Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Applied Soft Computing 4 (2004) 35–47

Applied Soft Computing

www.elsevier.com/locate/asoc

Recognizing environments from action sequences using self-organizing maps

S. Yamada

National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan

Received 20 June 2002; received in revised form 2 December 2002; accepted 22 July 2003

Abstract

In this paper, we describe development of a mobile robot which does unsupervised learning for recognizing an environment from action sequences. We call this novel recognition approach action-based environment modeling (AEM). Most studies on recognizing an environment have tried to build precise geometric maps with high sensitive and global sensors. However such precise and global information may be hardly obtained in a real environment, and may be unnecessary to recognize an environment. Furthermore unsupervised-learning is necessary for recognition in an unknown environment without help of a teacher. Thus we attempt to build a mobile robot which does unsupervised-learning to recognize environments with low sensitive and local sensors. The mobile robot is behavior-based and does wall-following in enclosures (called *rooms*). Then the sequences of actions executed in each room are transformed into environment vectors for self-organizing maps. Learning without a teacher is done, and the robot becomes able to identify rooms. Moreover, we develop a method to identify environments independent of a start point using a partial sequence. We have fully implemented the system with a real mobile robot, and made experiments for evaluating the ability. As a result, we found out that the environment recognition was done well and our method was adaptive to noisy environments.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Environment modeling; Behavior-based robots; Self-organizing maps

1. Introduction

In robotics research, most studies on recognizing an environment like occupancy grids have tried to build a precise geometric map with high sensitive sensors [1–5]. In general, since information obtained by a mobile robot is significantly restricted, the robot needs to integrate the fractions of information for modeling the whole environment based on the absolute coordinate gained by such dead reckoning.

However, a simple mobile robot only with low sensitive and local sensors, like infrared proximity sensors, can not integrate such partial information on an environment. Because cumulative error increases and localization in the absolute coordinate is very hard. Many natural agents like animals, insects recognize their environments only with low sensitive sensors, and a geometric map may not be necessary for a simple and important task like recognition of an environment. Furthermore, in view of engineering, it is worthful to develop a simple mobile robot which is able to recognize environments only with inexpensive sensors. In many real environments like a room without light, the global information may not be obtained by computer vision.

Moreover, since a mobile robot should be adaptive to unknown environments without help of a teacher,

E-mail address: seiji@nii.ac.jp (S. Yamada).

1568-4946/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2003.07.001

it needs to learn to recognize new environments by itself. Thus unsupervised learning is necessary.

Hence, we attempt to build a mobile robot which does unsupervised-learning to recognize environments with low sensitive and local sensors [6,7]. The robot is behavior-based and does wall-following in enclosures. Then the sequences of actions executed in each enclosure are obtained. The action sequences are transformed into real-value vectors, and inputted to a Kohonen's self-organizing maps (SOM). Learning without a teacher is done and a mobile robot becomes able to identify the enclosures. Since we carefully define transformation from an action sequence into an input vector and SOM works well for generalizing input vectors, the learned network is robust against noise like obstacles. Furthermore, for more robust recognition, we develop a method to identify environments independent of a start point using a partial action sequence. In this paper, we call an enclosure, which a robot tries to recognize, a *room*. We call our approach of environment modeling action-based environment modeling (AEM).

We have fully implemented the system using a real mobile robot with infrared proximity sensors. For evaluating our approach, we made experiments including environments with obstacles as noise. As a result, we found out the environment recognition was done well and our method was adaptive to noisy environments.

Though AEM needs actions by a robot in an environment, it has advantages in many situations like the following.

- *Environments without rich information*: For example, though a computer vision system is useful for environment recognition with global information, it is not available in environments without light, objects with visual features and so on. AEM is applicable to such environments without rich information because it needs only action sequences executed using a low-level sensors like a proximity sensor.
- *Robots without sensitive sensor*: Since AEM does not need sensitive sensors for global and accurate information of an environment, it is suitable to robots having only low-sensitive and inexpensive sensors.

Nehmzow and Smithers studied on recognizing corners in a simple environment with SOM [8,9]. Furthermore, Smart and Hallam discussed the similarity between localization of a rat and their robot by making additional experiments [10]. They used direction-duration pairs, which indicate the length of walls and shapes (convex or concave) of past corners, as an input vector to SOM. After learning, the SOM becomes able to identify corners. However, the transformation from raw data to input vectors is significantly sensitive to noise like small obstacles. We propose another transformation which maintains better topology than their one. Furthermore we experimentally evaluate robustness of our method to obstacles.

Mataric represented an environment using automata consisting landmarks as nodes [11]. Though the representation is more robust than a geometric one, a mobile robot must segment raw data into landmarks and identify them. The segmentation and identification of landmarks is difficult for a robot only with low sensitive and local sensors. Thus, though the approach is similar to ours, it is not suitable for our purpose.

Tsuji and Li have done excellent studies on vision-based memorizing route scenes [12]. A mobile robot stores qualitative panoramic representation, and locates itself in the route by matching the memorized representation to that of the incoming scenes. If a robot has a vision system, their approach will be valid. However, a robot may not have such an expensive and global sensing system. Hence, we need to build the robot which recognizes the environment only with local sensors.

Kato studied segmentation of an environment using statistics on action sequences [13]. A mobile robot does random walk which was modified to cover wide region, and action sequences are obtained. Their approach is similar to AEM in utilizing action sequences. However, their aim is segmentation, and it is different from ours. Also, since their method needs very large number of actions, the implementation on a real mobile robot is hard. In contrast with it, our approach is fully implemented on a real mobile robot and various experiments are made with it.

Nakamura et al. proposed environment modeling using a sequence of sensed data obtained through action executions instead of action sequences [14]. A mobile robot executes an obstacle-avoidance behavior and a sequence of data sensed by a sonar-ring. Local structures are extracted by principal component analysis, and a global map is build by integrating them. Unfortunately, it is difficult to distinguish different rooms only in shape using their approach. AEM makes a mobile robot to distinguish such rooms.

Ordinary use of SOM in robotics is just classification of raw input data [15]. In such applications, the raw sensor data are directly inputted to SOM for compressing information. In contrast with such studies, in our research, the output of SOM directly indicates a particular room. Thus, we need to carefully define the transformation from a action sequence into an input vector so that SOM may classify input data into rooms well. The design of the transformation is a difficult and significant problem which does not occur in such straightforward applications.

2. Action-based environment modeling: AEM

In this paper, we consider it *environment recognition* that a mobile robot identifies an environment in which it acted previously. We assume segmentation of each environment have been done. Our purpose is to develop a simple mobile robot which is able to recognize environments only with imprecise and local sensors. To achieve this purpose, we can not use methods to build a geometrical map by integrating locally sensed information based on absolute coordinate [1–5]. Because a cumulative error significantly increases and localization in the absolute coordinate is very hard.

Hence, robust environment recognition method only with local and imprecise sensing is necessary. We propose a novel method that a robot acts by executing behaviors depending on environmental structure and models the environment using a sequence of executed actions [6,7]. By designing robust behaviors against sensing noise, an action sequence is utilized as abstract description on an environment. We call this approach AEM.

The basic idea of AEM is that a robot models an environment using not sensed data, but a sequence of executed actions. The overview of AEM is shown in Fig. 1. It is consisting of two phases: a *training phase* and a *test phase*. In a training phase (Fig. 1(a)), a behavior-based mobile robot [16] goes around in each room (called a *training environment*), which it should recognize, by executing wall-following behavior, and the sequence of executed actions (called an *action sequence*) is obtained. For each room, a single action sequence is obtained at least. Note that an action sequence is different from a sequence of executed behaviors. In general, a behavior is a rule: IF condition on sensed data THEN action, thus different behaviors may have the same action in the THEN part.

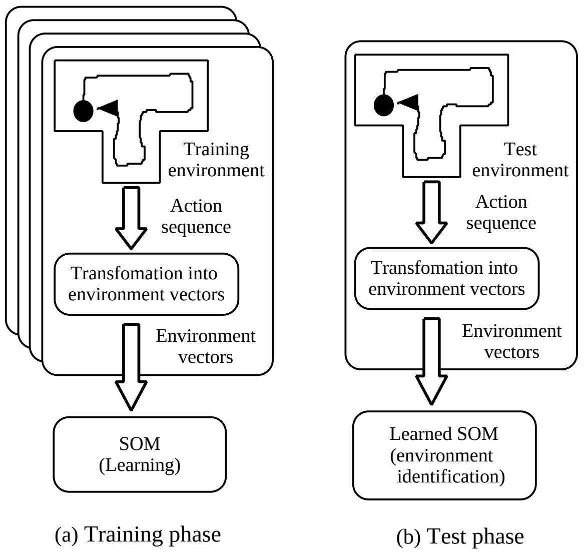


Fig. 1. AEM overview.

Next the action sequences (symbol lists) are transformed into the *environment vector*s, and they are given to Kohonen's SOM as input. There is no teacher giving the correspondence from input vectors to rooms. Unsupervised learning is done on the SOM using the Kohonen's learning rules. The set of environment vectors are repeatedly given to the SOM, and learning progresses. After learning, a winner node indicates one of the rooms in which the robot did wall-following.

After a training phase, a robot goes into a test phase (Fig. 1(b)). A *test environment*, which is one of training environments, is given to a mobile robot, and it does wall-following in the same way to a training phase. An action sequence is obtained and the environment vector is generated from it. The environment vector is input into learned SOM, and the test environment is identified with one of training environments by investigating the winner node on the input. Then environment recognition is done. The detail of the procedures will be mentioned later.

3. Wall-following by a behavior-based mobile robot

A mobile robot used in our research is shown in Fig. 2. It is designed for micro-mouse competition by Japan System Design Co. Ltd., and the size is 100 mm $(W) \times$ 165 mm $(D) \times$ 135 mm (H).

3.1. Sensors and actuators

The mobile robot has four infrared proximity sensors: two in front direction and other two in left and right direction on the both sides (Fig. 3). These sensors can sense obstacles within only 15 cm, and the sensed values are imprecise. Thus, this is considered a mobile robot, only with imprecise and local sensors, for which we attempt to develop AEM. Only two sensors on the left side are actually used for detecting a wall since the wall-following is fixed clockwise. The mobile robot also has an orientation sensor for steering a robot and an encoder for measuring movement distance roughly.

The four wheels of a mobile robot are shown in Fig. 3. The actuators consists of two stepping motors for driving left and right wheels independently and a dc motor for front and rear steering wheels (Fig. 3).

Also a mobile robot has CPU and DOS-like OS, and we can do programming to control it. The programming is done in a host PC and the program is loaded into a mobile robot through RS232C serial communication.

3.2. Behavior-based approach

Since the mobile robot needs to do wall-following even in a room where some obstacles exist, we use the behavior-based approach [16] which is known robust against the change of an environment. The behavior-based approach also can control a robot with low sensitive sensors, and have the advantage that the action sequence is invariant even when the geometric movement history of the mobile robot varies a little. In AEM, a *behavior* is described by a *reactive rule*: IF *sensing condition* THEN *action*, where an sensing condition can be directly checked on sensed values and an action can be executed by motor commands. The architecture of a behavior-based mobile robot is shown in Fig. 4. Sensed data are parallel input to behaviors and the sensing conditions are checked. Then an adequate rule is selected from applicable rules, and executed. A cycle of rule execution and sensing an environment is repeated quickly.

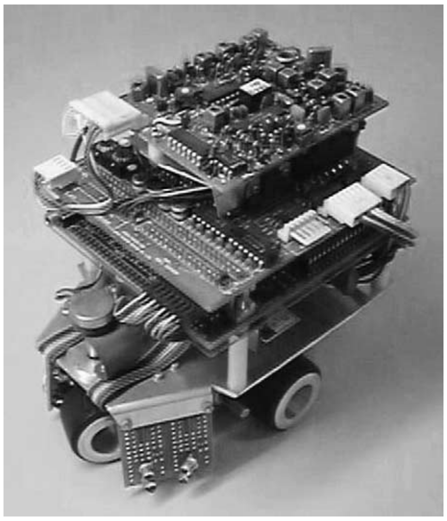


Fig. 2. A mobile robot.

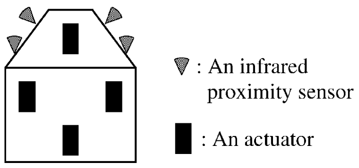
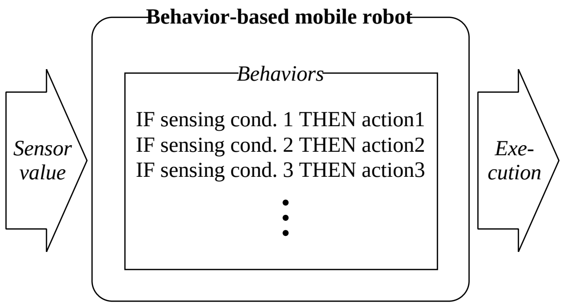


Fig. 3. Sensors and actuators (top view).


Behavior-based mobile robot
Behaviors
IF sensing cond. 1 THEN action1
IF sensing cond. 2 THEN action2
IF sensing cond. 3 THEN action3
Sensor value
Exe-cution
Fig. 4. A behavior-based mobile robot.

3.3. Wall-following

In this study, we assume rooms are distinctive in wall shape. Thus an action sequence needs to be characterized by the wall shape. For example, if a mobile robot does random-walk in rooms, the action sequences are very similar each other and environment recognition is hard. Hence we use wall-following behavior for AEM. When a robot returns into neighborhood of a start point, it stops.

Using an action sequence by wall-following, we can approximately estimate the size of a room, strictly the length of the wall, and information on the positions of corners is also acquired by designing behaviors corresponding to a corner.

We use four behaviors in the following for wall-following. The behaviors are periodically executed, and the mobile robot constantly goes forward. The steering is done relatively with current moving direction, and a robot follows a wall clockwise.

- Behavior-A (*turning in a concave corner*): *If* an obstacle within 10 cm in the front and within 10 cm on the left *then* turning 40 clockwise there. See Fig. 5(a).
- Behavior-B (*turning in a convex corner*): *If* no obstacle within 5 cm on the left and the right, and within 10 cm in the left front *then* turning 40 counterclockwise there. See Fig. 5(b).
- Behavior-C (*leaving a wall*): *If* an obstacle within 5 cm on the left *then* steering 13.5 clockwise. See Fig. 5(c).
- Behavior-D (*getting close to a wall*): *If* no obstacle within 5 cm on the left and an obstacle within 10 cm in the left front *then* steering 13.5 counterclockwise. See Fig. 5(d).

We experimentally verified that a mobile robot turned well at corners by executing the behavior A or B and followed a wall by executing the behavior C and D in turns. Although the above behaviors are very simple, a mobile robot is controlled well and smoothly follows walls.

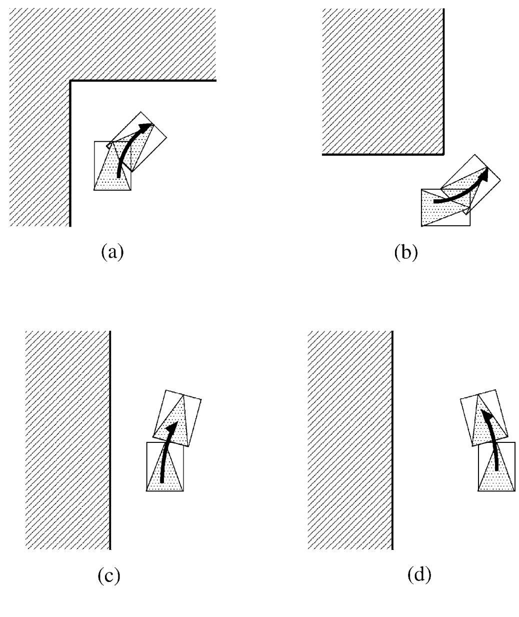


Fig. 5. Behavior A, B, C, D.

4. Generating environment vectors

An action sequence is considered to describe the structure of the environment. Thus, we can transform it into a vector, input the vector to SOM and the SOM is able to classify different rooms in shape. We call the vector an *environment vector*.

4.1. Transformation from an action sequence into an environment vector

Since a robot should be adaptive to unknown environments in which no other agents help it, it needs unsupervised-learning. Hence we introduce Kohonen's SOM for identifying rooms. The action sequence is a list of symbols, thus we need to transform it into a real-valued vector as input to SOM. Since SOM just classifies input vectors as keeping topology among them, the transformation of environment vectors should have carefully designed for good results. Furthermore, such a transformation needs to be robust against noise.

For satisfying preconditions above, we develop the following *behavior to input-vector* (BI)-*transformation*. Given an action sequence: $[r_1, r_2, \ldots, r_n]$ ($r_i \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$) and a environment vector: $\boldsymbol{I} = (v_0, v_1, v_2, \ldots, v_m)$ ($n \leq m$) the values of $\boldsymbol{I}$ are obtained using the following procedure. This BI-transformation is an improved version of chain coding (or a turning function) [17] by using relative rotation and invariant vector dimension rules for behavior C and D. If C and D increase vector dimension, more corners make more dimensions independently of the length of a wall and precise identification is hard. Hence, we add the invariant rules.

4.1.1. BI-transformation

(1) Initialize $i = j = 1$, $v_0 = v_1 = \cdots = v_m = 0$.
(2) If $i > n$, finish with output $I = (v_0, \cdots, v_m)$.
(3) If $r_i = A$, set $v_{j-1} = v_{j-1} + 0.5$, $i = i + 1$ and go to step (2).
(4) If $r_i = B$, set $v_{j-1} = v_{j-1} - 0.5$, $i = i + 1$ and go to step (2).
(5) If $r_i = C$ or $r_i = D$, set $v_j = v_{j-1}$, $i = i + 1$, $j = j + 1$ and go to step (2).

To explain the BI-transformation above, we show Fig. 6 standing for an environment (room), trajectory of a mobile robot, an action sequence and an environment vector generated from it. In a graph of an environment vector, x-axis and y-axis indicate the dimension and value of an environment vector, respectively. Also the number of actions is $n = 28$, and the dimension is set as $m = 30$. First all the values are initialize as $v_0 = v_1 = \cdots = v_{30} = 0$. Next the vector value is determined depending on an action r. Note that at step (3) or (4), the value increases or decreases because behavior A or B turns a mobile robot significantly, however the vector dimension j is not changed. In contrast with this, at step (5), the value is not changed and only dimension is increased. Hence the dimension of non-zero part of an environment vector (14 in Fig. 6) is less than the length of an action sequence. When the procedure is stopped at step (2), the remained vector value is left as zero, and the environment vector is output. Thus BI-transformation makes the dimension of a non-zero part of the environment vector indicate the length of a single side of a room.

In Fig. 6, a region in which the same vector value keeps correspond to a single side of a room and the change of value indicates the existence of a corner. Concretely, in Fig. 6, dimension 0–1, 2–3, 4–5, 6–7, 8–9, 10–13, 14–16 correspond to sides of a room and the changes of value in dimension 1–2, 3–4, 5–6, 7–8, 9–10, 13–14 correspond to corners of a room. The increasing change and the decreasing change correspond to a concave corner and a convex corner, respectively. See Fig. 11 for examples of environment vectors obtained through the experiments.

4.2. Influence of obstacles to an environment vector

We explain the robustness of BI-transformation using an example shown in Fig. 7. The trajectory of a mobile robot and an environment vector from the action sequences are indicated in Fig. 7. Fig. 7(a) stands for a room without obstacle and Fig. 7(b) stands for

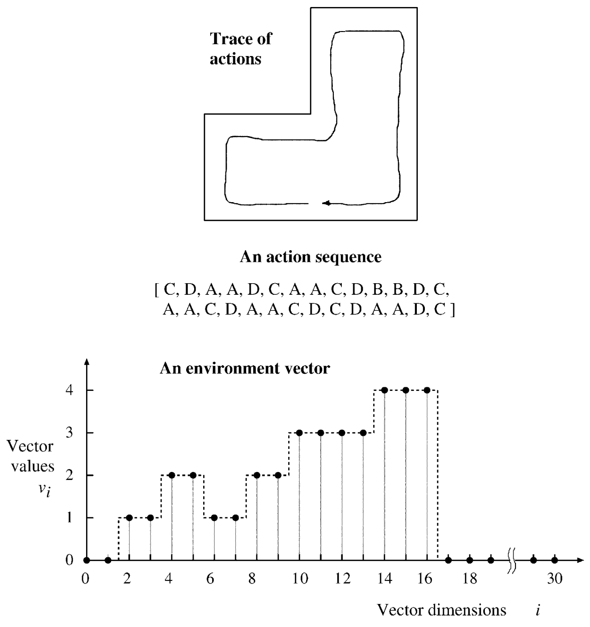


Fig. 6. An environment vector.

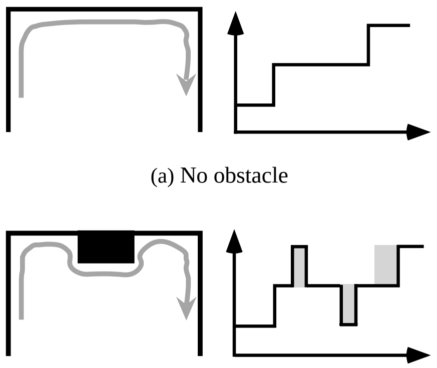


Fig. 7. Robustness of BI-transformation.

the same room including a obstacle on a wall. Thus, the mobile robot should identify them as the same one. The transformation needs to be defined so that the distance between the two environment vectors may be small. Using the BI-transformation results in the environment vectors as Fig. 7. The size of shaded areas in Fig. 7(b) indicates the distance, and it is relatively small.

Note that environment vector representation is more robust than symbol-based representation of an environment. If a robot uses more rigid transformation in which an environment vector is described with symbols of the length of walls, shapes (convex or concave) of past corners [8], noise by obstacles may make the distance significantly large. For example, consider symbol-based transformation for modeling an environment by using convex corner (VC), concave corner (CC), long wall (LW) and short wall (SW). By using this transformation, a symbol sequence [LW, CC, LW, CC, LW] is obtained from Fig. 7(a), and a symbol sequence [LW, CC, SW, CC, SW, VC, SW, VC, SW, CC, SW, CC, LW] is obtained from Fig. 7(b). Obviously, these two symbol sequences are quite different using a simple symbol-based matching method. Hence, we consider BI-transformation is robust, and the effectiveness will be experimentally verified.

5. Environment recognition using SOM

5.1. SOM: self-organizing maps

In this section, we briefly explain Kohonen's SOM [18]. It clusters large dimensional input vectors by mapping them to small discrete vectors, and is used widely in pattern recognition and robotics. SOM is a two-layered network consisting of an input layer and a competitive layer (Fig. 8). Any input node is linked to all competitive nodes, and all links have weights. As an input vector is given, input nodes have values corresponding to the input vector, and competitive nodes has values which stand for the distance between their weights and the input vector. A winner node having the minimum distance is determined, and weights of the winner node's neighbor nodes are updated.

Let an input vector and weights of links from all input nodes to a competitive node u_i be $\boldsymbol{I} = [v_1, v_2, \ldots, v_n]$ and $\boldsymbol{U}_i = [u_{i1}, u_{i2}, \ldots, u_{in}]$, respectively. First SOM computes the Euclidean distance between the input vector and competitive nodes. The distance is computed with the equation: $||\boldsymbol{I} - \boldsymbol{U}_i|| = \sqrt{\sum_j (v_j - u_{ij})^2}$.

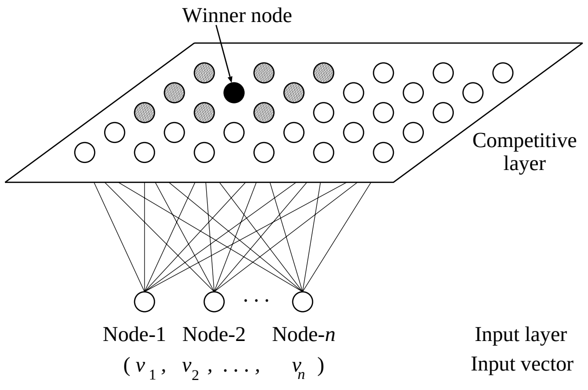


Fig. 8. The structure of SOM.

After a *winner node* with the minimum distance is determined, the weights of winner node's neighbor nodes are updated. The neighborhood is defined depending on the dimension of a competitive layer. If a competitive layer is two dimensional, a square having the winner node as the center is used as neighborhood. For example, in Fig. 8, the shaded neighbor nodes are determined in a minimal square with the black winner node as the center. The equation for updating the weights of neighbors is shown in the following, where α is a learning rate and u_{ij}^{new} means an updated weigh from u_{ij}^{old}.

$$u_{ij}^{\text{new}} = u_{ij}^{\text{old}} + \Delta u_{ij}$$

$$\Delta u_{ij} = \begin{cases} \alpha(v_j - u_{ij}), & i \text{ is neighbor} \\ 0, & \text{otherwise} \end{cases}$$

Update of the weights is done whenever a input vector is given. The learning rate and the size of the winner node's neighborhood is usually decreased as the learning progresses. The learning is finished when no update is done.

5.2. Identification in a test-phase

When learning begins to converge, the particular nodes become winner nodes frequently. We consider the winner nodes correspond to rooms, and call them *r*-nodes. Hence, the number of *r*-nodes is equal to the number of rooms recognized by a robot. In a test phase, at every time a test environment vector is given, a winner node of r-nodes is determined. We consider the winner *r*-node corresponds to a room in which the environment vector was obtained. In general, though the number of the learning iterations is very large, the identification in a test phase is very fast because it just needs to compute of distance between the environment vectors and *r*-nodes.

6. Utilizing a partial action sequence

With the method mentioned thus far, a mobile robot needs to completely go around in a room for environment recognition. However, if even a partial action sequence includes unique character, the environment recognition may be possible with it. Thus, we develop a method for recognizing environments with a partial action sequence like the following. The basic idea is that we use a environment vector which is shifted so that the longest wall may be positioned in the head and identify a room using threshold for similarity.

- A training phase.
  Learning with SOM is done using complete action sequences which are so that the longest wall may be positioned in the head.
- A test phase.
  1. A partial action sequence is incrementally obtained. An environment vector is generated by BI-transformation whenever the corner is detected. Find the longest wall region in an environment vector, and generate a partial environment vector by shifting the vector so that the longest wall region may be positioned in the head.
  2. Compute the distance between the partial environment vector and *r*-node's weight.
  3. If the distance is less than a threshold ξ, output the room corresponding to the *r*-node and this procedure stops. Otherwise go to step (1).

Using procedures above, wall-following may start at an arbitrary position near a wall. It is significant how to determine the threshold ξ. We determine it experimentally because of no analytical way. Since the execution of a partial action sequence is more inexpensive than that of a complete one, the above procedure makes the environment recognition more efficient.

7. Experiments

7.1. Setting and methods

Using the mobile robot mentioned earlier, we made experiments for evaluating the utility of AEM. The system consists of a mobile robot and an IBM-PC/AT PC (Pentium 133 MHz, 128M RAM) as a host computer. All programming was done on the host computer using C^{++}. The program of wall-following was down-loaded into a mobile robot through RS232C interface, and a robot autonomously followed walls. After wall-following, the action sequences were sent to the host computer and the learning by SOM was done there.

We built different rooms in shape with 13 cm height white plastic boards. Fig. 9 shows the scene that a mobile robot was doing wall-following. The robot stopped when it returns near a start point. It recognized the start location by dead reckoning using an orientation sensor and an encoder.

In a training phase, a mobile robot did wall-following one time for each of *N* rooms, and *N* action sequences and environment vectors were obtained in total. The *N* environment vectors were repeatedly input to SOM as input vectors in fixed order, and the

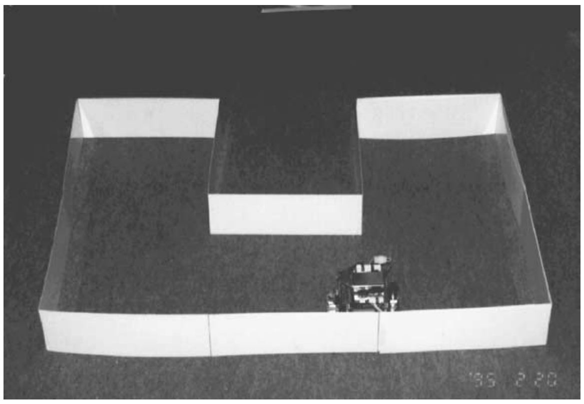


Fig. 9. Experimental environments.

SOM learned. Then, in a test phase, a mobile robot did wall-following five times for each of N rooms, and $5 \times N$ action sequences and environment vectors were obtained in total. A set of the $5 \times N$ environment vectors was input to the learned SOM as test environments and environment recognition was examined.

In the experiments, the largest length of action sequences obtained from rooms was about 1000, and this was the minimum dimension of an input vector. The parameter setting of SOM is shown in the followings.

- The number of input nodes: 1300.
- The number of competitive nodes: 128.
- Topology of a competitive layer: linear and circular with one-dimension.
- Initial weights of competitive nodes: initial weights are generated from uniform random number within the range 0.5 ± 10, 1.5 ± 10, and $4.0 \pm 10\%$.
- The number of learning iteration (the total number of input vectors): 100,000 which is known valid experimentally.

Also the learning rate α_t and the size d_t of winner node's neighborhood at t iteration were decreased as learning progresses using the following formulas. The d_t is the length of a side of a square:

$$\alpha_t = 0.8\left(1 - \frac{t}{100,000}\right); \quad d_t = 16\left(1 - \frac{t}{100,000}\right)$$

Fig. 10 shows the convergence as learning iteration progresses, where D of the y-axis is the square of Euclidean distance between the input node and the weighs of the winner node and the x-axis stands for the number of iterations for updating the weights in SOM.

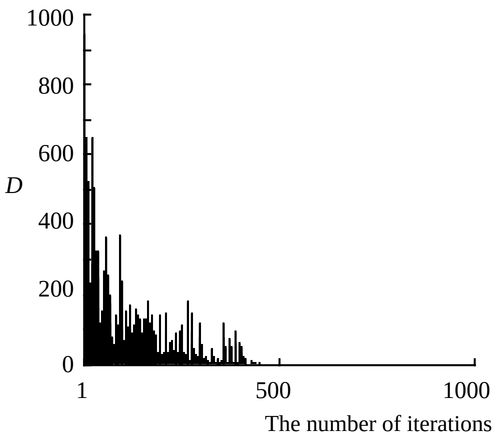


Fig. 10. Convergence of learning.

7.2. Recognition of seven rooms

First of all, the seven rooms shown in Fig. 11 were recognized. Fig. 11 stands for the shape of rooms, trajectories and environment vectors. The real length of a side of a large square in Fig. 11 is 146 cm,

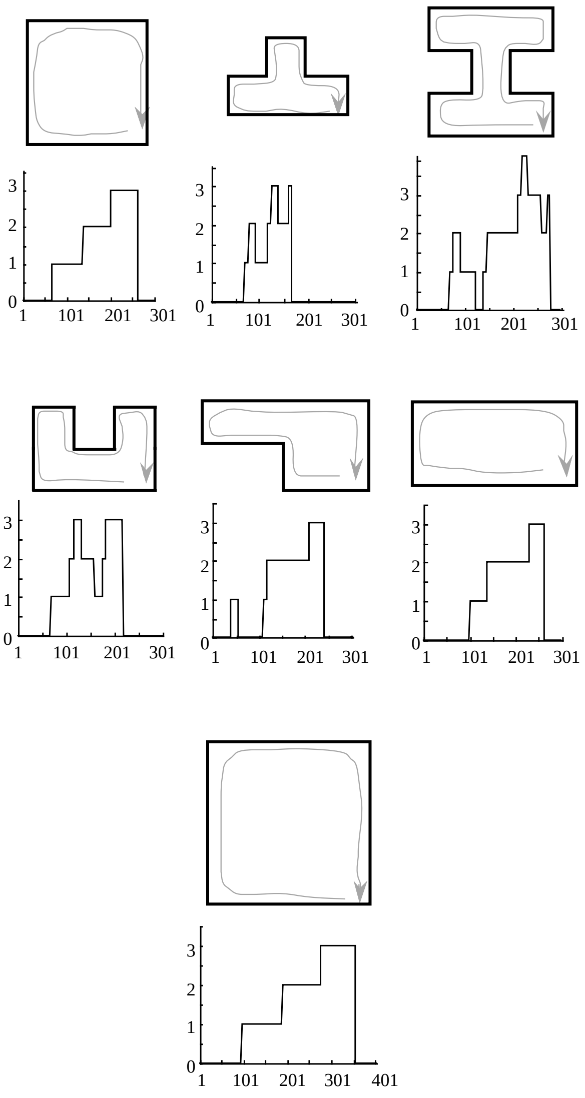


Fig. 11. Seven rooms.

and the rooms are figured with the same scale. In this experiment, complete action sequences were used.

As a result, for all settings of topology of a competitive layer and initial weights of competitive nodes, the recognition rate was 100%. Note that this task is not easy because the action sequences and environment vectors for an identical room are slightly different mutually because of sensor noise and failure of executing actions.

7.3. *Recognition of twenty rooms*

Furthermore, experiments were made using twenty rooms in Fig. 12, where the length of a side of E-3 is 146 cm and the rooms are figure with the same scale. Complete action sequences were used, and a sequence of [E-4, E-5, E-1, E-12, E-2, E-8, E-9, E-3, E-7, E-11, E-10, E-20, E-19, E-18, E-14, E-17, E-13, E-16, E-15, E-6] was input to SOM repeatedly. Table 1 shows the experimental results on recognition rate for various settings of SOM.

As seeing from the table, the recognition rate is not significantly dependent on topology of a competitive layer and initial weights of competitive nodes. Also the recognition rate became less than that on seven rooms because of many similar rooms in shape.

7.4. *Recognition with partial action sequences*

We made experiments for evaluating recognition with partial action sequences using twenty rooms in the last experiment. The results of each room on recognition rate are shown in Table 2, where the topology of a competitive layer is linear and the initial weight of a competitive node is $1.5 \pm 10\%$.

Table 1
Recognition rates on twenty rooms for various initial weighs, topologies of a competitive layer

Initial weights (%)	Linear topology (%)	Circular topology (%)
0.5 ± 10	74	69
1.5 ± 10	74	74
4.0 ± 10	74	69

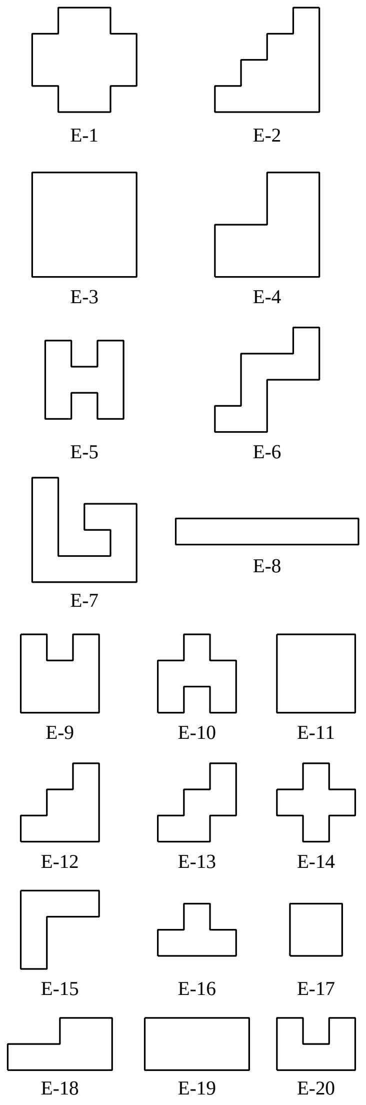


Fig. 12. Twenty rooms.

The average values of recognition rate and *partiality* are 60.8 and 47.8%, respectively, where

$$\text{partiality (\%)} = \frac{\text{the length of partial action sequence}}{\text{the length of complete action sequence}} \times 100$$

The reason of deterioration in recognition rate is considered that the procedure in Section 6 sets the undetermined vector values to zero. However, since the recognition rate of a complete action sequence is less

Table 2
Recognition rates on partial action sequences

Room	Recognition rate (%)
E-1	50
E-2	83
E-3	100
E-4	100
E-5	33
E-6	33
E-7	83
E-8	100
E-9	100
E-10	67
E-11	100
E-12	83
E-13	17
E-14	67
E-15	17
E-16	17
E-17	67
E-18	17
E-19	67
E-20	17

than 80%, this rate is not significantly low. Also, seeing from Table 2, the recognition rate strongly depends on the shape of a room. Recognition of the small rooms tends to be hard.

7.5. Environments with obstacles

Though the test data used thus far included noise, it was not so much. In this experiment, we dealt with more noise like obstacles. We located obstacles in the seven rooms, and a mobile robot did wall-following in the rooms. The nine complete action sequences were obtained and transformed into environment vectors. The environment vectors were given to the SOM which was trained in Section 2 as test environment vectors.

As a result, five rooms were correctly identified. Fig. 13 shows success and failure examples. In Fig. 13(a), the robot correctly recognized the room including either a single obstacle or two small cubical obstacles on a wall. However, in Fig. 13(b), the robot failed to recognize a room including either two large obstacles or an obstacle located obliquely to a wall. The large or many obstacles changed the shape of rooms from the original one, thus we consider the failure of recognition is natural.

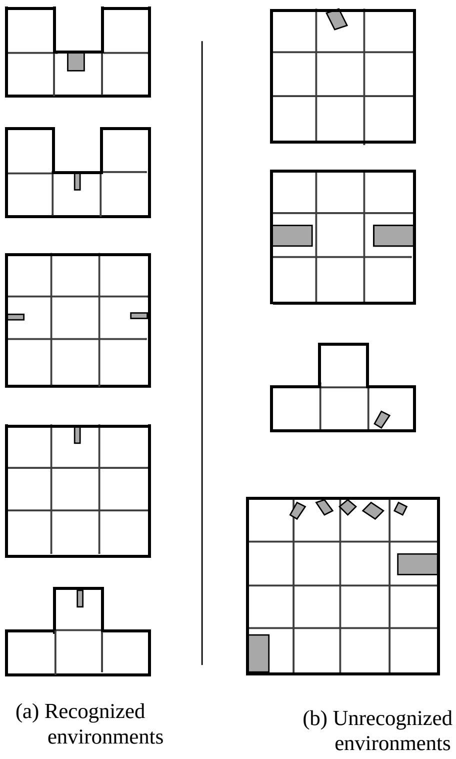


Fig. 13. Environments with obstacles.

The robot can not evaluate the results by itself since it does not have purpose to recognize the rooms. This problem will be discussed in next section.

8. Discussions

In this section, we discuss open problems of our study.

8.1. Toward real environments

All the experiments were made in a toy world of enclosures with white plastic boards. We are planning to make experiments in real world like real office environment consisting of rooms, corridors, desks and chairs. However, we consider the robot used in this study is not adequate for such environment because it is too small to does wall-following efficiently and effectively in a real room. Thus, we need to implement a suitable mobile robot to such experiments.

8.2. Open environments

Though we assume that an environment is a closed region, a real environment may not be closed, e.g. a room with doors. The assumption is used only to terminate wall-following. Thus we need to develop a method to terminate robot's action in real environments.

8.3. Restriction on the shapes of rooms

The shapes of rooms are restricted to rectangles with right corners. If a room consists of curves like a circle, neither the behavior A nor B will be executed and a robot will not use information of corners. In these cases, since only the length of the periphery is obtained, the robot can not distinguish a circle from a ellipse with the same periphery in length. We need to improve the behaviors and BI-transformation.

8.4. Suitable behaviors for recognizing environments

We used wall-following as behaviors for recognizing rooms, however we do not consider it is best one. There may be a more robust and suitable behavior for characterizing environments. Currently, we are developing evolutionary acquisition for suitable behaviors using genetic algorithm [19].

8.5. What is the identification of environment for

It is a very radical problem that the robot does not know what it recognizes the environments for. In supervised-learning, a teacher explicitly tells the robot the class in which the environment should be included, and thus the robot does not need to know the purpose. However, an autonomous robot should have the purpose of recognizing environments, and they it classifies the environments under the purpose. Since a robot does not have the purpose in our research, it can not evaluate the result of classifying rooms by itself. Hence, we need to give it the purpose of recognizing environments. It may be natural to interaction with evolution [20] though the experiments with a real mobile robot are difficult.

9. Conclusion

We proposed AEM approach to recognize environment and built a mobile robot which learned to recognize rooms from action sequences without teaching. SOM was used for learning since it was able to generalize an input vector without a teacher. We also developed BI-transformation and a recognition method with a partial action sequence. The experiments using a real mobile robot were made for rooms both with and without obstacles, and we verified the utility of our approach.

References

[1] J.L. Crowly, Navigation of an intelligent mobile robot, IEEE Trans. Robot. Autom. 1 (1) (1985) 31–41.

[2] A. Elfes, Sonar-based real-world mapping and navigation, IEEE Trans. Robot. Autom. 3 (3) (1987) 249–265.

[3] A. Elfes, Using occupancy grids for mobile robot perception and navigation, IEEE Comput. June (1989) 46–57.

[4] M. Asada, Map building for a mobile robot from sensory data, IEEE Trans. Syst. Man Cybernet. 20 (6) (1990) 1326–1336.

[5] R. Dillmann, F. Wallner, R. Graf, Real-time map refinement by fusing sonar and active stereo-vision, in: Proceedings of the 1995 IEEE International Conference on Robotics and Automation, 1995, pp. 2968–2973.

[6] S. Yamada, M. Murota, Applying self-organizing networks to recognizing rooms with behavior sequences of a mobile robot, in: Proceedings of the IEEE International Conference on Neural Networks, 1996, pp. 1790–1794.

[7] S. Yamada, M. Murota, Unsupervised learning to recognize environments from behavior sequences in a mobile robot, in: Proceedings of the 1998 IEEE International Conference on Robotics and Automation, 1998, pp. 1871–1876.

[8] U. Nehmzow, T. Smithers, Map-building using self-organizing networks in really useful robots, in: Proceedings of the First International Conference on Simulation of Adaptive Behavior, 1991, pp. 152–159.

[9] U. Nehmzow, T. Smithers, Using motor actions for location recognition, in: Proceedings of the First European Conference on Artificial Life, 1991, pp. 96–104.

[10] W.D. Smart, J. Hallam, Location recognition in rats and robots, in: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, MIT Press, 1994, pp. 174–178.

[11] M.J. Mataric, Integration of representation into goal-driven behavior-based robot, IEEE Trans. Robot. Autom. 8 (3) (1992) 14–23.

[12] S. Tsuji, S. Li, Memorizing and representing route scenes, in: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, 1992, pp. 225–232.

[13] K. Kato, Statistical analysis of robot behaviors in unstructured environment, in: Proceedings of Asian Conference on Computer Vision, 1995, pp. 219–223.

[14] T. Nakamura, S. Takamura, M. Asada, Behavior-based map representation for a sonar-based mobile robot by statistical methods, in: Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1996, pp. 276–283.

[15] R.D. Berns, U. Zachmann, Reinforcement learning for the control of an autonomous mobile robot, in: Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992, pp. 1808–1815.

[16] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Trans. Robot. Autom. 2 (1) (1986) 14–23.

[17] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, J.S.B. Mitchell, An efficiently computable metric for comparing polygonal shapes, IEEE Trans. Pattern Anal. Mach. Intell. 13 (3) (1991) 209–216.

[18] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, Berlin, 1989.

[19] S. Yamada, Evolutionary design of behaviors for action-based environment modeling by a mobile robot, in: Proceedings of Genetic and Evolutionary Computation Conference, 2000, pp. 957–964.

[20] D. Ackley, M. Littman, Interactions between learning and evolution, in: Artificial Life II, Addison-Wesley, Reading, MA, 1991, pp. 487–509.