Learning Behaviors for Environment Modeling by Genetic Algorithm

Seiji Yamada

Department of Computational Intelligence and Systems Science
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama 226-0026, JAPAN

Abstract. This paper describes an evolutionary way to lean behaviors of a mobile robot for recognizing environments. We have proposed AEM (Action-based Environment Modeling) which is an appropriate approach for a simple mobile robot to recognize environments, and made experiments using a real robot. The suitable behaviors for AEM have been described by a human designer. However the design is very difficult for them because of the huge search space. Thus we propose the evolutionary design method of such behaviors using genetic algorithm and make experiments in which a robot recognizes the environments with different structures. As results, we found out that the evolutionary approach is promising to automatically acquire behaviors for AEM.

1 Introduction

Primary research on an autonomous agent which recognizes a real environment have been done in robotics. The most studies have tried to build a precise geometric map using a robot with high-sensitive and global sensors like vision sensors [3]. Since their main aim is to navigate a robot with accuracy, the precise map is necessary. However, to recognize environments, such a strict map may be unnecessary. Actually many natural agents like animals seem to recognize environments only with low-sensitive and local sensors like touch sensors, and a precise geometric map is not necessary. In terms of engineering, it is important to build a mobile robot which can recognize environments only with the least sensors.

Thus we have tried to build a mobile robot which recognizes an environment only with low-sensitive and local sensors. Since such a robot does not know its position in an environment, it cannot build the global map of the environment using sensor data. Hence we proposed approach that a mobile robot can recognize an environment using *action sequences* generated by acting there. We call the approach AEM (Action-based Environment Modeling), and implemented it on a real mobile robot [15]. Using AEM, a robot can build a robust model of an environment only with low-sensitive and local sensors, and recognize an environment. In our research, the mobile robot is behavior-based and acts using *given* suitable behaviors (wall-following) for AEM in enclosures made of white

plastic boards. Then the sequences of the actions executed in each enclosure are obtained. They are transformed into real-value vectors, and inputted to a Kohonen's self-organizing network. Learning without a teacher is done and a mobile robot becomes able to identify enclosures. We fully implemented the system on a real mobile robot with two infrared proximity sensors, and made experiments for evaluating the ability. As a result, we found out the recognition of enclosures was done well.

However, in AEM, there is a significant problem: where the suitable behaviors come from. Although the design for such behaviors is very hard because of the huge search space, it has been done by a human designer thus far. Hence we propose an evolutionary design method of suitable behaviors for AEM using GA (Genetic Algorithm), and make preliminary experiments. For future implementation on a real mobile robot, we use a Khepera simulator in the experiments. From the experimental results, we found out that the evolutionary approach is promising to automatically acquire behaviors for AEM.

In the similar approach to AEM, several studies have been done in robotics [9][11] and A-Life [12]. Nehmzow and Smithers studied on recognizing corners in simple enclosures with a self-organizing network [12]. They used direction-duration pairs, which indicate the length of walls and shapes of past corners, as an input vector to a self-organizing network. After learning, the network becomes able to identify corners. Mataric represented an environment using automaton consisting landmarks as nodes [9]. Though the representation is more robust than a geometric one, a mobile robot must segment raw data into landmarks and identify them. Nakamura et al. utilized a sequence of sonar data in order to reduce uncertainties in discriminating local structure [11]. Though the sequence consists of sensor data (not actions), their approach is similar to AEM.

Wall-following and random-walking were used as suitable behaviors in [9][12][15] and [11] respectively. The behaviors were described by a human designer, and fixed. Hence they have the same significant problem that the design of the behaviors is very difficult as AEM.

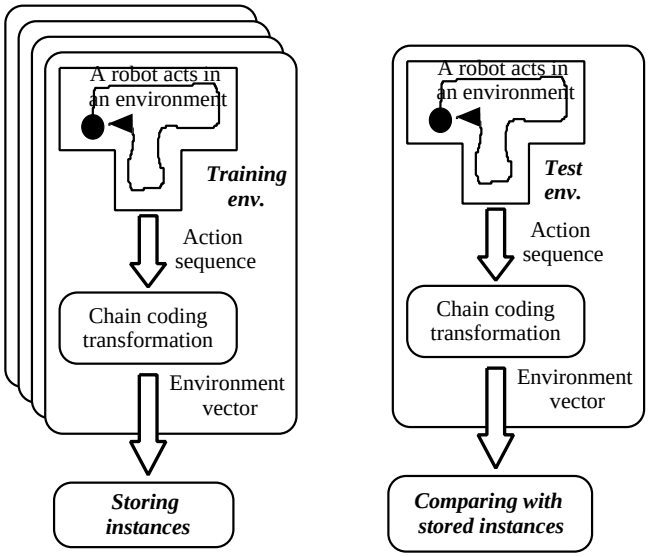
There are several studies for applying GP (Genetic Programming) [8] to behavior learning of a mobile robot [14][7][6][13]. Unfortunately, in the studies, very simple behaviors like obstacle avoidance were learned. In contrast with them, our aim is to learn the suitable behaviors to AEM, and the behaviors is complex one consisting of several kinds of primitive behaviors.

2 Action-base Environment Modeling and its Problem

In AEM, a mobile robot is designed with behavior-based approach [2]. The *behavior* means mapping from states to actions, and a human designer describes states, actions and behaviors so that sequences of executed actions can represent environment structure. Since AEM uses an action sequence, not sensed data, for describing an environment, the model is more abstract and robust than a geometric one [15].

An AEM procedure consists of two stages: a *training phase* and a *test phase* (Fig.1). It uses 1-Nearest Neighbor method [4], one of effective supervised-learning methods. In the training phase (Fig.1(a)), a robot is placed in a *training environments* having a *class* in which the environment should be included. Plural environments may be included in the same class. The behavior-based mobile robot acts in the environments using given behaviors, and obtains sequences of executed actions (called *action sequence*s) for each of them. The action sequences (lists of symbols) are transformed into the real-valued vectors (called *environment vectors*) using chain coding [1]. The environment vectors are stored as *cases*, and the training phase finishes.

Next, in the test phase (Fig.1(b)), a robot is placed in a *test environment*: one of the training environments. The robot tries to identify the test environment with one of training environments, and we call this task *recognition of an environment*. The identification is done by determined the most similar training environment (1-Nearest Neighbor) to the test environment. The similarity is evaluated with Euclidean distance between environment vectors. The robot considers that the most similar training environment has the same class to the test environment, and recognition of environments is done.



(a) Training phase (b) Test phase

Fig. 1. Overview of AEM

However, in AEM, there is a significant problem: where the suitable behav-

iors come from. Since the suitable behaviors depend on an environment structure which a robot should recognize, they have been described by a human designer thus far. However the task is very difficult for him or her. Because the search space for a suitable behavior is very huge: the computational complexity is $O(a^s)$, where a and s are the number of actions and states. Thus, we propose an evolutionary method to automatically acquire such behaviors using GA.

3 Describing a Mobile Robot, States and Actions

Using real mobile robots as individuals in GA is not practical because it is impossible to operate several tens of real robots over more than one hundred generations. Thus we use a simulator for acquiring behaviors, and intend to implement the learned behavior on a real mobile robot.

3.1 A Simple Mobile Robot: Khepera

We use a miniature mobile robot Khepera™(Fig.2, the radius and the height are 25mm and 32mm) which widely used in A-Life and AI. It has Motorola 68331 Micro processor, 256KByte RAM, and is programmable. As shown in Fig.3, it also has two DC motors (two black bars in Fig.3) as actuators and eight Infra-Red proximity sensors which measure both distance to obstacles and light strength. However, since the sensor data is imprecise and local, Khepera cannot localize itself in global map. In the later experiments, the simulator build for Khepera will be used.

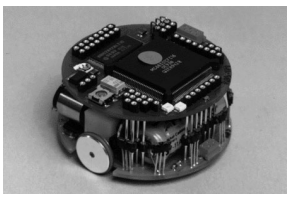


Fig. 2. Khepera

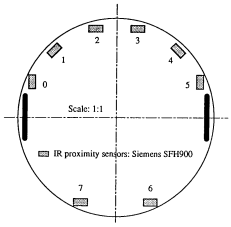


Fig. 3. Sensor positions

3.2 State Description

We describe a state with the range of a sensed value. For reducing the search space of behaviors, we restrict the number of states and actions as small as possible.

Though a sensor on Khepera returns 10 bit ($0 \sim 1023$) value for distance and light strength, the value is very noisy and crisp. Thus we transform the distance

value into binary vlues 0 and 1. The value "0" means an obstacle exists within 3cm from a robot. The value "1" means it does not exist. Furthermore only three (front, left and right) of eight sensors are used for reducing states.

Next states for light strength are described. Since only simple behaviors like approaching, leaving a light are considered suitable to AEM, we describe the state using the label of the sensor with the strongest light value and binary values which mean a light is "near" or "far". As well as states for distance, the only 4 sensors (front, left, right and back) are used. Additionally a state in which all of the sensors has almost same values is also considered. As a result, the number of states fot light is nine ($= 2 \times 4 + 1$). The total number of states is 72 ($= 2^3 + 9$)

3.3 Action Description

The following four actions are described. In experiments in the past research [15], we found the actions is sufficient for a mobile robot to do simple behaviors like wall-following. A mobile robot acts in an environment by executing the actions, and consequently an action-sequence like $[\,\mathrm{A2}, \mathrm{A4}, \mathrm{A1}, \mathrm{A1}, \cdots]$ is obtained.

- A1: Go 5mm straight on.
- A2: Turn 30° left.
- A3: Turn 30° right.
- A4: Turn 180° .

3.4 Transformation into an Environment Vector

The generated action-sequence is transformed into an environment vector. Let an action-sequence and its environment vector be $[a_1, a_2, \cdots, a_n]$ ($a_0 = 0$, $a_i \in$ {A1, A2, A3, A4}) and $\boldsymbol{V} = (v_1, v_2, \cdots, v_m)$ respectively. The vector values of $\boldsymbol{V}$ are determined by the following rules. They change the vector value when the direction of movement changes. These rules are considered a kind of chain coding [1]. For example, an action sequence $[\,\mathrm{A2}, \mathrm{A2}, \mathrm{A3}, \mathrm{A3}, \mathrm{A3}, \mathrm{A4}, \mathrm{A1}\,]$ is transformed into an environment vector $(\,1, 2, 1, 0, -1, 1, 1\,)$.

1. If $a_i = \mathrm{A1}$ then $v_i = v_{i-1}$.
2. If $a_i = \mathrm{A2}$ then $v_i = v_{i-1} + 1$.
3. If $a_i = \mathrm{A3}$ then $v_i = v_{i-1} - 1$.
4. If $a_i = \mathrm{A4}$ then $v_i = -v_{i-1}$.

As mentioned in § 2, in training phase, training environments are given to a robot for learning. The robots acts in the given environments, and stores the environment vectors transformed from the action sequences. Next, in test phase, test environments are given to the robot. It identifies the test environment with one of training environments by 1-Nearest Neighbor method using Euclidean distance as similarity.

4 Applying Genetic Algorithm to Acquire Behaviors

Since the number of states is 72 and that of actions is four, the number of possible behaviors is $4^{72} = 2.23 \times 10^{43}$. We have to search the suitable behaviors to AEM in such a huge search space. Genetic algorithm [5] is applied to the search because it does not need any domain-dependent heuristics.

4.1 GA Procedure and Coding Behaviors

In the followings, we describe GA procedure used in our research.

Step1 *Initializing population*: An initial population $I_1, \cdots, I_N$ are randomly generated.
Step2 *Computing fitness*: Compute the fitness $f_1, \cdots, f_N$ for each individual $I_1, \cdots, I_N$.
Step3 If a terminal condition is satisfied, this procedure finishes.
Step4 *Selection*: Using $f_1, \cdots, f_N$, select a child population C from the population.
Step5 *Crossover*: Select pairs randomly from C on probability P_{cross} (called *crossover rate*). Generate two children by applying crossover to each pair, and exchange the children with the pairs in C.
Step6 *Mutation*: Mutate the individuals in C based on mutation rate P_{mut}.
Step7 Go to **Step2**.

We set the following parameters which are considered effective experimentally.

- *Population size*: 50
- *Selection method*: Elite strategy and tournament selection (the size = 2).
- *Crossover operator*: Uniform crossover.
- *Crossover rate* P_{cross}: 0.8
- *Mutation rate* P_{mut} *per gene*: 0.05

Since we deal with deterministic action selection, not probabilistic, the behavior is mapping from a single state to a single action. Thus we use the coding in which one of actions {A1, $\cdots$,A4} is assigned to each state of $s_1, \cdots, s_{72}$ (Fig.4).

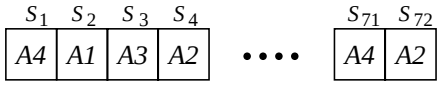


Fig. 4. A coded behavior

4.2 Defining a Fitness Function

Fitness is a very important for GA. Thus we have to carefully define the fitness function for AEM. We consider three conditions for suitable behaviors to AEM: *termination of actions*, *accuracy of recognition* and *efficiency of recognition*. The fitness functions for each conditions are defined, and then are integrated.

Termination of Actions A mobile robot needs to stop its actions by itself. Otherwise it may act forever in an environment, and no action sequence is obtained. Thus the termination of actions is the most important condition. We use *homing* which a robot returns his home. Because homing is considered a general method to terminate actions, and makes the length of an action sequence depend on the size of an environment. A method to terminate actions within a fixed length of an action sequence does not have such advantage. The homing concretely means turning to the neighborhood of a start point, and the termination is evaluated with the following function g. Its range is [0, 1], and returns 1 when a robot succeeded in homing in all the training environments.

$$g = \frac{\text{(No. of E-trial) + (No. of H-trial)}}{2 \times \text{(Total No. of trials)}}$$

where E-trial and H-trial means trials in which a robot escaped from the neighborhood of the start point and trials in which it succeeded in homing.

Accuracy of Recognition Another important criterion is accuracy of identifying test environments. The accuracy is evaluated with the following function h. Its range is [0, 1], and when $h = 1$, all the test environments were recognized correctly.

$$h = \begin{cases} \dfrac{\text{No. of correctly identified test env.}}{\text{Total No. of test env.}} & g = 1 \\ 0 & 0 \leq g < 1 \end{cases}$$

Efficiency of Recognition In AEM, a robot needs to *act* by operating actuators for recognizing an environment, and the actions significantly cost. Hence the actions should be as small as possible for efficiency of recognition. We hence introduce the following fitness function for evaluating the efficiency[1].

$$k = \begin{cases} 1 - \dfrac{\sum_{i=1}^{n} S_i}{n * S_{max}} & h = 1 \\ 0 & 0 \leq h < 1 \end{cases}$$

where S_i is the size of an action sequence obtained in ith test environment, S_{max} is the limited size of an action sequence, and n is the number of test environments. The function have range (0, 1] and has larger value as more efficient.

[1] The obstacle avoidance is implicitly evaluated by the function k because the collision increases the length of an action sequence.

We finally integrate three fitness function into the following fitness function f having range [0, 3], and it is used in this research. Since the function h (or k) takes 0 when g (or h) does not take 1, the function f is phased: the termination of actions is satisfied when $1 \leq f$, the recognition is completely correct when $2 \leq f$, and the recognition efficiency is improved when $2 \leq f < 3$.

$$f = g + h + k$$

5 Experiments with Simulation

It is impractical that we use real robots as individuals in GA. Thus we implement the system using a Khepera simulator [10], and make experiments in it. The parameters used in all experiments are described in the followings: the neighborhood of a start point is a circle with 100mm radius, and the limited size of an action sequence is 2000 actions.

In the simulator, the motor has $\pm 10\%$ random noise in velocity, $\pm 5\%$ one in rotation of robot's body. Furthermore an Infra-Red proximity sensor has $\pm 10\%$ random noise in distance, and $\pm 5\%$ one in light strength. These noise makes the simulator close to a real environment.

If a robot cannot return home within the limited size, the trial ends in failure. If the fitness value becomes more than two, the trial ends in success, and then both termination and accuracy are satisfied. In all the following experimental results, we show one of success trials, not averaged results.

In all experiments, we give each of training environments to a robot once. The robot acts in the environments, and the environment vectors transformed from the action sequences are stored as instances. Next each of the *training* environment is given to the robot as a *test* environment, and the robot identifies each of the test environments with one of the training environments. The start points and directions is fixed in bottom center and left.

Note that though the test environments are same to training environments, the action sequences are different because of the random noise in a simulator. Thus the obtained behavior by our method has robustness[14][6].

In all the experiments, we had 20 trials which have different initial conditions for GA and the generation was limited to 100. Some trials failed depending on the initial condition. In the followings, we present the succeeded trials for each experiment.

5.1 Exp-1: Environments with Different Contours in Shape

First we made experiments Exp-1 using environments with different contours in shape. Parts of five environments: { empty, L, L2, invL, small-empty } are given to a robot, and we investigated the ability to recognize them. Additionally twelve different shape environments including the four ones are used.

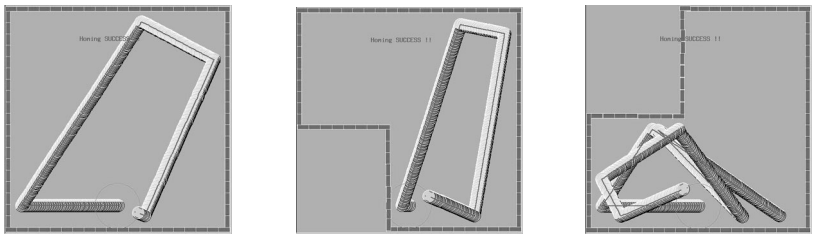
The experimental results are shown in Table.1, and Fig.5 indicates the trace of the robot with the maximum fitness in the expriment for the five environments.

The "GN" is the generation number in which the fitness value becomes more than two, and "Max fitness" means the maximum fitness value.

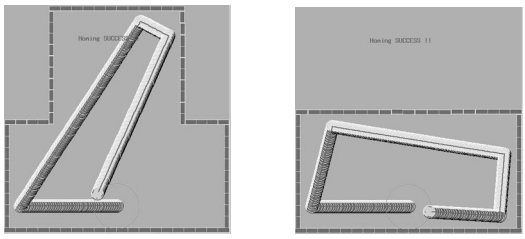
In such simple environments, the suitable behaviors for AEM were obtained within few generations. Seeing from Fig.5, different action sequences were obtained depending on the environment structure.

Table 1. Experimental results in Exp-1

Training environments		GN	Max fitness
(1)	{ empty, L }	1	2.84
(2)	(1) + L2	2	2.40
(3)	(2) + invL	3	2.44
	(3) + small-empty	2	2.46
	twelve environments	9	2.45



(a) empty (b) L (c) L2



(d) invL (e) small-empty

Fig. 5. Trace of actions in Exp-1

5.2 Exp-2: Environments with Different Lights in Number and Position

Next, by adding different lights to environments in number and position, we made five environments: { empty, 1-lamp, 2-lamp, 3-lamp, 4-lamp }. Exp-2 is made by using parts of the environments. A light was so strong that a robot can detect the light direction in any place. The experimental results are shown in Table.2. Fig.6 indicates the trace of the robot with the maximum fitness in the experiment for the five envirnments. In the figures, a black circle stands for a light.

Though the GN increased more than ones in Exp-1, the suitable behaviors were obtained. In Fig.6(a) ∼ Fig.6(c), the actions are slightly different. In contrast with them, the actions of Fig.6(d) and Fig.6(e) are significantly different from that of Fig.6(a) ∼ Fig.6(c). The lights in the left area seem to influence them strongly.

Table 2. Experimental results in Exp-2

Training environments		GN	Max fitness
(1)	{ empty, 1-lamp }	1	2.85
(2)	(1) + 2-lamp	7	2.53
(3)	(2) + 3-lamp	8	2.77
	(3) + 4-lamp	12	2.77

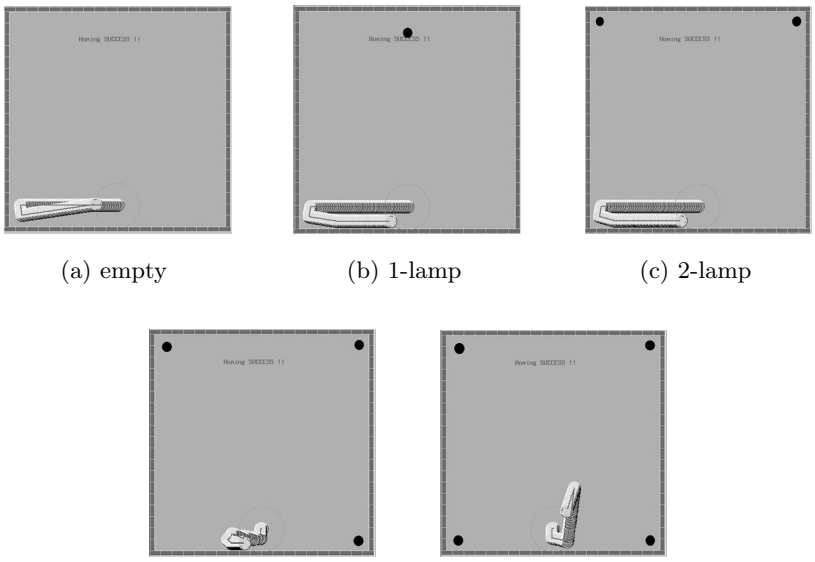
5.3 Exp-3: Environments with Different Contours and Lights

We set six environments: { empty, 1-lamp, L, L-1-lamp, invT, invT-1-lamp } by adding a light to three environments with different contours, and made experiments Exp-3 using them. Each environment is included different class and all of them should be distinguished.

As a result, GN was 13 and the maximum fitness was 2.38. Fig.7 shows the trace of the robot with the maximum fitness. Over all the environments, actions are very different mutually.

5.4 Exp-4: A Single Class Includes the Plural Training Environments

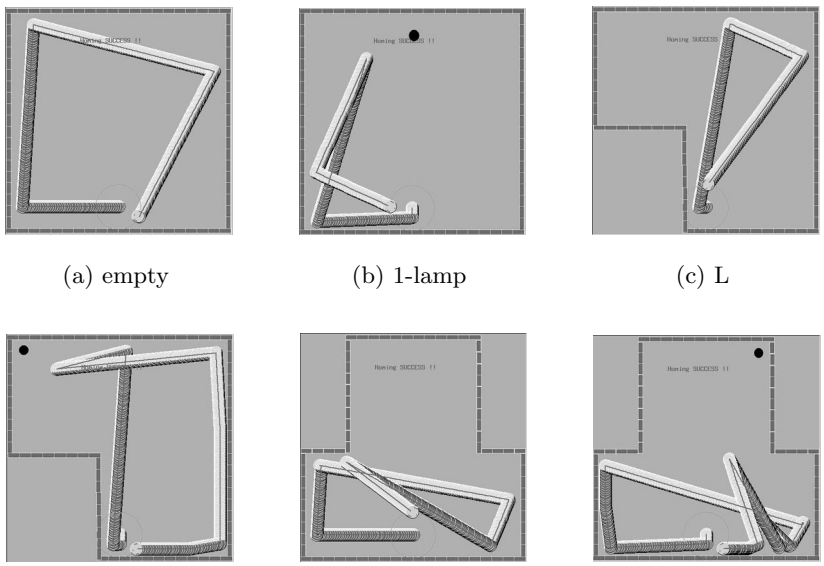
In Exp-1 ∼ Exp-3, all the environments are included in different classes. However, in Exp-4, plural environments are included in a single class. For recognizing such environments, generalization is necessary. We assigned three classes to six environments used in Exp-3: { empty, 1-lamp }, { L, L-1-lamp }, { invT, invT-1-lamp }, and made experiments. As a result, GN was 15 and the maximum



(a) empty (b) 1-lamp (c) 2-lamp

(d) 3-lamp (e) 4-lamp

Fig. 6. Trace of actions in Exp-2



(a) empty (b) 1-lamp (c) L

(d) L-1-lamp (e) invT (f) invT-1-lamp

Fig. 7. Trace of actions in Exp-3

fitness was 2.63. Thus our approach is valid for induction from sevral instances of a class.

6 Conclusions

We proposed the evolutionary acquisition of suitable behaviors to Action-based Environment Modeling. GA was applied to search the behaviors, and the simulated mobile robots were used as the population. States and actions were described for coding chromosomes. We made experiments in different environments in shape and lights, and found out our approach is effective to learn behaviors. However there are open problems like the followings.

- *Analysis and more complex domain*: We must analyze the experimental results for clarify how to acquire the suitable behaviors. Furthermore we will make experiments in more complex domains, and clarify problems there.
- *Robustness against initial conditions*: In all the experiments, start points and start direction were fixed. However, when a real robot is used, such a precise initial situation are impractical. Thus we must attempt experiments in which the initial situation is noisy. The learning may be difficult because a mobile robot acts sensitively to the initial situation [14].
- *Implementing learned behaviors on a real robot*: Implementation on a real robot is our final target. The gap between simulation and an real environment may make it difficult.

References

1. E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.
2. R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Transaction on Robotics and Automation*, 2(1):14–23, 1986.
3. J. L. Crowly. Navigation of an intelligent mobile robot. *IEEE Transaction on Robotics and Automation*, 1(1):31–41, 1985.
4. B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.
5. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
6. T. Ito, H. Iba, and M. Kimura. Robustness of robot programs generated by genetic programming. In *Genetic Programming 1996, Proceedings of the First Annual Conference*, pages 321–326, 1996.
7. J. R. Koza. Evolution of subsumpton using genetic programming. In *Proceedings of the First European Conference on Artificial Life*, pages 110–119, 1991.
8. J. R. Koza. *Genetic Programming*. MIT Press, 1992.
9. Maja J. Mataric. Integration of representation into goal-driven behavior-based robot. *IEEE Transaction on Robotics and Automation*, 8(3):14–23, 1992.
10. O. Michel. *Khepera Simulator v.2 User Manual*. University of Nice-Sophia Antipolis, 1996. (http://wwwi3s.unice.fr/~om/khep-sim.html).

11. T. Nakamura, S. Takamura, and M. asada. Behavior-based map representation for a sonor-based mobile robot by statistical methods. In *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 276–283, 1996.
12. U. Nehmzow and T. Smithers. Map-building using self-organizing networks in really useful robots. In *Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 152–159, 1991.
13. P. Nordin and W. Banzhaf. A genetic programming system learning obstacle avoiding behavior and controlling a miniature robot in real time. Technical report, Department of Computer Science, University of Dortmund, 1995.
14. C. W. Reynolds. Evolution of obstacle avoidance behavior: Using noise to promote robust solutions. In Jr. K. E. Kinnear, editor, *Advances in Genetic Programming*, volume 1, chapter 10, pages 221–241. MIT Press, 1994.
15. S. Yamada and M. Murota. Applying self-organizing networks to recognizing rooms with behavior sequences of a mobile robot. In *IEEE International Conference on Neural Networks*, pages 1790–1794, 1996.

This article was processed using the LaTeX macro package with LLNCS style