Applying Self-Organizing Networks to Recognizing Rooms with Behavior Sequences of a Mobile Robot

Seiji Yamada and Morimichi Murota

Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuda, Midori-ku, Yokohama, Kanagawa 226, Japan
{yamada, murota}@ai.sanken.osaka-u.ac.jp

ABSTRACT

In this paper, we describe the application of a self-organizing network to the robot which learns to recognize rooms (enclosures) using behavior sequences. In robotics research, most studies on recognizing environments have tried to build the precise geometric map with high sensitive sensors. However many natural agents like animals recognize the environments with low sensitive sensors, and a geometric map may not be necessary. Thus we attempt to build a mobile robot using a self-organizing network to recognize the enclosures, in which it acts, with low sensitive and local sensors. The mobile robot is behavior-based and does wall-following in an enclosure. Then the sequences of behaviors executed in each enclosure are obtained. The sequences are transformed into real-value vectors, and inputted to the Kohonen's self-organizing network. Unsupervised-learning is done and a mobile robot becomes able to distinguish and identify enclosures. We fully implemented the system using a real mobile robot and made experiments for evaluating the ability. Consequently we found out the recognition of enclosures was done well and our method was robust against small obstacles in an enclosure.

1. Introduction

In robotics research, most studies on recognizing an environment have been done for building the precise geometric map with high sensitive sensors [4]. However, since many natural agents like animals recognize their environments with low sensitive sensors, a geometric map and high sensitive sensors may not be necessary for recognizing environments. Thus we apply a self-organizing network to a mobile robot recognizing the rooms with low sensitive sensors. In this paper, the *rooms* mean the *enclosures* in which a robot acts. The robot does wall-following in rooms, and behavior sequences are obtained. The sequences are transformed into real-value vectors, and inputted to a self-organizing network. Learning without a teacher is done and the robot becomes able to identify rooms. Since we carefully defined the transformation from a behavior sequence into an input vector and a self-organizing network worked well for generalizing data, the learned network is robust against the noisy data from the rooms including small obstacles. We fully implemented the system with a real mobile robot, and made experiments. As a result, we found out the recognition of rooms was done well and our method was robust against the noisy data.

Nehmzow et al. studied on recognizing corners in rooms with a self-organizing network [6]. They used a sequence of the length of walls, shapes (convex or concave) of corners as an input vector to the network, and it learned to identify the corners. However the transformation is considered significantly sensitive to noise like obstacles. In § 4, we proposes more robust representation than Nehmzow's one.

In robotics, a self-organizing network has been used just to generalize raw sensor data [2]. In contrast with such studies, in our research, the output of a self-organizing network directly indicates a particular room. Thus we need to carefully define the transformation from a behavior sequence into an input vector so that a network may cluster input data into correct classes corresponding to rooms. Defining the transformation is an important problem which does not occur in straightforward application, and we propose a solution.

2. Overview

The overview of a whole system is shown in Fig. 1. First a behavior-based mobile robot [3] goes round once in each of n rooms by wall-following, and obtains n sequences of executed behaviors. Next the sequences (symbol lists) are transformed into the real-valued vectors, and the vectors are given to a self-organizing network as input. Note that there is no teacher explicitly giving the correspondence from input vectors to rooms.

The n input vectors are repeatedly given to the self-organizing network, and learning progresses. After learning, a winner node of a self-organizing network indicates a particular one of the rooms explored by the mobile robot. The test data are given to the learned network, and the mobile robot is able to identify them to rooms. We mean this identification by *room recognition*.

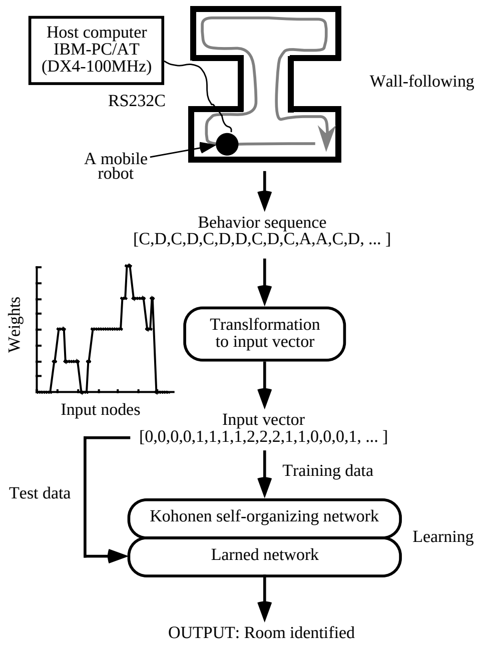

Fig. 1 System overview

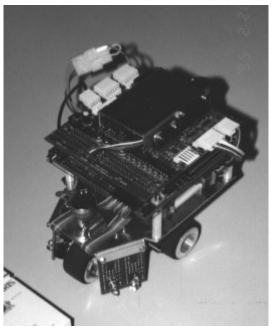

Fig. 2 A mobile robot

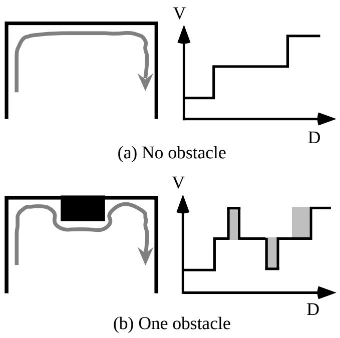

Fig. 3 Influence of an obstacle

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

The mobile robot (Fig. 2) has only two low-sensitive infrared proximity sensors: one in front direction and the other in left direction on left side. It also has an orientation sensor and an encoder. Since the mobile robot needs to do wall-following even in a room in which obstacles exist, we use the behavior-based approach [3] which is robust against the change of an environment. We used four behaviors in the followings. As executing the behaviors, the mobile robot always goes forward, and does wall-following clockwise. The walls are considered as obstacles. We experimentally verified that the mobile robot did wall-following well using the four behaviors.

Behavior A (*turning in the concave corner*): *if* any obstacle within 10 cm in the front and within 10 cm in the left *then* turning 40° clockwise.

Behavior B (*turning in the convex corner*): *if* no obstacle within 5 cm in the left and the right, and within 10 cm in the front *then* turning 40° counterclockwise.

Behavior C (*wall-following-1*): *if* any obstacle within 5 cm in the left *then* steering 13.5° right.

Behavior D (*wall-following-2*): *if* no obstacle within 5 cm in the left *then* steering 13.5° left.

4. Transformation from a behavior sequence into an input vector

By wall-following, a sequence of executed behaviors is obtained. The sequence has information on the shape of the room: the length of a sequence of behavior C and D indicates the length of a wall, and behavior A and B indicates a concave corner and a convex corner respectively. Thus we consider a mobile robot is able to identify rooms with the sequences.

Since the ability of generalization and the robustness against noise, we introduce the Kohonen's self-organizing network for identifying rooms. The behavior sequence is a list of symbols, thus we need to transform it into a real-valued vector as an input to the self-organizing network. The transformation should be also robust and easily computed.

We use the transformation, called *BI-transformation*, like turning function [1]. The dimension of an input vector is m which is the larger number than the length of the behavior sequence, n. Given a behavior sequence $[r_1, r_2, \ldots, r_n]$ ($r_i \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$) and an input vector $\boldsymbol{I} = (0, v_2, \ldots, v_m)$, the values of $\boldsymbol{I}$ are determined using the following rules.

BI-transformation

(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 the behavior sequence in a square room. The x and y axis indicate the dimensions and values of the input vector. For an input vector, the dimension of an input vector indicates the periphery length of a room, the continuous line of the same vector value indicates a wall, and the change of values stands for a corner.

We explain the robustness of BI-transformation using an example shown in Fig. 3. The moving histories of a mobile robot and the input vector transformed from the behavior sequences by BI-transformation are indicated in Fig. 3. Fig. 3(a) stands for the case of a room without obstacle and Fig. 3(b) stands for the same room including a obstacle (a black rectangle). Since the two rooms are identical, the mobile robot should identify them as the same one. Because a self-organizing network classifies input vectors with the Euclidean distance, the transformation needs to be defined so that the Euclidean distance between the two input vectors may be small. Using the BI-transformation results in the input vectors in the graphs of Fig. 3. The size of shaded areas in Fig. 3(b) indicates the distance between the two input vectors, and it is relatively small. Note that only a part of the input vector is shifted by an obstacle. If the robot uses more rigid transformation in which the input vector is described with the length of walls, shapes (convex or concave) of corners used in [6], all the input vector values will be shifted and the distance will become significantly large. Hence we consider BI-transformation is robust, and the effectiveness will be experimentally verified in § 6.

5. Learning by a self-organizing network

In this section, we briefly explain the Kohonen's self-organizing network [5]. The self-organizing network 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 (Fig. 4) is a two-layered network consists of an input layer and a competitive layer. Any input node is linked to all competitive nodes, and all links have weights. As an input vector is given, the input nodes have values corresponding to the input vector, and the competitive nodes has values which stand for the distance between their weights and the input vector. The *winner node* which has the minimum distance is determined, and the 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 $\boldsymbol{E}$ and $\boldsymbol{U}_i$ respectively.

$$\boldsymbol{E} = [e_1, e_2, \cdots, e_n] \qquad \boldsymbol{U}_i = [u_{i1}, u_{i2}, \cdots, u_{in}]$$

First a self-organizing network computes the distance between the input vector and competitive nodes.

The distance is computed with the following equation. Next the winner is determined.

$$\|\boldsymbol{E} - \boldsymbol{U}_i\| = \sqrt{\sum_j (e_j - u_{ij})^2}$$

After a winner is determined, the weights of winner'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 as the center is often used as neighborhood. The equation for updating the weights of neighbors is shown in the following, where the α is a learning rate.

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij} \qquad \Delta w_{ij} = \begin{cases} \alpha(e_j - w_{ij}) & : \text{node } i \text{ is within neighborhood} \\ 0 & : \text{otherwise} \end{cases}$$

The learning is finished when no update is done. Next the input vector is given as test data and the winner indicates the cluster including the input vector. The clustering is automatically done without a explicit teacher.

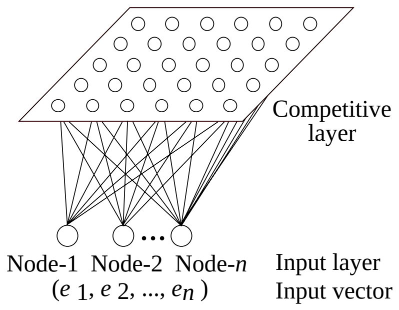

Fig. 4 A self-organizing network

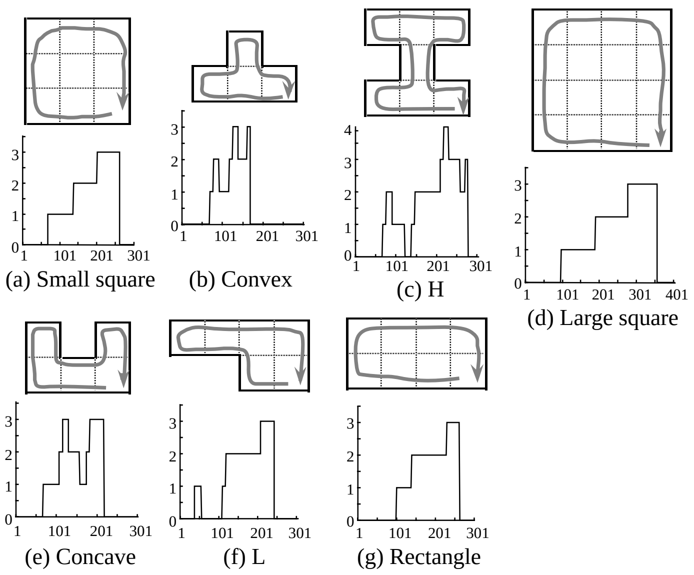

Fig. 5 Rooms

6. Experiments with a real mobile robot

Using the mobile robot mentioned earlier, we made experiments for evaluating the utility of our approach. As seeing from Fig. 1, the system consists of a mobile robot and an IBM-PC/AT compatible personal computer (i486-DX4 100MHz) 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 the robot autonomously follows 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 learning

We made experiments for evaluating the utility of our approach. Using white plastic boards, we built seven rooms. Fig. 5 shows the shapes and input vectors of the rooms. A mobile robot did wall-following ten times for each room, and 70 behavior sequences were obtained in total. For each room, a single behavior sequence was used as an input vector for learning, and other 9 sequences were used for testing. The largest length of the behavior sequences was about 1400, and all sequences were compressed into 1/4 for fast computation.

We constructed a self-organizing network consisting of 520 input nodes and 32 competitive nodes located in one dimension. Thus the neighborhood in the competitive layer is defined with d nodes on both sides of the winner. We decreased the learning rate α and d linearly as learning iteration progresses. The initial weights of links were set randomly over 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 rooms were randomly given to a self-organizing network until the total number becomes 4200. When the learning began to converge, the particular nodes got to be winners frequently. We considered the winner nodes correspond to rooms, and called them *r-node*s. Hence the number of r-nodes is the number of rooms recognized by a robot.

In the test phase, at every time a test input vector was given, a winner node was determined with the distance between the input vector and competitive nodes' weights. We considered the nearest r-node to the winner node in the competitive layer indicated a room in which the input vector was obtained.

6.2. Experiments

Exp-1: (identifying the rooms) 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, and verified the utility of our approach. Note that the 10 input vectors for the identical room were somewhat different mutually because the wall-following included noise like failure of executing behaviors.

Exp-2: (the rooms with obstacles) 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 some obstacles in the rooms, and the mobile robot did wall-following in the rooms. 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 data were correctly identified. The robot failed to recognize a room in which several obstacles scattered. However the obstacles made the room the different shape from the original one, thus we consider the failure of identification is natural.

7. Conclusion

We described application of a self-organizing network to recognizing rooms with behavior sequences of a mobile robot. A self-organizing network is used since it is able to generalize the input vector without a teacher and robust against noise. The experiments using a real mobile robot were made for rooms both with and without obstacles, and we verified the utility of our approach. We currently develop a method to identify rooms with a part of wall-following.

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*, vol. 13, no. 3, pp. 209–216, 1991.

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

[3] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Transaction on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[4] J. L. Crowly, "Navigation of an intelligent mobile robot," *IEEE Transaction on Robotics and Automation*, vol. 1, no. 1, pp. 31–41, 1985.

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

[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*, pp. 152–159, 1991.