Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

Reinforcement Learning of Walking Behavior for a Four-Legged Robot

Hajime Kimura GEN@FE.DIS.TITECH.AC.JP
Toru Yamashita YAMA@FE.DIS.TITECH.AC.JP
Shigenobu Kobayashi KOBAYASI@DIS.TITECH.AC.JP
Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama 226–8502 JAPAN

Abstract

In this paper, we investigate a reinforcement learning of walking behavior for a four-legged robot. The robot has two servo motors per leg, so this problem has eight-dimensional continuous state/action space. We present an action selection scheme for actor-critic algorithms, in which the actor selects a continuous action from its bounded action space by using the normal distribution. The experimental results show the robot successfully learns to walk in practical learning steps.

1. Introduction

Reinforcement learning (RL) is a promising method for robots to obtain and improve control rules from interaction with their environment. The most standard approach of RL is the *value-function approach*, that is, estimating state-action value function and thereafter determining an optimal policy from it. Learning control in real world applications requires dealing with both continuous state and/or continuous action spaces. There are many works about the generalization techniques to approximate value functions over continuous state space; CMAC (Sutton & Barto 1998) is one of linear architectures (Bertsekas & Tsitsiklis 1996) and the others are neural-networks, instance-based methods (Santamaria et al. 1998), etc. However, the value function approach is infeasible in high-dimensional state-action space, because it needs enormous trials and memory-resources to find optimal deterministic policies.

Actor-critic algorithms are an alternative approach based on *policy gradient methods*, in which a parameterized stochastic policy is updated according to the gradient of the value function with respect to the policy parameters. Although this approach can find only a locally optimal policy, it has several practical features for solving RL problems which have high-dimensional state-action space. In this paper, we investigate an actor-critic algorithm and apply it to a four-legged robot. The robot has two servo motors per leg, so this problem has eight-dimensional continuous state/action space. We present a new action selection scheme for actor-critic algorithms, in which the actor selects a continuous action from its bounded action space by using the normal distribution. The experimental results show the robot successfully learns to walk in practical learning steps.

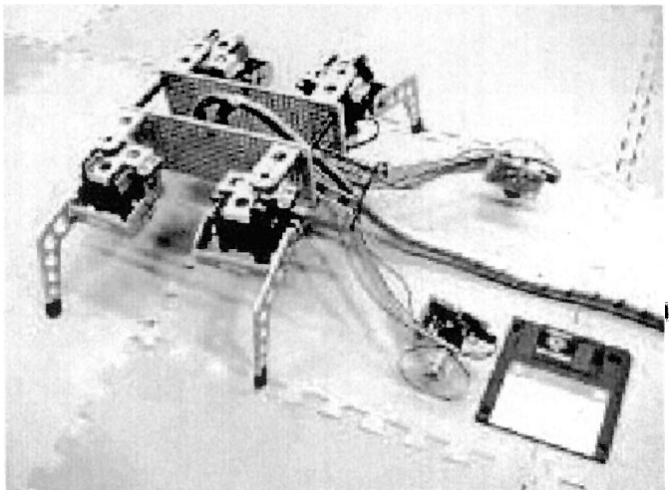

Figure 1. The robot. Each leg is controlled by two servo motors that react to angular-position commands.

2. Problem Formulation

2.1 Four Legged Robot

We consider a real four legged locomotion task shown in Figure 1. The objective of learning is to find control rules to move forward, but the controller does not know the dynamics ahead of time. The control rules are specified by a policy function, that is, a mapping from state to a probability distribution over actions. The controller improves its behavior through a process of trial and error. As shown in Figure 1, each leg is

0-7803-7061-9/01/$10.00 © 2001 IEEE

controlled by two servo motors that react to angular-position commands. Therefore, the robot has 8 degree of freedom.

At each time step, the learning controller observes current angular-position of 8 motors as the current state, and selects action according to its policy. The action is also the angular-position of 8 motors. Accordingly, the current state is equal to the previous action. At an interval of 0.5 sec, an immediate reward is given to the learner as a result of the action, and the time step proceeds to the next step. Two wheels shown in the right-hand side of Figure 1 detect a movement of the body, and generate the reward signal; the average of the moved distance of the wheels is the moved distance of the robot, and the differential of the wheels indicates the amount of turning the head. Since we want the robot to go straight, the immediate reward is defined as the average of the wheels minus absolute value of the differential of the wheels. The wheels have a diameter of 5cm, and turning full circle generates 200 pulses.

2.2 Markov Decision Problem

We modeled the robot's learning task as a reinforcement learning task in a Markov decision process shown in the following. Let $\mathcal{S}$ denote state space, $\mathcal{A}$ be action space, $\mathcal{R}$ be a set of real number. At each discrete time t, the agent observes state $s_t \in \mathcal{S}$, selects action $a_t \in \mathcal{A}$, and then receives an instantaneous reward $r_t \in \mathcal{R}$ resulting from state transition in the environment. In general, the reward and the next state may be random, but their probability distributions are assumed to depend only on s_t and a_t in Markov decision processes (MDPs), in which many reinforcement learning algorithms are studied. In MDPs, the next state s_{t+1} is chosen according to the transition probability $T(s_t, a, s_{t+1})$, and the reward r_t is given randomly according to the expectation $r(s_t, a)$.

The learning agent does not know $T(s_t, a, s_{t+1})$ and $r(s_t, a)$ ahead of time. The objective of RL is to construct a policy that maximizes the agent's performance. A natural performance measure for infinite horizon tasks is the cumulative discounted reward:

$$V_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \ , \qquad (1)$$

where the discount factor, $0 \leq \gamma \leq 1$ specifies the importance of future rewards, and V_t is the value at time t. In MDPs, the value can be defined as:

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \, r_t \middle| s_0 = s, \pi \right] \ , \qquad (2)$$

where $E\{\cdot\}$ denotes the expectation. The objective in MDPs is to find an optimal policy that maximizes the value of each state s defined by Equation 2. In this robot task, the state space is continuous, bounded and eight dimensional, and the action space is the same.

3. Reinforcement Learning Algorithms

3.1 Actor-Critic Methods

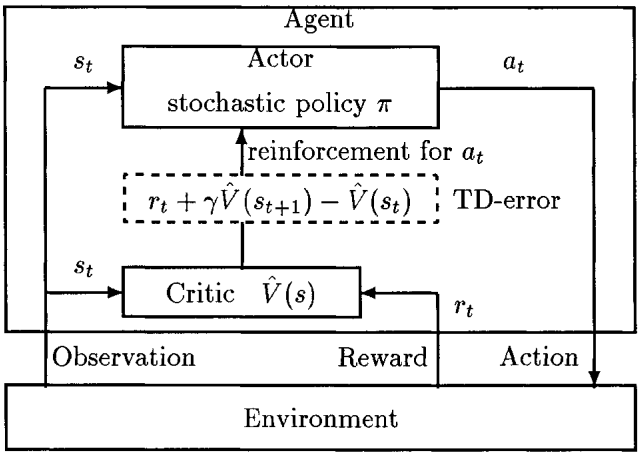

Figure 2. A standard actor-critic architecture. The critic estimates state values and provides the TD-error to the actor. The actor updates the policy using it. If the TD-error > 0, the actor raises probability of action a_t because the action a_t would lead the agent to a better state. Otherwise, it decreases the probability of a_t.

Actor-critic architecture (see Figure 2) is one of promising methods to solve reinforcement learning problems not only in MDPs, but also some POMDPs. The actor implements a *stochastic policy* that maps from state to a probability distribution over actions. The critic attempts to estimate the evaluation function for the current policy. The actor improves its control policy using critic's *temporal difference (TD)* as an effective reinforcement. In many cases, the policy improvement is executed concurrently with the policy evaluation, because it is not feasible to wait for the policy evaluation to converge.

Figure 3 specifies an actor-critic algorithm we used. It is noteworthy that both the actor and the critic adopt eligibility traces; obviously TD(λ) in the critic refers to use it, and the actor uses eligibility traces on policy parameters (Baird & Moore, 1999; Kimura & Kobayashi, 1998).

3.1.1 Learning Rules in the Critic

The calculation scheme of the TD(λ) in the critic is described in Step 3 and 5 in Figure 3. The parameter λ_v specifies the eligibility trace of the TD($\lambda = \lambda_v$).

3.1.2 Learning Rules in the Actor

In the actor, the policy is parameterized by parameters and updated according to the gradient of value function with respect to the policy parameters (Kimura and Kobayashi, 1998; Sutton, McAllester, Singh & Mansour,2000). Let $\pi(a|\theta, s)$ denote probability of selecting action a under the policy π in the state s. The

1. Observe state s_t, choose action a_t with probability $\pi(a_t|\theta, s_t)$, and perform it.
2. Observe immediate reward r_t, resulting state s_{t+1}, and calculate the TD-error according to

$$(\text{TD-error}) = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \ , \quad (3)$$

where $0 \leq \gamma \leq 1$ is the discount factor, $\hat{V}(s)$ is an estimated value function by the critic.

3. Update the estimating value function $\hat{V}(s)$ in the critic according to the TD(λ) method as:

$$\begin{aligned} e_v(t) &= \frac{\partial}{\partial w}\hat{V}(s_t) \ , \quad (4) \\ \overline{e_v}(t) &\leftarrow e_v(t) + \overline{e_v}(t) \ , \\ \Delta w(t) &= (\text{TD-error})\, \overline{e_v}(t) \ , \\ w &\leftarrow w + \alpha_v \, \Delta w(t) \ , \end{aligned}$$

where e_v denotes the eligibility of the parameter w in the function approximator $\hat{V}(s)$, $\overline{e_v}$ is its trace, and α_v is a learning rate.

4. Update the actor's stochastic policy by

$$\begin{aligned} e_\pi(t) &= \frac{\partial}{\partial \theta} \ln\Big(\pi(a_t|\theta, s_t)\Big) \ , \quad (5) \\ \overline{e_\pi}(t) &\leftarrow e_\pi(t) + \overline{e_\pi}(t) \ , \\ \Delta\theta(t) &= (\text{TD-error})\, \overline{e_\pi}(t) \ , \\ \theta &\leftarrow \theta + \alpha_\pi \, \Delta\theta(t) \ , \end{aligned}$$

where e_π is the eligibility of the policy parameter θ, $\overline{e_\pi}$ is its trace, and α_π is a learning rate.

5. Discount the eligibility traces as follows:

$$\begin{aligned} \overline{e_v}(t+1) &\leftarrow \gamma \lambda_v \, \overline{e_v}(t) \ , \\ \overline{e_\pi}(t+1) &\leftarrow \gamma \lambda_\pi \, \overline{e_\pi}(t) \ , \end{aligned}$$

where λ_v and λ_π ($0 \leq \lambda_v, \lambda_\pi \leq 1$) are discount factors in the critic and the actor respectively.

6. Let $t \leftarrow t+1$, and go to step 1.

Figure 3. An actor-critic algorithm using eligibility traces in both the actor and the critic. The critic is assumed to be a linear architecture.

$\pi(a|\theta, s)$ is taken to be a probability density function when the set of possible action is continuous. The agent improves the policy π by modifying the parameter θ.

The actor's learning rule is shown in the step 4 and 5 in Figure 3. The parameter λ_π specifies the actor's eligibility trace, but its features are somewhat different from TD(λ)'s. When λ_π is close to 0, the policy would be updated according to the gradient of the estimated value function $\hat{V}$, and when λ_π is close to 1, the policy would be updated by the gradient of the actual return, defined by Equation 1.

3.2 Implementation for the Robot

The vector $(s_1, s_2, \cdots, s_8)$ represents the angular positions of 8 motors as the state input, which are normalized as $-1 \leq s_i \leq 1$, where $i = 1, 2, \cdots, 8$. In the critic, the continuous state-space is discretized into 2^8 hyper square cells, and the state is encoded by a unit basis vector $(x_1, x_2, \cdots, x_{256})$ of length $= 2^8$, in which one component corresponding the current state is 1, and the others are 0. The estimated value $\hat{V}(s_t)$ using in Figure 3 is given by

$$\hat{V}(s_t) = \sum_{i=1}^{256} x_i w_i \ , \quad (6)$$

where w_i is a parameter for the value function approximation. Let $e_v(t)_i$ be the eligibility for the i^{th} parapeter w_i in Equation 4, i.e., $e_v(t) = (e_v(t)_1, e_v(t)_2, \cdots e_v(t)_i \cdots, e_v(t)_{256})$, then from Equation 6, it is given by

$$e_v(t)_i = \begin{cases} 1 & , \text{where } x_i = 1 \ , \\ 0 & , \text{where } x_i = 0 \ . \end{cases} \quad (7)$$

Let a vector $(a_{(1)}, a_{(2)}, \cdots, a_{(8)})$ be the angular positions of 8 motors as an action output, which are also normalized as $-1 \leq a_{(i)} \leq 1$, where $i = 1, 2, \cdots, 8$. The actor uses the following action-selection scheme; for each motor (i), the actor draws random samples from the normal distribution $N(\mu_{(i)}, \sigma^2_{(i)})$ until a sample which satisfies $[-1, 1]$ is generated, and takes the last one as the $a_{(i)}$. The parameters $\mu_{(i)}$ and $\sigma_{(i)}$ are given by the sigmoid function:

$$\begin{aligned} \mu_{(i)} &= \frac{2}{1 + \exp\left(-\sum_{k=1}^{8} s_k \theta_{k,(i)}\right)} - 1 \ , \\ \sigma_{(i)} &= \frac{1}{1 + \exp\left(-\theta_{9,(i)}\right)} \ , \end{aligned}$$

where $\theta_{k,(i)}$ $(k = 1, 2, \cdots 9)$ denotes policy parameters. Let P_{in} be probability of generating a sample within $[-1, 1]$ from the distribution $N(\mu_{(i)}, \sigma^2_{(i)})$ as

$$P_{in} = \int_{-1}^{1} \frac{1}{\sigma_{(i)}\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_{(i)})^2}{2\sigma^2_{(i)}}\right) dx \quad (8)$$

Then, the policy function is given by

$$\begin{aligned}
&\pi(a_{(i)}|\theta, s_t\ ) \\
&= (1+(1-P_{in})+(1-P_{in})^2+\cdots) \\
&\quad \times \frac{1}{\sigma_{(i)}\sqrt{2\pi}} \exp\left(-\frac{(a_{(i)}-\mu_{(i)})^2}{2\sigma_{(i)}^2}\right) \\
&= \frac{1}{P_{in}} \frac{1}{\sigma_{(i)}\sqrt{2\pi}} \exp\left(-\frac{(a_{(i)}-\mu_{(i)})^2}{2\sigma_{(i)}^2}\right), \quad (9)
\end{aligned}$$

where $a_{(i)}$ is bounded by $[-1, 1]$. Let $e_\pi(t)_{k,(i)}$ be the eligibility for the k^{th} policy parapeter $\theta_{k,(i)}$ associated with the i^{th} motor, i.e., θ and $e_\pi(t)$ shown in in Equation 5 represent a 9×8 matrix:

$$\theta = \begin{Bmatrix} \theta_{1,(1)} & \theta_{2,(1)} & \cdots & \theta_{9,(1)} \\ \theta_{1,(2)} & \theta_{2,(2)} & \cdots & \theta_{9,(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{1,(8)} & \theta_{2,(8)} & \cdots & \theta_{9,(8)} \end{Bmatrix}$$

$$e_\pi(t) = \begin{Bmatrix} e_\pi(t)_{1,(1)} & e_\pi(t)_{2,(1)} & \cdots & e_\pi(t)_{9,(1)} \\ e_\pi(t)_{1,(2)} & e_\pi(t)_{2,(2)} & \cdots & e_\pi(t)_{9,(2)} \\ \vdots & \vdots & \ddots & \vdots \\ e_\pi(t)_{1,(8)} & e_\pi(t)_{2,(8)} & \cdots & e_\pi(t)_{9,(8)} \end{Bmatrix}$$

From Equation 5 and 9, each $e_\pi(t)_{k,(i)}$ is given by

$$\begin{aligned}
e_\pi(t)_{k,(i)} &= \frac{\partial}{\partial\theta_{k,(i)}} \ln \pi(a_{(i)}|\theta, s_t) \\
&= \frac{\partial\mu_{(i)}}{\partial\theta_{k,(i)}} \frac{\partial}{\partial\mu_{(i)}} \ln\left(\frac{1}{P_{in}\sigma_{(i)}\sqrt{2\pi}} \exp\frac{(a_{(i)}-\mu_{(i)})^2}{-2\sigma_{(i)}^2}\right) \\
&= \frac{s_i\,(1-\mu_{(i)})(1+\mu_{(i)})}{2}\left(\frac{a_{(i)}-\mu_{(i)}}{\sigma_{(i)}^2} + P_{in}\frac{\partial}{\partial\mu_{(i)}}\frac{1}{P_{in}}\right) \quad (10)
\end{aligned}$$

where $k = 1, 2, \cdots 8$. The eligibility for $k = 9$ is

$$\begin{aligned}
e_\pi(t)_{k,(i)} &= \frac{\partial}{\partial\theta_{k,(i)}} \ln \pi(a_{(i)}|\theta, s_t) \\
&= \frac{\partial\sigma_{(i)}}{\partial\theta_{k,(i)}} \frac{\partial}{\partial\sigma_{(i)}} \ln\left(\frac{1}{P_{in}\sigma_{(i)}\sqrt{2\pi}} \exp\frac{(a_{(i)}-\mu_{(i)})^2}{-2\sigma_{(i)}^2}\right) \\
&= \sigma_{(i)}(1-\sigma_{(i)}) \\
&\quad \times \left(\frac{(a_{(i)}-\mu_{(i)})^2-\sigma_{(i)}^2}{\sigma_{(i)}^3} + P_{in}\frac{\partial}{\partial\sigma_{(i)}}\frac{1}{P_{in}}\right), \quad (11)
\end{aligned}$$

where $k = 9$. The second term of Equation 10 and 11 involve differentiating a definite integral with respect to $\mu_{(i)}$ or $\sigma_{(i)}$. We give it by an approximated numerical calculation, but the computational cost can be easily reduced by using a table, because the function is only depend on $\mu_{(i)}$ and $\sigma_{(i)}$ which are bounded.

Here we must draw attention to the fact that the eligibility is to divergent when $\sigma_{(i)}$ goes close to 0, because $\sigma_{(i)}$ is occupying the denominators of Equation 10 and 11. The divergence of the eligibility leads the algorithm to learning failure. For this reason, we adopt a heuristics that controls the step size of the update parameters so that it is proportional to $\sigma_{(i)}^2$. Then, the eligibilities are given by

$$\begin{aligned}
&e_\pi(t)_{k,(i)} \\
&= s_i\,(1-\mu_{(i)})(1+\mu_{(i)})/2\,(a_{(i)}-\mu_{(i)}) \\
&\quad + \frac{s_i\,(1-\mu_{(i)})(1+\mu_{(i)})}{2}\sigma_{(i)}^2\left(P_{in}\frac{\partial}{\partial\mu_{(i)}}\frac{1}{P_{in}}\right) \quad (12) \\
&\quad \text{where } k = 1, 2, \cdots 8, \\
&= (1-\sigma_{(i)})\left((a_{(i)}-\mu_{(i)})^2-\sigma_{(i)}^2\right) \\
&\quad + (1-\sigma_{(i)})\sigma_{(i)}^3\left(P_{in}\frac{\partial}{\partial\sigma_{(i)}}\frac{1}{P_{in}}\right), \quad (13) \\
&\quad \text{where } k = 9.
\end{aligned}$$

4. Results

We applied the algorithm to the robot on two conditions; one is on a carpet, the other is on a high-frictional rubber mat. Throughout the experiments, we use the following parameters: $\gamma = 0.9$, $\alpha_v = 0.1$, $\alpha_p = 0.002$, $\lambda_v = 1.0$, $\lambda_p = 1.0$.

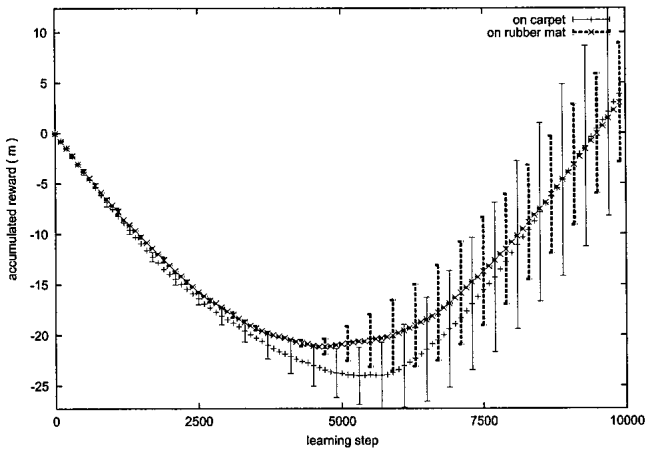

Figure 4. Learning curves averaged over 2 trials on the carpet and the high-frictional rubber mat. Each run consisted of 10000 steps (about 80 minutes).

Figure 4 shows learning curves for the robot on the carpet and the rubber mat. The vertical axis shows the accumulated reward translated into moved distance to the front, but the reward signal is given according to (average of the two wheel's moved distance to the front) − (absolute value of the differential of the wheels). Therefore, the falling curves do not (always) mean that the robot is moving backward in the early stage of the learning in Figure 4. On both conditions, the robot typically found good behavior in fewer than 6000 steps (about 50 minutes), and the actual velocity of the learned robot is about 5cm/sec after 10000

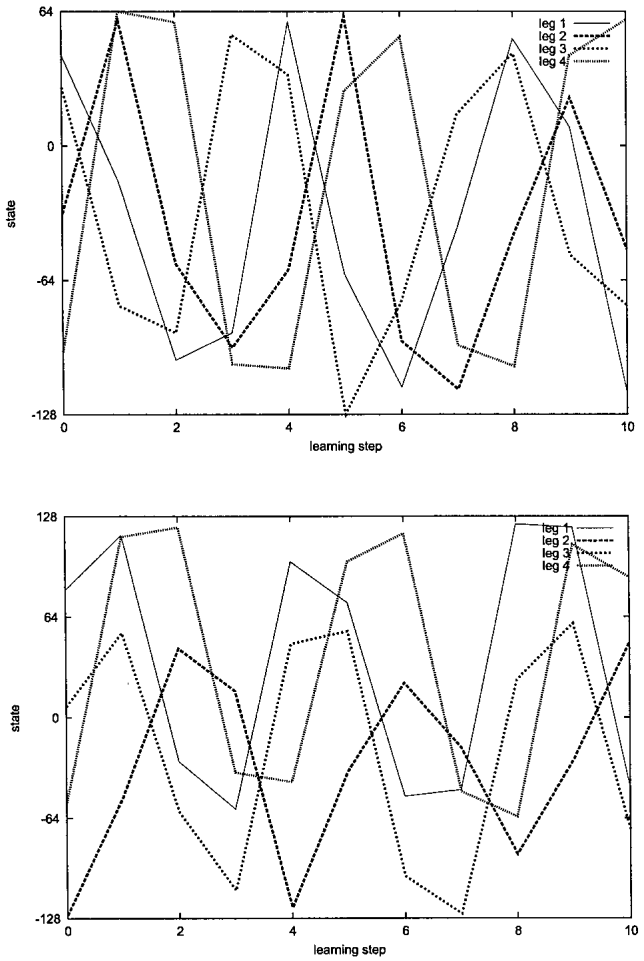

Figure 5. Gait pattern of the learned behavior on the carpet after 10000 steps (about 80 minutes). The top graph shows the movement for lifting legs, and the bottom graph is the movement for swinging legs. This gait seems a trot.

steps. The difference of the performance between the carpet and the rubber mat arise from the difference of the area. Since the rubber mat has small area, more loss of reward was enforced to keep moving on the mat.

In both case, the robot learned to behave like a turtle as shown in Figure 6, and Figure 5 is its gait pattern. The right-hand foreleg and the left hind-leg move in the same phase, and the left foreleg and the right hind-leg also move in the same phase. This gait seems a trot.

However, the behavior is slightly different between the carpet and the rubber mat. On the carpet, the robot tends to drag the body, because the body easily slips on it. It rather learns to weight down the supporting leg which contributes most to moving forward. On the other hand, on the high-frictional rubber mat, the robot keeps the body away from the ground. These observations support that the robot found good behavior adaptively for each condition.

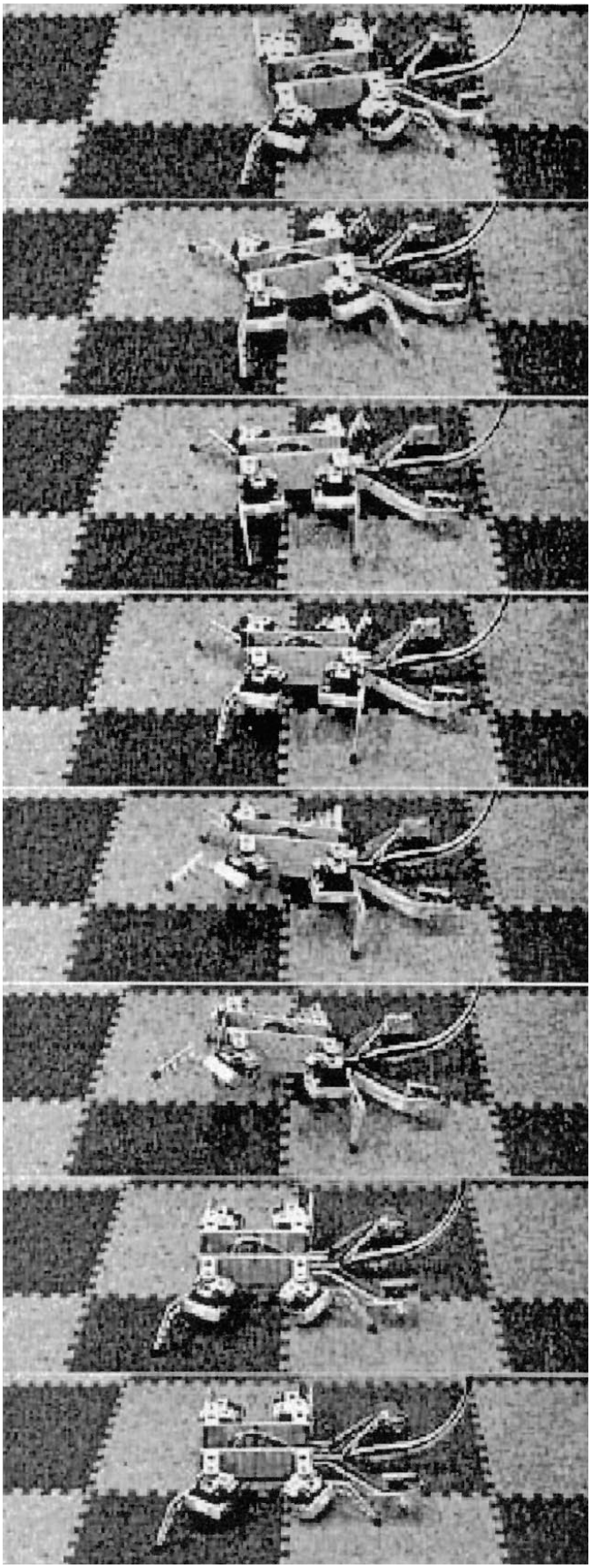

Figure 6. Learned behavior on the carpet after 10000 steps (about 80 minutes).

5. Discussion

Although there are several works making use of the normal distribution in the actor (e.g., Williams 1992, Doya 1997, Kimura & Kobayashi 1998, etc.), the naive implementation easily causes learning failure in our domain. The main reason is that the naive implementation in the actor does not use the information of boundaries of the action space. In the naive method, when the actor selects an action which is out of the boundaries, then the robot pretends to perform it, but the actually executed action is on the boundary. If the mean value of the distribution is moved to out of the boundaries, the robot can hardly move randomly, and learning would be failed. The proposed method avoids it by simply rejecting such actions. In compensation for avoiding failure, the additional computational cost is needed to calculate the eligibilities.

The coarse state-space quantization used in the critic is one of simple and promising methods to cope with high-dimensional state space, but it has two undesirable effects; one is that it makes the environment non-Markovian, the other is difficulties of finding good policies from such coarse value functions. The experimental results support that the actor's eligibility makes the robot less sensitive to these effects. Konda and Tsitsiklis (2000) proposed the other approach that the critic should use the actor's eligibility as the feature vector in order to approximate value (or q) functions. Their approach is mathematically sound and also promising in high-dimensional problems.

The following three subjects restricted flexibility of policy representation; one is the relatively small number of policy parameters, the use of the sigmoid function, and the other is the use of the normal distribution for the policy. These were given by experts as knowledge of the domain, and contributed reducing search space for learning within the practical steps. Thanks to the policy gradient theorem (Sutton et al. 2000), the actor can adopt any non-linear distribution functions as its policy under the assumption that the policy is differentiable with respect to its parameters. This flexibility of the policy function is useful for incorporating the expert's knowledge into the policy. Also supervised learning techniques would be available because the knowledge is provided by input-output mapping in many cases.

In our sense, the robot learned somewhat undesirable behavior such as dragging the body in the experiment. If the reward setting reflects conditions such as whether the body touches the floor or not, or costs of moving legs, then the learner will find better behavior in the context of the robotics.

References

Bertsekas, D.P. & Tsitsiklis, J.N. (1996). *Neuro-Dynamic Programming*, Athena Scientific.

Doya, K. (1997). Efficient Nonlinear Control with Actor-Tutor Architecture, *Advances in Neural Information Processing Systems 9*, pp. 1012–1018.

Kimura, H. & Kobayashi, S. (1998). An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imperfect Value Function, *15th International Conference on Machine Learning*, pp.278–286.

Konda, V.R. & Tsitsiklis, J.N. (2000). Actor-Critic Algorithms, *Advances in Neural Information Processing Systems 12*, pp. 1008–1014.

Santamaria, J.C., Sutton, R.S. & Ram, A. (1998). Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces, *Adaptive Behavior 6 (2)*, pp.163–218.

Singh, S.P. & Sutton, R.S. (1996). Reinforcement Learning with Replacing Eligibility Traces, *Machine Learning 22*, pp.123–158.

Sutton, R.S. & Barto, A. (1998). Reinforcement Learning: An Introduction, *A Bradford Book*, The MIT Press.

Sutton, R.S., McAllester, D., Singh, S. & Mansour, Y. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation, *Advances in Neural Information Processing Systems 12 (NIPS12)*, pp. 1057–1063.

Williams, R. J. (1992). Simple Statistical Gradient Following Algorithms for Connectionist Reinforcement Learning, *Machine Learning 8*, pp. 229–256.