Unsupervised Learning to Recognize Environments from Behavior Sequences in a Mobile Robot

Seiji Yamada Morimichi Murota

CISS, IGSSE
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama 226-0026, JAPAN
yamada@ymd.dis.titech.ac.jp
http://www.ymd.dis.titech.ac.jp/~yamada/

Abstract

In this paper, we describe development of a mobile robot which does unsupervised learning for recognizing environments from behavior sequences. 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 not be obtained in real environments. Furthermore unsupervised-learning is necessary for recognition in unknown environments 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. Then the sequences of behaviors executed in each enclosure are transformed into input vectors for a self-organizing network. Learning without a teacher is done, and the robot becomes able to identify enclosures. Moreover we developed 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.

1 Introduction

In robotics research, most studies on recognizing an environment have tried to build a precise geometric map with high sensitive sensors [3]. However many natural agents like animals recognize their environments only with low sensitive sensors, and a geometric map may not be necessary. In view of engineering, it is important to develop a simple robot which is able to recognize environments only with inexpensive sensors because the cost is low. In many real environments like a dark room, the global information like vision may not be obtained.

Moreover, since a robot should be adaptive to unknown environments without help of a teacher, it needs to learn to recognize new environment 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. The robot is behavior-based and does wall-following in enclosures. Then the sequences of behaviors executed in each enclosure are obtained. The sequences 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. Since we carefully define transformation from a behavior sequence into an input vector and the self-organizing network works well for generalizing data, 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 sequence.

We have fully implemented the system using a real mobile robot with two 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.

Nehmzow and Smithers studied on recognizing corners in simple enclosures with a self-organizing network [6]. They used direction-duration pairs, which indicate the length of walls and shapes (convex or concave) of past corners, as an input vector to a self-organizing network. After learning, the network be-

comes able to identify corners. However the transformation from raw data to an input vectors is significantly sensitive to noise like small obstacles. We proposes another transformation which maintains better topology than their one. Furthermore we experimentally evaluate robustness of our method against obstacles.

Mataric represented an environment using automata consisting landmarks as nodes [5]. 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 her approach is similar to ours, it is not suitable for our purpose.

Tsuji and Li have done the excellent study on vision-based memorizing route scenes [7]. The mobile robot stores qualitative panoramic representation, and locates itself in the route by matching the memorized representation against that of the incoming scenes. If a robot has a vision system, their approach will be valid.

2 Overview

The overview of a whole system is shown in Fig.1. First a behavior-based mobile robot [2] goes round once in each of n enclosure by wall-following, and obtains n sequences of executed behaviors. Next the sequences (symbol lists) is transformed into the real-valued vectors, and they are given to a self-organizing network as input. Note that there is no teacher giving the correspondence from input vectors to enclosures. Learning is done on the network using the Kohonen's update rules. The n input vectors are repeatedly given to the network, and learning progresses. After learning, a winner node indicates one of the enclosures in which the robot did wall-following. The test data are given to the learned network, and the mobile robot identifies the test data with enclosures already trained. We mean this identification by *environment recognition*.

3 Wall-following by a behavior-based mobile robot

A mobile robot used in our research is shown in Fig.2(a). It has four infrared proximity sensors: two in front direction and other two in left and right direction on the both sides (top view in Fig.2(b)). Only two

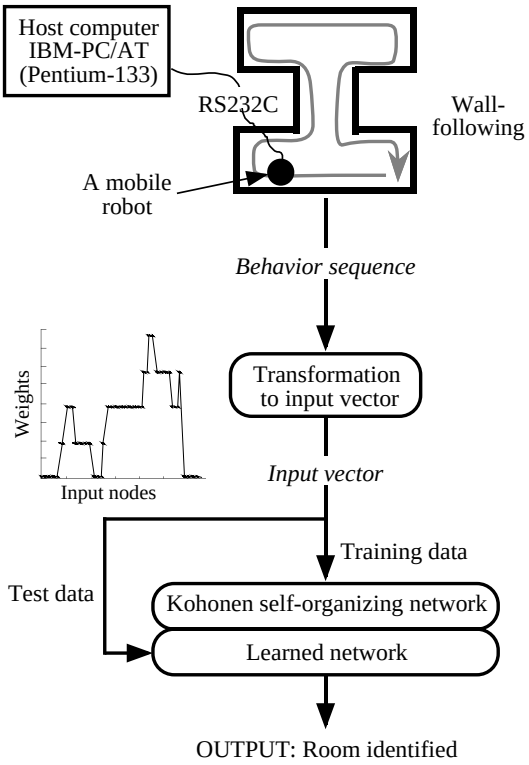


Figure 1 System overview

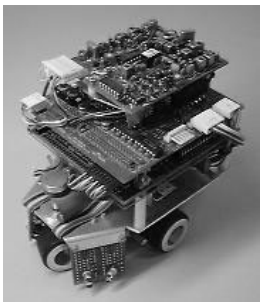
sensors on the left side are actually used since the wall-following is done clockwise. The mobile robot also has an orientation sensor for steering and an encoder for movement distance. Note that the infrared proximity sensors are low sensitive and obtain just local information within 20 cm. The actuators consists of two stepping motors for driving left and right wheels independently and a DC motor for a front steering wheel (Fig.2(b)).

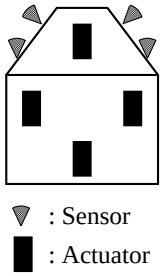
Since the mobile robot needs to do wall-following even in an enclosure where some obstacles exist, we use the behavior-based approach [2] which is known robust against the change of an environment. The behavior-based approach also can controls a robot with low sensitive sensors, and have the advantage that the behavior sequence is invariant even when the geometric movement history of the mobile robot varies a little.

We use four behaviors (reactive rules) in the following for wall-following. The behaviors are periodically executed, and the mobile robot constantly goes forward.

Behavior-A (*turning in a concave corner*): *If* an obstacle within 10cm in the front and within 10cm on the left *then* turning 40° clockwise there.

Behavior-B (*turning in a convex corner*): *If* no





(a) (b)

Figure 2 A mobile robot

obstacle within 5cm on the left and the right, and within 10cm in the front *then* turning 40° counterclockwise there.

Behavior-C (*following-1*): *If* an obstacle within 5cm on the left *then* steering 13.5° clockwise.

Behavior-D (*following-2*): *If* no obstacle within 5cm on the left *then* steering 13.5° counterclockwise.

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

4 Transformation from a behavior sequence into an input vector

By wall-following, a sequence of executed behaviors is obtained. It has information on the shape of the enclosure: the length of a continuous sequence of behavior C and D indicates the length of a wall, and behavior A and B indicate the existence of a concave corner and a convex corner respectively. Thus we consider a mobile robot can identify enclosures with the sequences.

Since a robot should be adaptive to unknown environments in which no other agents help it, it needs unsupervised-learning. Hence we introduce a Kohonen's self-organizing network for identifying enclosures. The behavior sequence is a list of symbols, thus we need to transform it into a real-valued vector as input to a self-organizing network. The transformation also needs to be robust against noise and easily computed.

We propose *BI-transformation*. Given a behavior sequence: $[r_1, r_2, \cdots, r_n]$ $(r_i \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\})$ and an input vector: $\boldsymbol{I} = (v_1, v_2, \cdots, v_m)(n \leq m)$, the values of $\boldsymbol{I}$ are obtained using the following rules, where $v_1 = 0$.

1. If $r_i = \mathsf{A}$ then $v_i = v_{i-1} + 1$.
2. If $r_i = \mathsf{B}$ then $v_i = v_{i-1} - 1$.
3. If $r_i = \mathsf{C}$ or D then $v_i = v_{i-1}$.
4. Otherwise $v_i = 0$ $(i > n)$.

For example, Fig.5(a) shows an input vector transformed from a behavior sequence in a square enclosure. The x and y axis indicate dimensions and values of an input vector. For an input vector, the number of dimensions indicates the periphery length of an enclosure, a series of the same value indicates a wall, and the change of values stands for a corner. The increase and the decrease indicate a concave corner and a convex corner.

We explain the robustness of BI-transformation using an example shown in Fig.3. The movement history of a mobile robot and an input vector from the behavior sequences are indicated in Fig.3. Fig.3(a) stands for an enclosure without obstacle and Fig.3(b) stands for the same enclosure 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 input vectors may be small. Using the BI-transformation results in the input vectors as Fig.3. The size of shaded areas in Fig.3(b) indicates the difference, and it is relatively small. Note that the whole pattern of the input vector is not shifted. If a robot uses more rigid transformation in which an input vector is described with the length of walls, shapes (convex or concave) of past corners [6], the whole pattern will be shifted and the distance will be significantly large. Hence we consider BI-transformation is robust, and the effectiveness will be experimentally verified.

The idea of BI-transformation is similar to the *turning function* (or the chain coding)[1] in pattern recognition. In general, the turning function is extracted from the raw data like a bit image, while the input vector is directly obtained from a behavior sequence using BI-transformation.

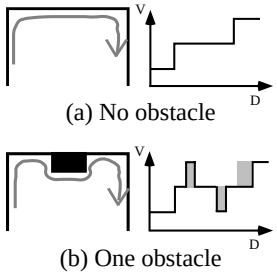


Figure 3 Robustness

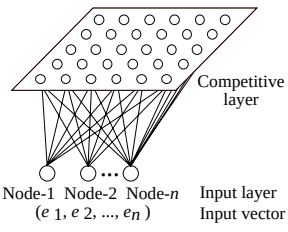


Figure 4: Self-organizing network

5 Learning by a self-organizing network

In this section, we briefly explain a Kohonen's self-organizing network [4]. It clusters large dimensional input vectors by mapping them to small discrete vectors, and is used widely in pattern recognition and robotics. A self-organizing network is a two-layered network consists of an input layer and a competitive layer (Fig.4). 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's neighbor nodes are updated.

Let an input vector and weights of links from all input nodes to a competitive node u_i be $\boldsymbol{E} = [e_1, e_2, \cdots, e_n]$ and $\boldsymbol{U}_i = [u_{i1}, u_{i2}, \cdots, u_{in}]$ respectively. First a self-organizing network computes the Euclidean distance between the input vector and competitive nodes.

After a *winner node* with the minimum distance is determined, the weights of winner's neighbor nodes are updated using the following formula.

$$u_{ij}^{\text{new}} = u_{ij}^{\text{old}} + \Delta u_{ij} \quad \Delta u_{ij} = \begin{cases} \alpha(e_j - u_{ij}) : & i \text{ is neighbor} \\ 0 : & \text{otherwise} \end{cases}$$

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

Next the input vector is given as test data and the winner indicates the class including the input vector. The clustering is automatically done without a teacher.

In our research, the dimension of an input vector is set larger than the length of any behavior sequence. The competitive layer is one-dimensional because the computational complexity of learning with one-dimensional competitive layer is far less than that with a two-dimensional one. If we set sufficiently many nodes in an one-dimensional competitive layer, the resolution of classification will not worse than that of a two-dimensional one.

6 Experiments

Using the mobile robot mentioned earlier, we made experiments for evaluating the utility of our approach.

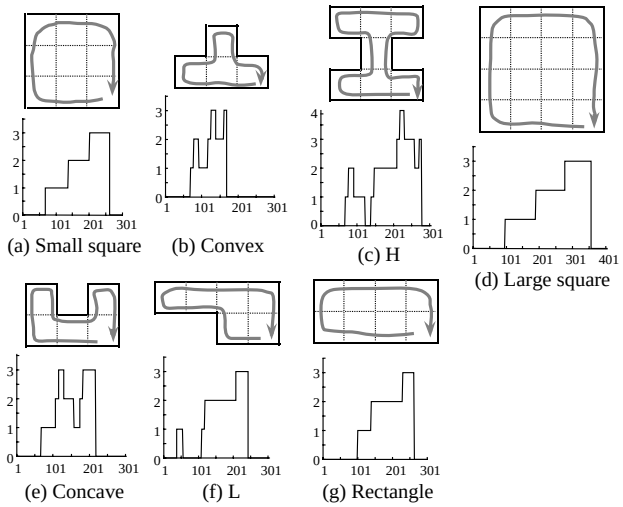


Figure 5 Seven enclosures

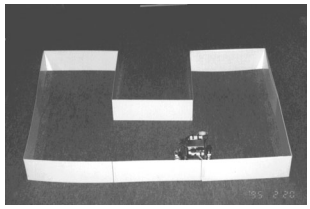


Figure 6 Experimental environments

As seeing from Fig.1, the system consists of a mobile robot and an IBM-PC/AT (Pentium 133MHz, 128M RAM) compatible personal computer 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 behavior sequences were sent to the host computer and the learning by a self-organizing network was done there.

6.1 Environment and a learning phase

Using white plastic boards, we built seven different enclosures in shape. Fig.5 shows the shapes of the enclosures and the input vectors obtained by wall-following in each enclosure. A mobile robot did wall-following ten times for each enclosure, and 70 behavior sequences were obtained in total. For each enclosure, a single behavior sequence was used as an input vector for learning, and other 6 sequences were used for testing. Fig.6 shows the scene that a mobile robot is doing wall-following.

The robot stops when it returns near a start point.

It can recognize the start location by dead reckoning using an orientation sensor and an encoder. A mobile robot, however, starts wall-following at the same position for the same enclosure. This restriction will be removed in §6.4.

The largest length of behavior sequences obtained from the enclosures was about 1400, and this was the minimum dimension of an input vector. Since the length was too large for a self-organizing network to learn tractably, all sequences were compressed into 1/4 and the length was 500 at most.

We constructed a self-organizing network consisting of 520 input nodes and 32 competitive nodes located in one dimension. The neighborhood in the competitive layer is defined with d nodes on both sides of the winner. The initial value d_0 and α_0 were set 5 and 0.2. We changed α and d as learning progresses. The initial weights of competitive nodes were set randomly within 1.5 ± 0.15.

All of 70 behavior sequences were transformed into input vectors using BI-transformation. The seven training data (Fig.5) consisting of a single input vector for each of seven enclosures were randomly given to the self-organizing network until the total number becomes 4200, and learning was done.

When the learning began to converge, the particular nodes got to be winners frequently. We considered the winner nodes correspond to enclosures, and called them *r-node*s. Hence the number of r-nodes is the number of enclosures recognized by a robot.

In the test phase, at every time a test input vector was given, a winner of r-nodes was determined. We considered the winner r-node corresponds to an enclosure in which the input vector was obtained.

6.2 Exp-1: Identifying the enclosures

After learning with training data, the test data (63 input vectors) were given to the learned self-organizing network. As a result, we found all the test data were correctly identified (the accuracy = 100%), and verified the utility of our approach. Note that the 10 input vectors for the identical enclosure are slightly different mutually because of noise like failure of executing behaviors.

Furthermore additional experiments were made using twelve enclosures in Fig.7. 120 action sequences (20 for each enclosure) were used for training and test examples. The network consisted of 1300 input nodes and 128 competitive nodes. As a result, the accuracy decreased to 76%.

Using the twelve enclosures, we made experiments for investigating the influence of self-organizing network's parameters on the accuracy. Setting the initial weight values: $0.5 \pm 10\%$, $1.5 \pm 10\%$, $4.5 \pm 10\%$ and the structure of competitive layer: a line and a circle. As a result, the accuracy was $69\% \sim 74\%$, and the influence was considered slight.

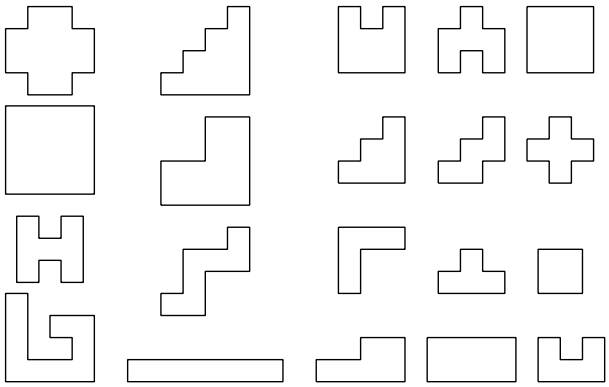


Figure 7 Twelve Enclosures

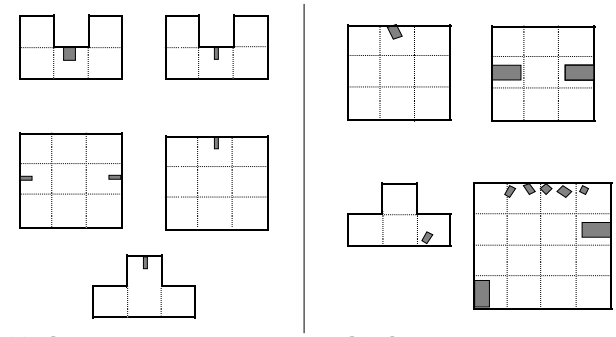
6.3 Exp-2: Noisy environments

Though the test data used in Exp-1 included noise, it was not so much. In this experiment, we dealt with more noise like obstacles. We located obstacles in the seven enclosures, and the mobile robot did wall-following in the enclosures. The nine behavior sequences were obtained and transformed into input vectors. The input vectors (test data) were given to the self-organizing network which was trained in Exp-1.

As a result, five enclosures were correctly identified. Fig.8 shows success and failure examples. In Fig.8(a), the robot correctly recognized the enclosure including either a single obstacle or two small cubical obstacles on a wall. However, in Fig.8(b) the robot failed to recognize an enclosure including either two large obstacles or an obstacle located obliquely to the wall. The large or many obstacles changed the shape of enclosures from the original one, thus we consider the failure of recognition is natural.

6.4 Exp-3: Utilizing partial sequences

Although the experimental results were satisfactory, there are two important problems: a start point must be fixed in every trial, and a complete behavior sequence is necessary for recognition. The former makes recognition less robust, and the latter causes expensive recognition. Thus we developed a method to recognize environments with partial behavior se-



(a) The correct recognition (b) The wrong recognition

Figure 8 Enclosures with obstacles

quences independently of a start point. The following shows the procedure.

Training phase : Sift a behavior sequence so that the longest wall may be the head, and learn with it. After learning, r-nodes are obtained.

Testing phase : At every corner, the following is done. Make an input vector by BI-transformation, and obtain the partial input vector by shifting it so that the longest wall may be the head and setting value zero when a behavior was not executed yet. Compute the distance between the partial input vector and r-nodes. If the distance is sufficiently short, the environment is determined the r-node's environment.

Sifting an action sequence makes the recognition independent of a start point. Since the execution of a partial action sequence is more inexpensive than that of a complete one, the above procedure makes the recognition far more efficient.

We made experiments with the twelve enclosures in Exp-1. As a result, the accuracy and the partial ratio was 56.7% and 47.8%, where the partial ratio = $\frac{\text{The length of partial behavior sequences}}{\text{The length of complete behavior sequences}}$ (%).

7 Conclusion

We built a mobile robot which learns to recognize enclosures from behavior sequences without teaching. A self-organizing network was used for learning since it was able to generalize an input vector without a teacher and robust against noise. We also developed BI-transformation and a recognition method with a partial behavior sequence. The experiments using a real mobile robot were made for enclosures both with and without obstacles, and we verified the utility of our approach. Though this report may be preliminary, our approach is considered to contribute for designing a mobile robot which autonomously recognizes environments. While we obtained good experimental results, there are open problems like the followings.

- *Open environments*: Though we assume that an environment is a closed region, real environments 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.
- *Restriction on the shapes of enclosures*: The shapes of enclosures are restricted to rectangles with right corners. If an enclosure consists of curves like a circle, neither the behavior A nor B will be executed and a robot will not use information of corners. We need to improve the behaviors and BI-transformation.
- *Suitable behaviors for recognizing environments*: We used wall-following as behaviors for recognizing enclosures, 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 [8].

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 PAMI*, 13(3):209–216, 1991.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Transaction on R&A*, 2(1):14–23, 1986.

[3] J. L. Crowly. Navigation of an intelligent mobile robot. *IEEE Transaction on R&A*, 1(1):31–41, 1985.

[4] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, 1989.

[5] M. J. Mataric. Integration of representation into goal-driven behavior-based robot. *IEEE Transaction on R&A*, 8(3):14–23, 1992.

[6] 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.

[7] S. Tsuji and S. Li. Memorizing and representing route scenes. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 225–232, 1992.

[8] S. Yamada. Learning behaviors for environment modeling by genetic algorithm. In *The First European Workshop on Evolutionary Robotics*, 1998. to appear.