number of produced actions reaches 500. The parameters are set as follows: $P = 5$ for the global reward, $P = 5$ for the local reward, $P = -5\%$ of the corresponding utility for the punishment, $u_0 = 10$, $u_{\min} = 9.5$, $c_f = 0.015$, $\gamma = 0.8$, $\kappa = 0.1$, $\eta = 0.98$, $T = 3$, $T_m = 100$, and $T_{\mathrm{r}} = 0.5$.

Ten simulation runs are conducted in row, and for each of them the required behavior is evolved. The simulation runs differ only in the initially generated random numbers. The robot trajectories for the sixth and ninth simulation runs are shown in Fig. 10. In both cases the robot reaches the goal area. Note that even in the successful episodes the robot does not go to the goal by the shortest path. Actually, the information about the heading direction of the robot does not enter the sensor space of the learning scheme. Therefore, no credit assignment (reward or punishment) is performed for the heading direction. Note, that even though the robot does not always go to the goal straightforwardly, it still can acquire stable walking patterns in the leg action space.

Dynamics of the learning process are presented in Fig. 11, plotting the number of punishments, sub-rewards, and the total number of steps necessary to reach the goal area by,

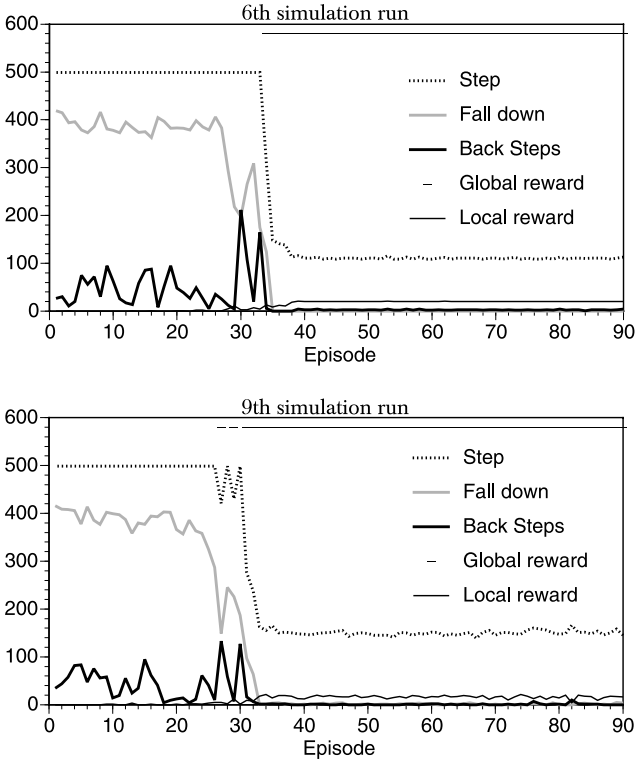

Fig. 11. Learning history.

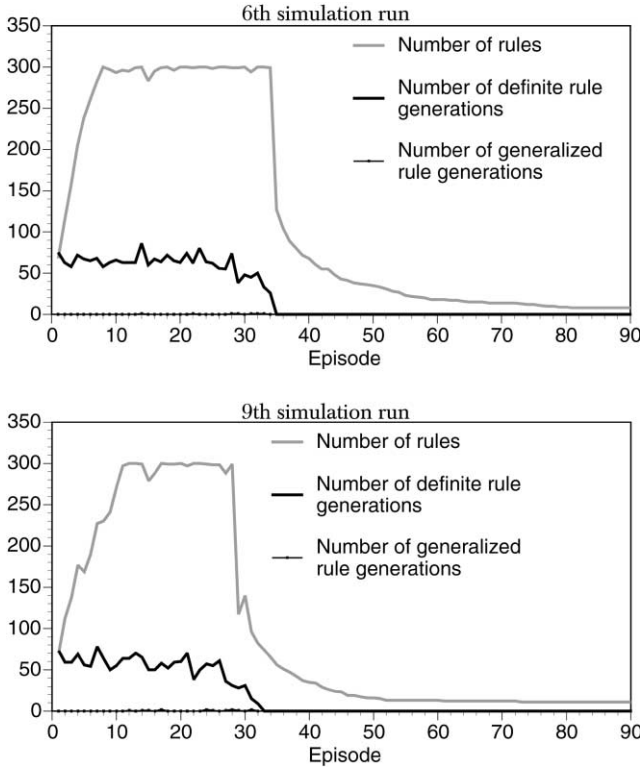

Fig. 12. Rule reproduction history.

respectively, dotted lines, thin lines, and thick lines. Here, the episodes at which the robot obtains the global reward are marked by down arrows. As can be seen, the number of punishments gradually decreases as learning progresses.

Fig. 12 plots the total number of rules, the number of generations of definite rules, and the number of generation of the generalized rules by, respectively, dotted lines, thin lines, and thick lines.

The dynamics of the learning process reveals a certain correlation between the total number of rules and the total number of steps necessary to reach the goal. Namely, the number of steps decreases soon after the total number of rules does. Also, there exists a correlation between the number of punishments and the number of generation of the new rules. This indicates, indirectly, that the exploration function of the indefinite rule is gradually subsumed by the exploitation function of the definite rules with high utilities. In fact, only particular definite rules, causing 'useful' behavior, trigger their actions and increase their utility. At the same time, 'irrelevant' rules decrease their utility and evaporate eventually. Thus, after some time the behavior acquired by the survivors becomes dominant.

The number of steps necessary to reach the goal is stabilized after, approximately, 35 episodes. The stabilization can be associated with acquiring certain gait patterns that can be seen from Fig. 13, showing the leg action history

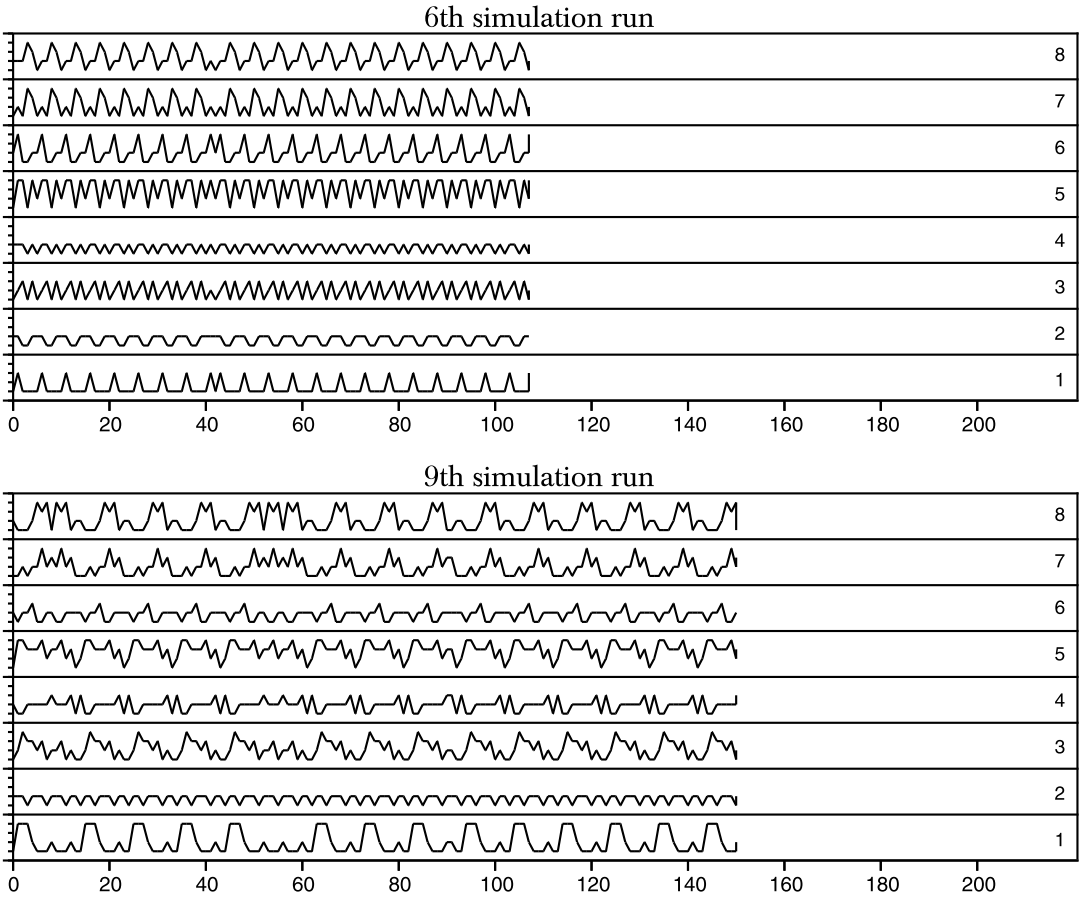

Fig. 13. Leg action history (simulation).

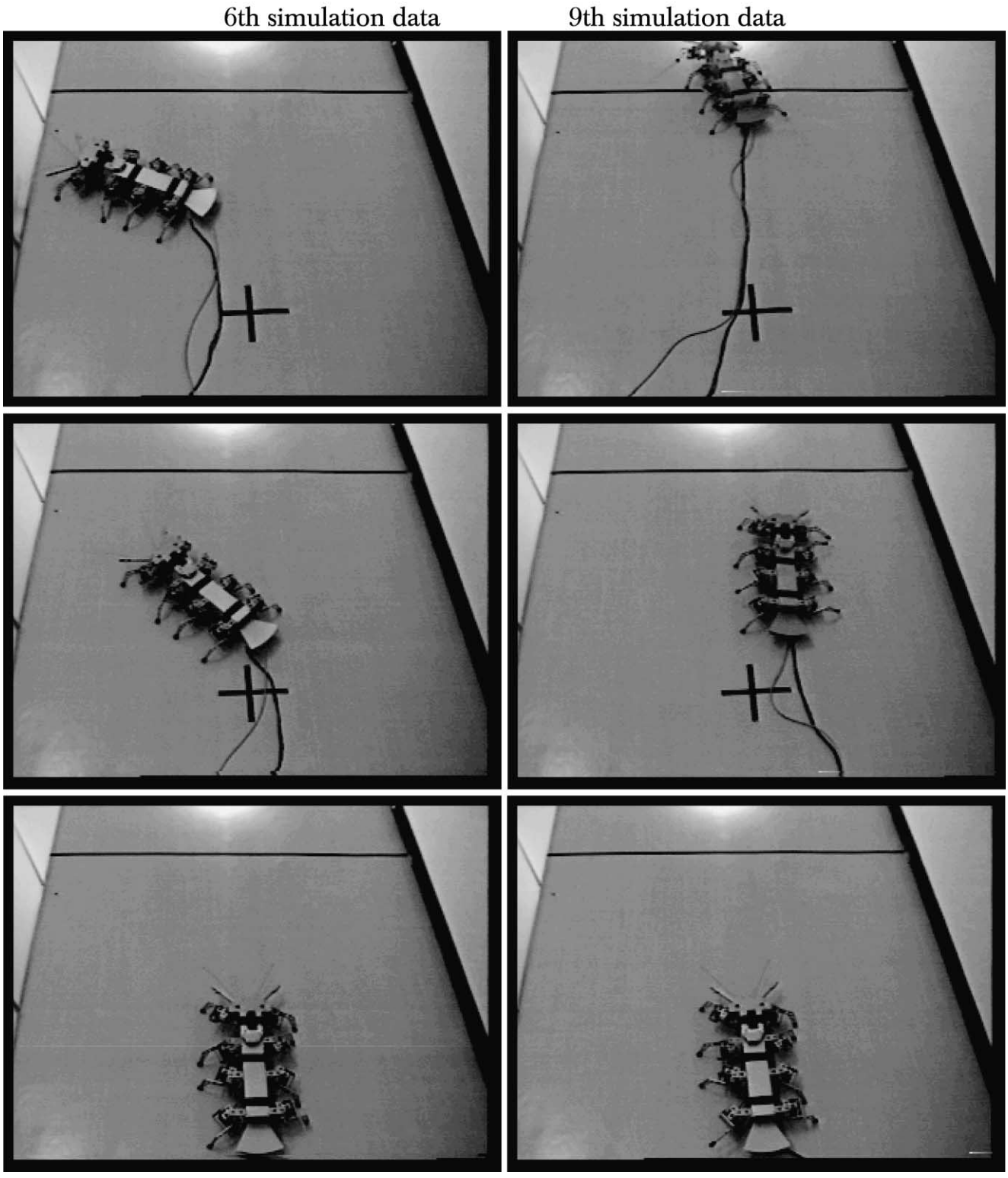

Fig. 14. Real robot behavior.

(action number at the corresponding step) for the 90th episode.

The ability of the controllers to reproduce the behavior evolved under simulation is now tested under experiments. The simulation data (the rules after the 90th episode) is downloaded to the control system of the OCT1-b robot, and one trial motion is performed. During the trial motion the robot is governed by the same classifier system with the same parameters.

Overall, of 10 simulation data seven were successful in

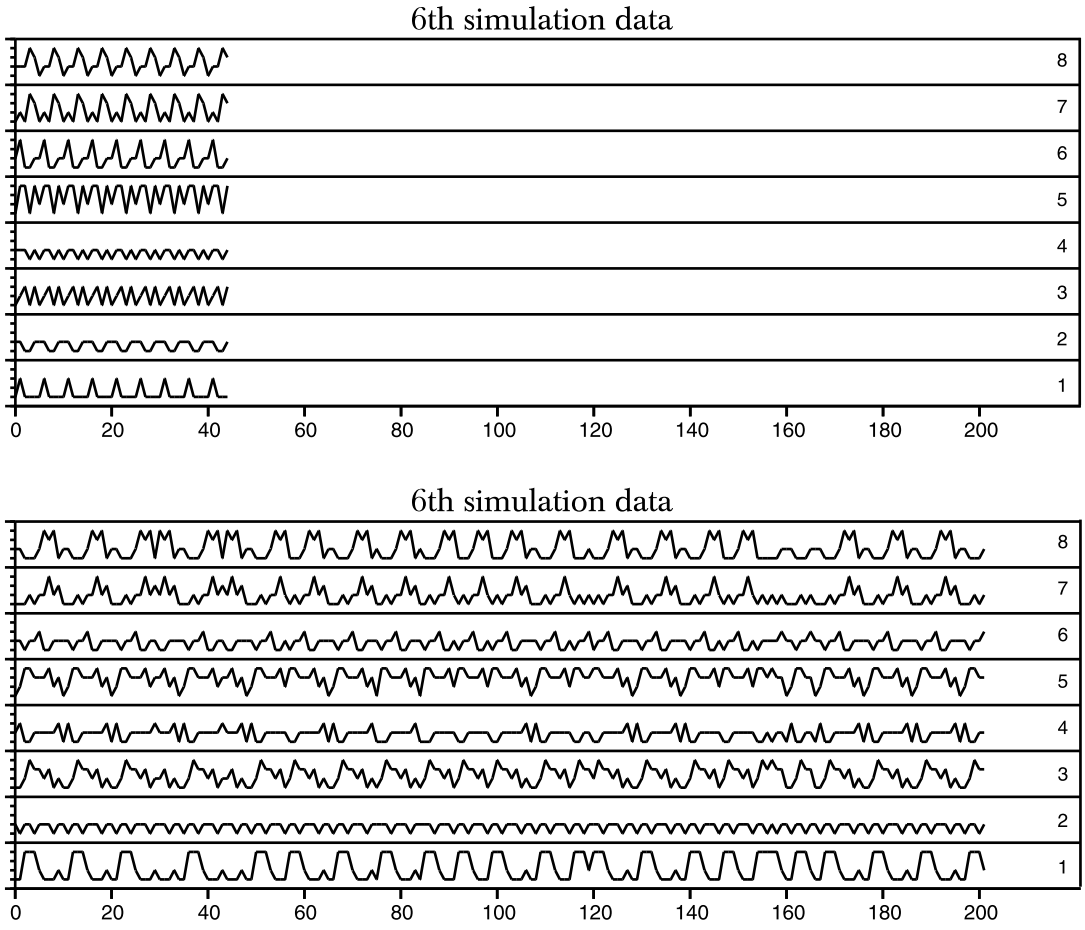

Fig. 15. Leg action history (experiment).

running the real robot. To illustrate the experimental results, we select one successful and one unsuccessful trial motions. Behavior of the robot corresponding to the sixth and nineth simulation data is shown in Fig. 14. It is inspected that a purely straightforward motion evolved under simulation is less robust comparing to the no so perfect simulation data. This brings us to the problem of correct modeling of noise and evolving the robot controller in a noisy environment.

Fig. 15 shows the robot gaits (leg angle time history) during the trial motion. In all the experiments, even in the unsuccessful ones, the robot tried to follow the gait patterns acquired during simulation. In a sense, it confirms that the gait patterns can be considered as the genetic 'material' of the robot control system.

6. Conclusions

Emergence of stable gaits in locomotion robots is studied in this paper. A classifier system, implementing an instance-based reinforcement-learning scheme, is used for sensory-motor control of an eight-legged mobile robot. The robot does not have a priori knowledge of the environment, its own internal model, and the goal coordinates. It is only assumed that the robot can acquire stable gaits by learning how to reach a goal area. During the learning process the control system is self-organized by reinforcement signals. Reaching the light source defines a global reward. Forward motion gets a local reward, while stepping back and falling down get a local punishment.

The control actions are specified at the leg level. As learning progresses, the number of the action rules in the classifier systems is stabilized to a certain level, corresponding to the acquired gait patterns. Thus, motion patterns of the global behavior (stable gaits) emerge, as the rules of the classifier system are self-organized during the learning process. Feasibility of the system proposed is tested under simulation and experiment. The virtual boat model is constructed and used for evolving the robot controllers under simulation. The experimental results validate feasibility of the model.

The results presented in this paper reflect our initial research on the emergent synthesis of motion patterns for locomotion robots. As such, there is enough room for critical points that should be addressed in the future. For example, the minimal model seems to work well for simple navigation task but its performance has not been tested for more complex behaviors. Next, the classifier system that we

used as a learning engine has many parameters that are difficult to tune to optimal values. Also, we did not use formal systematic procedures to evaluate the performance because this is a non-trivial problem. In this connection the result of work [31] may provide some valuable insights.

Speaking about further development of the framework of minimal simulation models, we think that the computational simplicity of the minimal models should be inversely proportional to the complexity of the controlled object. Considering the simplicity and complexity as time varying quantities that depend on the learning experience, the minimal models can also be viewed in an 'evolvable' way. The evolutionary component would allow for closing the control loop and reducing the gap between the learning controller and the model being used. From this point of view, it would be interesting to establish and exploit the duality between the models (action-to-state mappings) and the controllers (state-to-action mappings) in developing co-evolutionary scenarios. The key problem here is in the relation between the real evaluation and the self-evaluation. To get some basic understanding of how the frequency of real evaluation influences fitness of the evolved individuals under the computational time constraints, we plan to probe this problem on relatively simple one or two-dimensional control tasks.

References

[1] Manko DJ. A general model of legged locomotion on natural terrain. Boston: Kluwer Academic Publishers, 1992.

[2] Schwind WJ, Staudacher EM, editors. Workshop on Biology, Mechanics and Theory of Walking, Detroit, 10–15 May 1999. IEEE Conference on Robotics and Automation.

[3] Cruse H. What mechanisms coordinate leg movements in walking arthropods? Trends NeuroSci 1990;13(1):15–21.

[4] Cruse H, Bartling C, Cymbalyuk GS, Dean J, Dreifert M. A modular artificial neural net for controlling a six-legged walking system. Biol Cybernetics 1995;72:421–30.

[5] Cruse H, Kindermann T, Schumm M, Dean J, Schmitz J. Walnet — a biologically inspired network to control six-legged walking. Neural Networks 1998;11:1435–47.

[6] Matsuoka K. Sustained oscillations generated by mutually inhibiting neurons with adaptation. Biol Cybernetics 1985;52:367–76.

[7] Matsuoka K. Mechanisms of frequency and pattern control in the neural rhythm generators. Biol Cybernetics 1987;56:345–53.

[8] Bay JS, Hemami H. Modeling of a neural pattern generator with coupled nonlinear oscillators. IEEE Trans Biol Engng 1987; 34(4):297–306.

[9] Taga G, Yamaguchi Y, Shimizu S. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. Biol Cybernetics 1991;65:147–59.

[10] Kitamura S. Neuro oscillator and biped locomotion. Proceedings of the Sixth Japanese–German Seminar on Nonlinear Problems in Dynamical Systems. Theory and Applications, Tateshina, Japan, 1994. p. 121–6.

[11] Zielinska T. Utilisation of biological patterns in walking machines. Proceedings of the Fifth IFAC Symposium on Robot Control, SYROCO'97, vol. 3, Nantes, France, 1997. p. 1228–34.

[12] Yuasa H, Ito M. Coordination of many oscillators and generation of locomotory patterns. Biol Cybernetics 1990;63:177–84.

[13] Ilg W, Albiez J, Jedele H, Berns K, Dillmann R. Adaptive periodic movement control for the four legged walking machine bisam. Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, Michigan, USA, 1999. p. 2354–9.

[14] Beer RD, Callagher JC. Evolved dynamical neural networks for adaptive behavior. Adap Behavior 1992;1(1):91–122.

[15] Callagher JC, Beer RD. A qualitative dynamical analysis of evolved locomotion controllers. From Animals To Animats. Proceedings of the Second International Conference on Simulation of Adaptive Behavior, 1992.

[16] Lewis MA, Fagg AH, Bekey GA. Genetic algorithms for gait synthesis in a hexapod robot. Recent trends in mobile robots. New Jersey: World Scientific, 1994. p. 317–31.

[17] Callagher JC, Beer RD, Espenschied KS, Quinn RD. Application of evolved locomotion controllers to a hexapod robot. Robotics Autonomous Syst 1996;19:95–103.

[18] Sutton JH, Barto AG. Reinforcement learning: an introduction. Cambridge: MIT Press, 1998.

[19] Holland JH, Reitman JS. Cognitive systems based on adaptive algorithms. In: Waterman DA, Hayes-Roth F, editors. Pattern-directed inference systems. New York: Academic Press, 1978.

[20] Holland JH, Holyoak KJ, Nisbett RE, Thagard PR. Induction: processes of inference, learning, and discovery. Cambridge: MIT Press, 1986.

[21] Holland JH. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Michalski RS, Carbonell JG, Mitchell M, editors. Machine learning, an artificial intelligence approach, vol. 2. Los Altos: Morgan Kaufmann, 1986. p. 593–623.

[22] Booker LB, Goldberg DE, Holland JH. Classifier systems and genetic algorithms. Artif Intell 1989;40:235–82.

[23] Dorigo M, Colombetti M. Robot shaping. An experiment in behavior engineering. Cambridge: MIT Press, 1998.

[24] Mitchell TM. Machine learning. Boston: WCB/McGraw-Hill, 1997.

[25] Svinin MM, Yamada K, Okhura K, Ueda K. Decentralized reinforcement learning control and emergence of motion patterns. Proceedings of the SPIE International Symposium on Intelligent Systems and Advanced Manufacturing, vol. 3523, Boston, Massachusetts, 1998. p. 232–4.

[26] Jakobi N. Minimal simulations for evolutionary robotics. PhD thesis, University of Sussex, UK, 1998.

[27] Pin FG. A fuzzy behaviorist approach to sensor-based reasoning and robot navigation. In: Ghosh BK, Xi N, Tarn TJ, editors. Control in robotics and automation: sensor-based integration. London: Academic Press, 1999. p. 381–417.

[28] Wilson SW. Zcs: A zeroth level classifier system. Evolut Comput 1994;2(1):1–18.

[29] Cymbalyuk GS, Borisyuk RM, Muller-Wilm U, Cruse H. Oscillatory network controlling six-legged locomotion. Optimization of model parameters. Neural Networks 1998;11:1449–60.

[30] Svinin MM, Yamada K, Ueda K. Reinforcement learning approach to acquisition of stable gaits for locomotion robots. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol. 6, Tokyo, Japan, 1999. p. 936–41.

[31] Reich Y, Barai SV. Evaluating machine learning models for engineering problems. Artif Intell Engng 1999;13:257–72.