Neural Computation & Application manuscript No.
(will be inserted by the editor)

Task Segmentation in a Mobile Robot by mnSOM : A New Approach To Training Expert Modules

M. Aziz Muslim[1], **Masumi Ishikawa**[1], **Tetsuo Furukawa**[1]

Department of Brain Science and Engineering,
Kyushu Institute of Technology,
2-4 Hibikino, Wakamatsu, Kitakyushu
808-0196, Japan

Received: date / Revised version: date

Abstract Proposed is a new approach to task segmentation in a mobile robot by a modular network SOM (mnSOM). In a mobile robot, however, the standard mnSOM is not applicable as it is, because it is based on the assumption that class labels are known a priori. In a mobile robot, only a sequence of data without segmentation is available. Hence, we propose to decompose it into many subsequences, supposing that a class label does not change within a subsequence. Accordingly, training of mnSOM is done for each subsequence in contrast to that for each class in the standard mnSOM. The resulting mnSOM demonstrates good segmentation performance of 94.05% for a novel dataset.

Keywords: task segmentation, mobile robot, subsequence, modular network SOM

1 Introduction

Task segmentation in navigation of a mobile robot has attracted wide attention. Tani et al. proposed to generate a series of actions based on sensory-motor signals using a forward model represented by a recurrent neural network [6]. Tani and Nolfi [7] proposed 2-level hierarchical mixture of recurrent experts (MRE), which is an extension of the network architecture proposed by Jacobs et al. [3]. Tani et al. also proposed 2-level prediction networks for extracting spatio-temporal regularities [5]. Wolpert and Kawato [13] proposed MOSAIC architecture for motor control with the soft-max function for assigning responsibility signal to each module.

In the conventional competitive learning, modules or units are in isolation; there exists no notion of similarity between them. Because of this, interpolation among modules or units is not applicable as it is. We think that there are two aspects in "interpolation". The one is creating an output which is interpolated by outputs from multiple modules. The other is creating a module which is an interpolation of multiple modules. Let the former be called output interpolation and the latter be called module interpolation. The present study focuses on the module interpolation.

The conventional SOM can be said to possess unit interpolation. The soft-max [13] is an improvement over the conventional competitive learning in that the output interpolation is possible based on the responsibility signal produced by the soft-max function. Similarity between modules, however, is not explicitly represented. Furthermore, the soft-max function and segmentation generally do not coexist; only when the soft-max function becomes winner-take-all, segmentation is possible at the sacrifice of interpolation.

Tani et al. recently proposed a recurrent neural network with a parametric bias [8]. It has the ability of the output interpolation, but has no longer the capability of segmentation.

Self Organizing Maps (SOM)[9] is a well known method for classifying data while preserving topological relationship. The resulting topological maps represent unit interpolation among classes in dataset. Martinez [10] proposed Neural Gas (NG) to alleviate a difficulty in SOM: mapping of high dimensional input space onto fixed lattice. Walter et al. did a comparative study of SOM and NG in robotic applications [11].

In contrast to SOM and NG using a vector as its element, we propose to use a modular network SOM (mnSOM) [1][2][4] for task segmentation in navigation of a mobile robot. mnSOM is an extension of SOM in that function modules instead of vector units are used to increase its representation and learning capability. Owing to competitive learning among function modules, mnSOM is capable of segmentation. Owing to topographic mapping of function modules on a plane, the neighboring function modules tend to have similar characteristics. Hence, interpolation among function modules becomes possible. The simultaneous realization of segmentation and interpolation is unique and unparalleled characteristics of mnSOM.

In case of a mobile robot, however, the standard mnSOM is not applicable as it is, because it is based on the assumption that class labels are known a priori. In a mobile robot, however, only a sequence of data without segmentation is available. Hence, we propose to decompose it into many subsequences, supposing that a class label does not change within a subsequence. Accordingly, training of mnSOM is done for each subsequence in contrast to that for each class in the standard mnSOM.

We already proposed a similar method taking temporal continuity of winner modules into account [14]. It turns out, however, that the present proposal is simpler than and superior to the previous one. Detailed comparison can be found in Section 4.

Section 2 briefly explains an mnSOM algorithm. Section 3 describes a proposed method for task segmentation in a mobile robot. Section 4 gives experimental results for training data and for test data. Section 5 provides conclusions and discussions.

2 The mnSOM

2.1 Architecture

mnSOM is an extension of SOM in that each vector unit is replaced by a function module such as a feedforward neural network or a recurrent neural network. An important issue here is to choose appropriate function modules and similarity measure between modules[4]. To deal with dynamical systems, recurrent neural networks (RNN) are suitable for function modules. In this case, mnSOM learns nonlinear dynamics of a given input and output sequences, and forms a topological map composed of modules [1].

Fig.1 illustrates the architecture of mnSOM and the function module (i.e. recurrent neural network) as its element. Each vector unit in the conventional SOM is replaced by a fully connected recurrent neural network. A weight vector of a recurrent neural network may be regarded as a feature vector. However, the distance between two modules in mnSOM is not measured by the Euclidian distance between two corresponding weight vectors, but is measured by the difference between output sequences of two corresponding modules, given an input subsequence. As in the conventional SOM, the closer a module is to the best matching one, the more it learns during a learning process. Each module is trained by backpropagation through time (BPTT) [12]. Details of mnSOM learning algorithm follows.

2.2 Learning Algorithm

As mentioned in Introduction, training of and competition among modules in mnSOM is done for each subsequence instead of each class, supposing that a class label

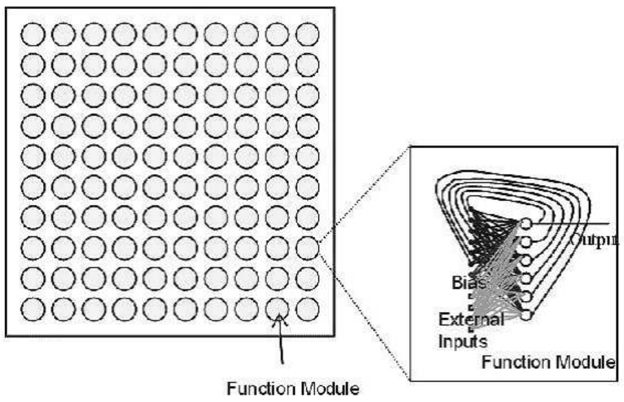

Fig. 1 Array of modules in mnSOM and the function module as its element. The function module is a fully connected RNN.

does not change within a subsequence. A learning algorithm of mnSOM is conceptuallysimilar to that of the batch learning SOM. It consists of 4 processes [2]: evaluative, competitive, cooperative and adaptive processes. Let a set of input-output signals of a dynamical system be $\{x_{ij}, y_{ij}\}(i = 1, ..., M; j = 1, ..., L)$, where M and L are the number of subsequences and the number of data in each subsequence, respectively.

1) Evaluative process.

Inputs, $\{x_{ij}\}$, are given to all modules, and the corresponding outputs, $\{\tilde{y}_{ij}^{(k)}\}$, are evaluated by,

$$E_i^{(k)} = \frac{1}{L}\sum_{j=1}^{L} \|\tilde{y}_{ij}^{(k)} - y_{ij}\|^2$$

$$k = 1, ..., K; i = 1, ..., M; j = 1, ..., L \quad (1)$$

where k stands for the module number, K stands for the number of modules, i stands for the subsequence number, and j stands for the data number in each subsequence.

2) Competitive process

The module with the minimum $E_i^{(k)}$ with respect to k is the winner for subsequence i.

$$v_i^* = \arg_k \min E_i^{(k)} \quad (2)$$

3) Cooperative process

A Learning rate of the module is determined by the following normalized neighborhood function:

$$\psi_i^{(k)}(t) = \frac{\phi(r(k, v_i^*); t)}{\sum_{i=1}^{M} \phi(r(k, v_i^*); t)} \quad (3)$$

$$\phi(r; t) = \exp[-\frac{r^2}{2\sigma^2(t)}] \quad (4)$$

$$\sigma(t) = \sigma_{min} + (\sigma_{max} - \sigma_{min})e^{-\frac{t}{\tau}} \quad (5)$$

where $r(k, v_i^*)$ stands for the distance between module k and the winner module, v_i^*, t is the iteration number in

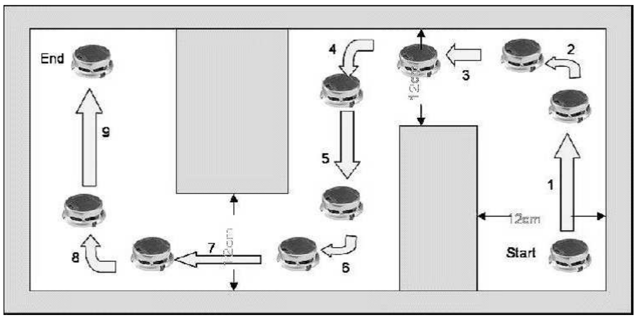

Fig. 2 Mobile Robot Field.

mnSOM, σ_{min} is the minimum neighborhood size, σ_{max} is the maximum neighborhood size, and τ is a decay time constant of a neighborhood size.

4) Adaptive process
Connection weights of module k, $\mathbf{w}^{(k)}$, are modified by the following BPTT algorithm,

$$\Delta\mathbf{w}^{(k)} = \sum_{i=1}^{M} \psi_i^{(k)}(t)(-\eta\frac{\partial E_i^{(k)}}{\partial\mathbf{w}^{(k)}}) \qquad (6)$$

At each mnSOM iteration, we repeat this BPTT algorithm for sufficient number of times.
These 4 processes are repeated and terminate when no significant change is observed in the connection weights and the resulting map.

3 Task Segmentation

Experiments are carried out using a Khepera II robot. It has 8 infra-red (IR) proximity sensors and 2 separately controlled DC motors. The sensors can detect an obstacle within 5 cm. The robot is controlled by a PC via serial connection.

3.1 Robot Field and Division of Data

The robot moves from the start position to the end position by wall following on the robot field in Fig.2. Let the whole movement from the start position to the end position be called a path. During a path, robot turns left twice and turns right twice. When the robot moves in the reverse direction, it experiences similar movements. For later evaluation of training and test results, the path in Fig.2 is manually segmented into 9 sequences based on motor commands as in Fig.2. Sequences, 1, 3, 5, 7 and 9, correspond to a class of forward movement, sequences, 2 and 4, correspond to a class of left turn, and sequences, 6 and 8, correspond to a class of right turn.

The path in Fig.2, comprising 843 samples, is divided into shorter "subsequences" with uniform length. In case of a subsequence with the length 20, the path is split into 42 subsequences with the last 3 subsequences being the length of 21. Table 1 shows the division, where

Table 1 Division of the path into subsequences with the length 20

Sequence	Data Numbers	Subsequence Numbers	Labels
1	1-131	1,2,3,4,5, 6,7	F,F,F,F,F, F,L/F
2	132-187	7,8,9,10	L/F,L,L,L/F
3	188-312	10,11,12,13, 14,15,16	L/F,F,F,F,F, F,L/F
4	313-368	16,17,18,19	L/F,L,L,L/F
5	369-496	19,20,21,22, 23,24,25	L/F,F,F,F,F, F,R/F
6	497-549	25,26,27,28	R/F,R,R,R/F
7	550-666	28,29,30,31, 32,33,34	R/F,F,F,F,F, F,R/F
8	667-719	34,35,36	R/F,R,R/F
9	720-843	36,37,38,39, 40,41,42	R/F,F,F,F,F, F,F

labels "F", "L", "R", "L/F", and "R/F" stand for forward movement, left turn, right turn, the transition between forward movement and left turn, and the transition between forward movement and right turn, respectively. Because of the regular division of the path in Table 1, several subsequences stretch over two consecutive sequences (i.e., a forward movement sequence and a left turn sequence). They are called "transition" subsequences, constituting virtual classes.

3.2 Training and Segmentation

Each mnSOM module is a fully connected recurrent neural network (FRNN), and learns an internal model of robot-environment interaction, by minimizing mean prediction errors in sensory-motor signals at the next time step, given the past sensory-motor signals. Table 2 gives parameters in mnSOM and FRNN. External input units in FRNN correspond to 8 IR sensors and 2 motor commands. Target outputs are sensory-motor signals at the next time step. All units in FRNN have sigmoidal activation functions.

After training, the resulting mnSOM provides a label to each module by a procedure in Section 4. Given a training subsequence or a novel one, one of the modules becomes a winner. The corresponding label reveals segmented task for each subsequence.

4 Experiments

For evaluating the performance of classification, we train mnSOM using subsequences with varying length, i.e., 10, 15, 20, 25, and 30. Five icons in Fig.3 corresponding to "F","L","R","L/F", and "R/F", are used to facilitate

Table 2 Parameters in mnSOM

#x stands for the number of x.

mnSOM	
map size	10x10(100 modules)
neighborhood size (Eq.(4))	$\sigma_{max} = 10; \sigma_{min} = 1; \tau = 80$
# mnSOM iterations	400
# BPTT iterations for each mnSOM iteration	30
mnSOM Module : FRNN	
# external input units	10
# visible (output) units	10
# hidden units	27
learning rate	0.02

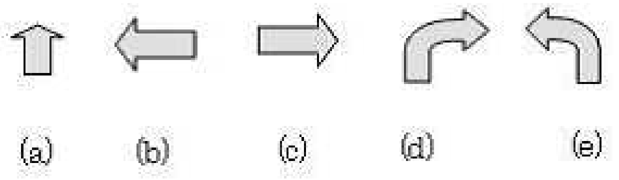

Fig. 3 Icons used in the resulting mnSOM : (a) forward movement (b) left turn (c) right turn (d) transition between forward movement and left turn (e) transition between forward movement and right turn

understanding. A module with one of the icons in Fig.3 indicates that it becomes a winner for a subsequence with the corresponding movement. For later examination, let a module with icon (a) be called a "stationary" module, a module with icon (b) or (c) be called "turning" module, and a module with icon (d) or (e) be called "transition" module.

Fig.4 illustrates the training result using subsequences with the length 20. The numbers in each module shown in the winner modules represent subsequences which become a winner at the corresponding module. White color modules are those which never win the competition for any subsequence. The resulting mnSOM is evaluated by the number of misclassifications.

4.1 Definition of misclassification

Misclassification, here, is defined as the mismatch between the label of a module and that of a subsequence. To evaluate the resulting mnSOM, we define the following degree of badness of misclassification.

1. The degree of badness of misclassifications between "L/F" and "L", and that between "R/F" and "R" are assumed to be 0, because they are between a turning module and a transition module.
2. The degree of badness of misclassifications between "F and "L/F", and that between "F" and "R/F" are assumed to be 0.5, because they are between a stationary module and a transition module.
3. The degree of badness of misclassification between "F" and "L", "F" and "R", "L" and "R", and "R/F" and "L/F" are assumed to be 1, because the difference between them is significantly large.

Table 3 Classification performance for training

Number of samples in each subsequence	Number of misclassifications	Equivalent number of misclassifications	Correct classification rate (%)
10	10	8.5	89.88
15	5	3.5	93.75
20	2	1.0	97.62
25	4	3.0	90.90
30	3	3.0	89.29

4.2 Generating a Task Map Based on a Single Path

We trained mnSOM using subsequences with varying length. Table 3 summarizes the classification performance for subsequences with varying length. It shows that the length of 20 is the best in terms of the correct classification rate. Hereafter, only the results with the length of 20 are shown.

Generally speaking, the shorter a subsequence, the less sufficient the training of a module becomes. On the other hand, the longer a subsequence, the more likely a label changes within a subsequence. Accordingly, we may say that a moderate length of a subsequence exists. Taking this characteristic into account, we empirically find the best length of a subsequence under the criterion of the classification rate.

Fig.4 indicate that modules in white color have never become a winner for any subsequence, hence unlabeled. For labeling unlabeled modules, we adopt a similar labeling method as the conventional SOM; label an unlabeled module by the label of the subsequence with the minimum Mean Square Error (MSE). Fig.5 illustrates the resulting fully labeled task map. Because most of the robot motion is forward movement, many unlabeled modules in Fig.4 are labeled by "F" in Fig.5. The resulting task map is then evaluated using a novel dataset. Fig.6 depicts the test results, which enables to count the number of misclassifications; the number of misclassifications is 11 and the equivalent number of misclassifications is 10.5.

4.3 Generating a Task Map Based on Multiple Paths

For better generalization, we propose to create a task map based on multiple paths. In our experiment, we use 4 paths obtained from the same environment but with slightly different routes. The detailed procedure is:

– Generate four paths by moving a Khepera II robot in the forward direction twice as in Fig.2, and in the reverse direction twice.

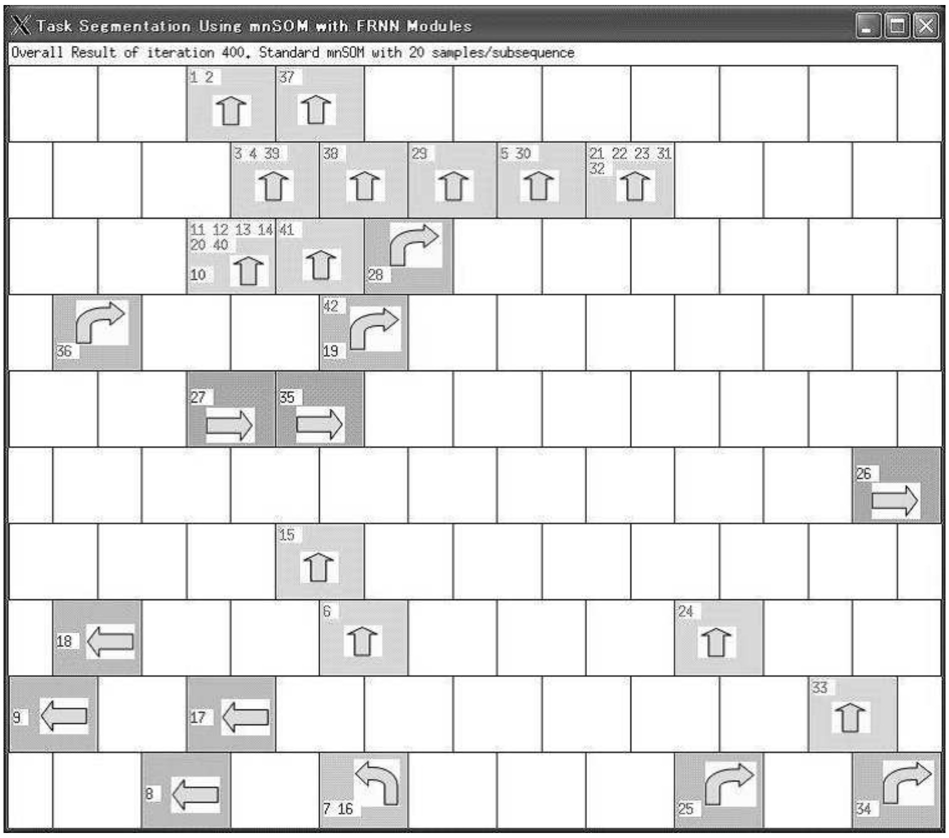
Task Segmentation Using mnSOM with FRNN Modules
Overall Result of iteration 400, Standard mnSOM with 20 samples/subsequence

Fig. 4 Resulting mnSOM trained by subsequences of the length 20 in a single path

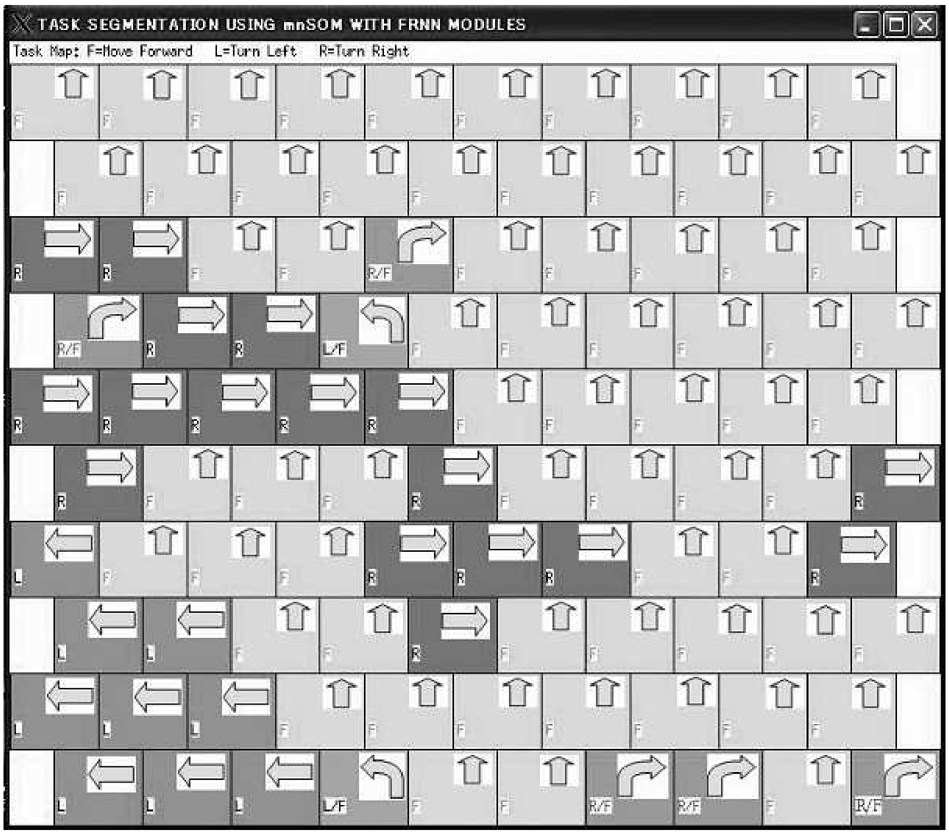
TASK SEGMENTATION USING mnSOM WITH FRNN MODULES
Task Map: F=Move Forward L=Turn Left R=Turn Right

Fig. 5 Fully Labeled Task Map based on a Single Path Composed of Subsequences with the Length 20

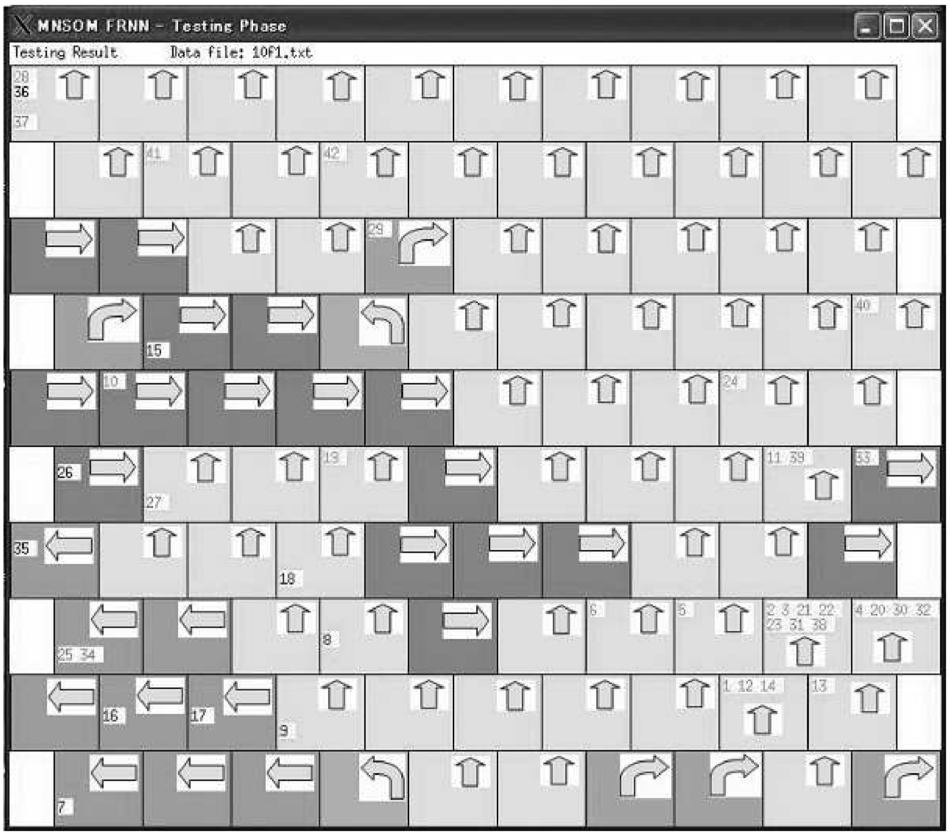

Fig. 6 Test Result of the Task Map based on a Single Path Composed of Subsequences with the Length 20

Table 4 Label Combinations

"X" stands for unlikely label combination.

	First Label				
Second Label	L	R	F	L/F	R/F
L	L	X	L/F	L/F	X
R	X	R	R/F	X	R/F
F	L/F	R/F	F	L/F	R/F
L/F	L/F	X	L/F	L/F	X
R/F	X	R/F	R/F	X	R/F

– Train mnSOM based on them. At each mnSOM iteration, all paths are used in random order for stable learning.
– Label modules in the resulting mnSOM as follows. In cases where a module becomes a winner for subsequences with different labels, we adopt the majority voting. When it does not provide a solution, we decide the label based on rules in Table 4. In case of more than 2 different labels, we repeatedly apply rules in Table 4 starting from the first two labels. If "X" in Table 4 occurs, the corresponding module is left unlabeled, but it is considered to be unlikely. Fig. 7 illustrates the resulting task map. Again, not all modules become a winner. Hence, it is necessary to assign labels to those modules.

4.3.1 Labeling of Unlabeled Modules We assign labels by the same labeling method as in Section 4.2. Fig.8 shows the resulting fully labeled task map based on multiple paths. Each module is an expert representing the nonlinear dynamics of the corresponding subsequences.

Table 5 Between- and within- class distance matrix corresponding to Fig.8

	L	R	F	L/F	R/F
L	1.868	2.666	4.213	3.309	3.569
R	2.666	0.848	5.656	2.647	1
F	4.213	5.656	3.555	3.692	5.997
L/F	3.309	2.647	3.692	1.454	2.544
R/F	3.569	1	5.997	2.544	3.6

$\sum$(between class distance) = 35.2944
$\sum$(within class distance) = 11.325
Ratio = 35.2944/11.325 = 3.1165

Table 6 Between- and within- class distance matrix corresponding to Fig.9

	L	R	F	L/F	R/F
L	3.255	2.923	4.043	3.064	3.343
R	2.923	1.536	3.426	2.162	0.449
F	4.043	3.426	3.572	1.299	3.695
L/F	3.064	2.162	1.299	5.021	2.476
R/F	3.343	0.449	3.695	2.476	1.978

$\sum$(between class distance) = 26.8817
$\sum$(within class distance) = 15.362
Ratio = 26.8817/15.362 = 1.7499

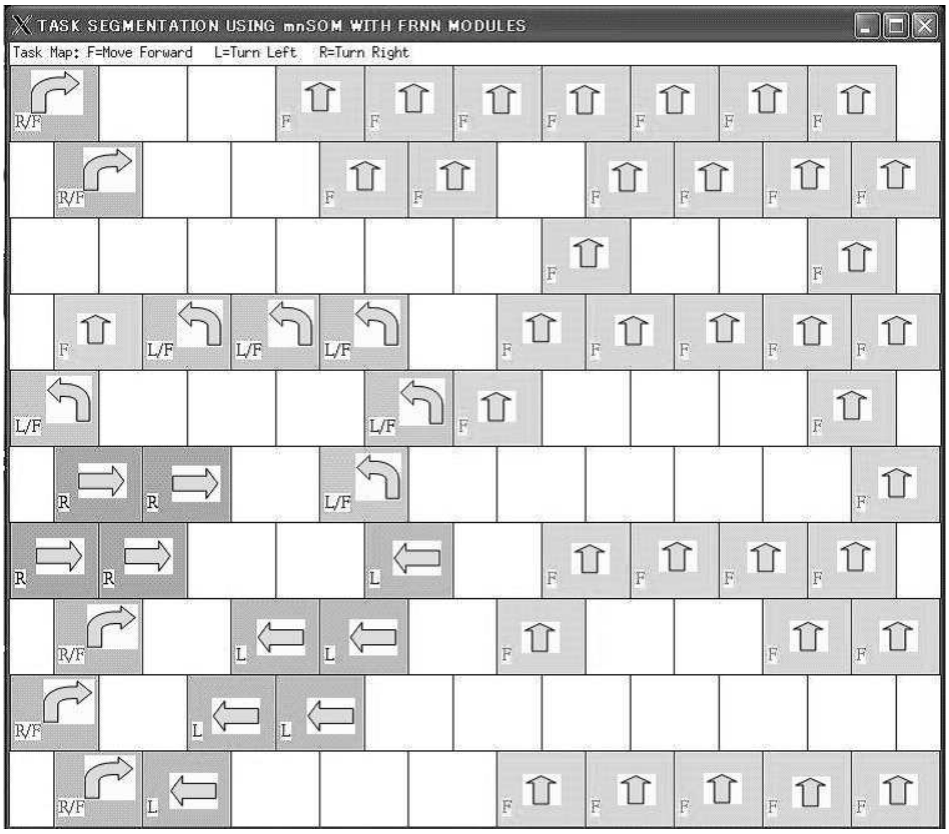
TASK SEGMENTATION USING mnSOM WITH FRNN MODULES
Task Map: F=Move Forward L=Turn Left R=Turn Right

Fig. 7 Resulting Task Map based on Multiple Paths

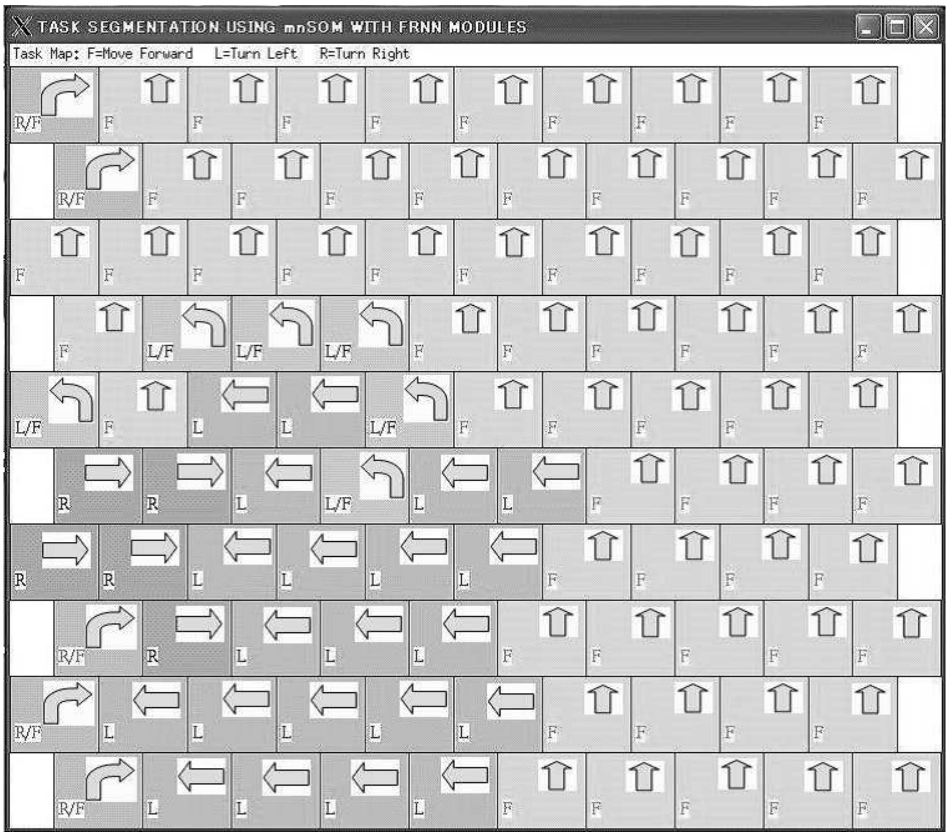
TASK SEGMENTATION USING mnSOM WITH FRNN MODULES
Task Map: F=Move Forward L=Turn Left R=Turn Right

Fig. 8 Fully Labeled Task Map based on Multiple Paths

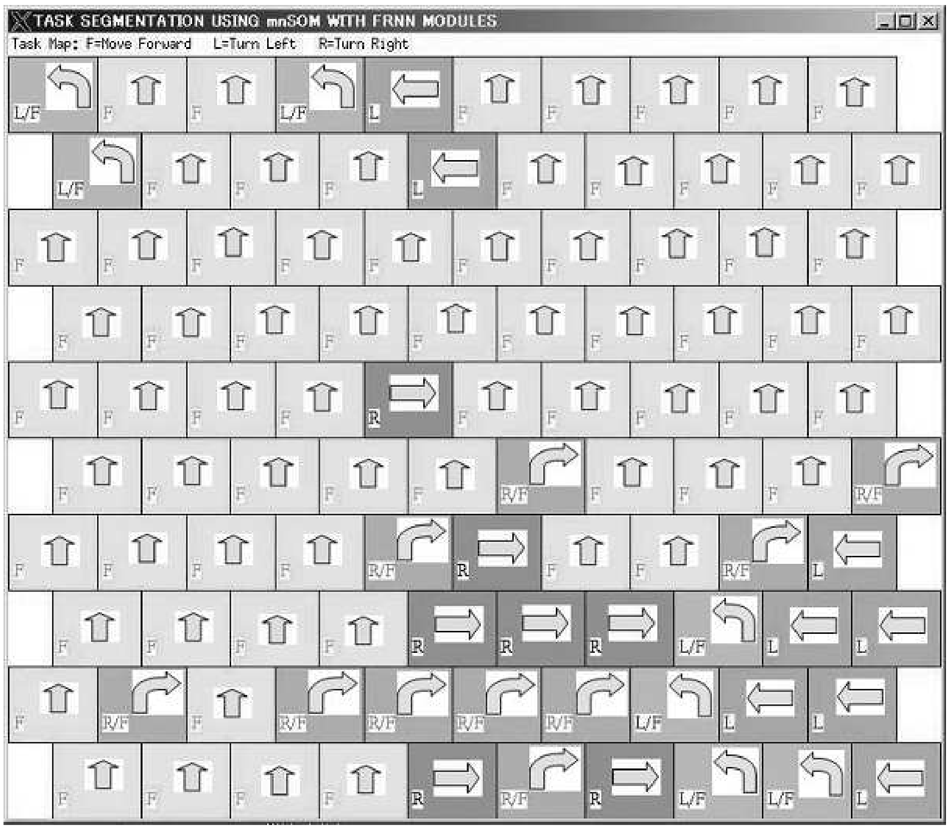

Fig. 9 Fully Labeled Task based on Multiple Paths[14]

4.3.2 Comparative Study We proposed a similar approach taking temporal continuity of winner modules into account [14]. It introduced a threshold function to the competitive process in mnSOM to prevent rapid switching of winner modules. Since the winner module for current subsequence will tend to the same as the previous one, it was hope that the temporal continuity of the resulting mnSOM increased. Figure 9 illustrates the resulting fully labeled task map based on multiple paths in our previous study[14].

Tables 5 and 6 provide within- and between- class distances for our proposal and our previous study[14], respectively. They indicate that our proposal is superior to our previous study [14]. Our proposal has smaller within class distances (except for R/F) and bigger between class distances (except for distance between L and R). The large value of the ratio of "Between class distance" to "Within class distance" also indicates the superiority our proposal to our previous study [14].

4.3.3 Test Results We evaluate the resulting task map using a novel path. Each module in Fig. 10 represents subsequence numbers for which the corresponding module becomes a winner. Comparison between Fig. 10 and Fig. 7 indicates that 11 winner modules out of 27 winner modules for a novel path (40.74%) have not become a winner during training. We can interpret that mnSOM takes advantage of its module interpolation capability to find an appropriate expert module for a novel subsequence. Table 7 summarize the overall performance for training and test. As before, subsequences with the length 20 is the best.

Table 7 Classification and segmentation performance of the resulting task map

NS*stands for length of a subsequence.
†stands for the best result in [14]

NS*	The number of misclassifications					Correct Segmentation rate (%)	
	Datasets					training datasets	novel dataset
	1	2	3	4	novel		
10	6	6	5	8	15	92.56	82.14
15	4.5	1.5	3.5	2	5	94.87	91.07
20	1.5	1.5	2.5	0	2.5	96.73	94.05
25	3	1.5	1	2	4.5	94.32	86.36
30	2	1.5	1	2	3.5	94.30	87.50
20†	3.5	0.5	2	1	4.5	95.80	89.30

4.4 Computational Cost

To exhibit module interpolation capability of mnSOM, it is better to use many modules. This, however, increases computational cost. Taking this trade-off into account, we use 10x10 modules in mnSOM.

We use a PC with Pentium 4 (3.2GHz, 1GB RAM). mnSOM training based on a single path takes 8.5 hours

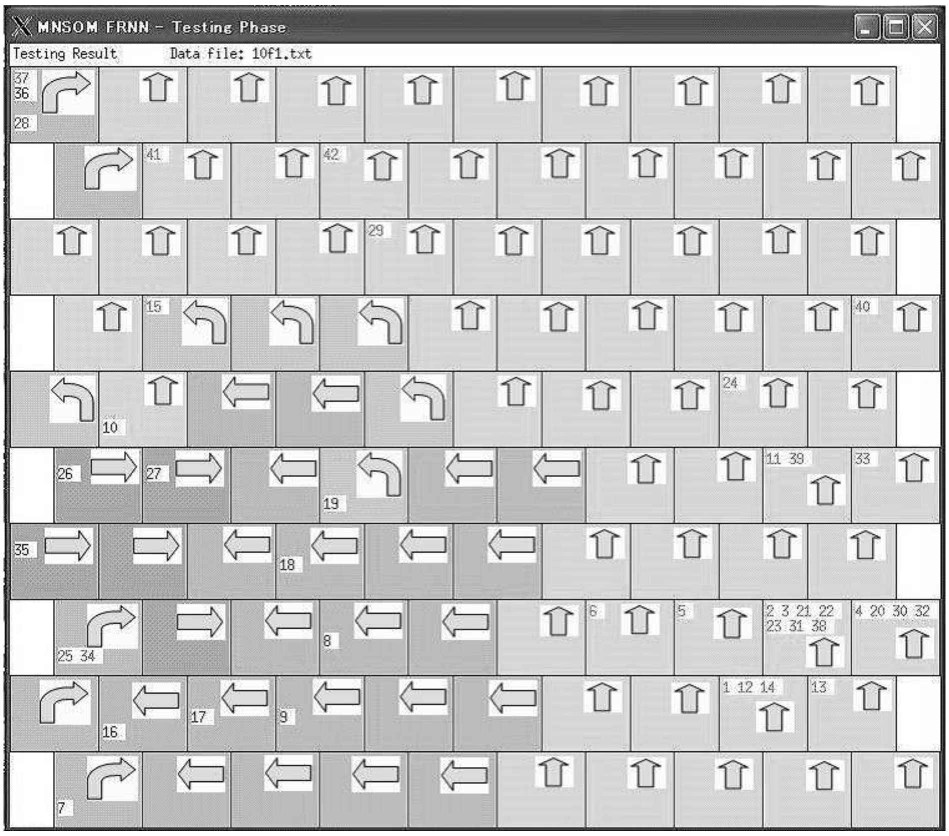

Fig. 10 Test result for a novel dataset

for 400 mnSOM iterations, and 34 hours based on 4 paths. On the other hand, mnSOM test takes only 1.79 seconds for a single path comprising 843 samples. It means that it takes 2.12 m seconds (1.79/843) at each time step. Since significant portion of computational cost is for graphical interface, the actual computational cost could be smaller than this.

Supposing that the training is over, this run-time is small enough for real time control of a mobile robot. In a changing environment, however, on-line learning is required. This takes a lot of time, hence some hardware such as FPGA is necessary for this on-line learning.

5 Conclusions and Discussions

We proposed in the present paper to use mnSOM for task segmentation in navigation of a mobile robot. Modification of the standard mnSOM has been proposed. In contrast to the standard mnSOM, data classes are unknown a priori, hence a path is decomposed into shorter subsequences, supposing that a class label does not change within a subsequence. We explored subsequences with varying lengths. Subsequences with the length of 20 produce the best classification performance of 97.62% for training samples.

To generate a task map with larger generalization, we proposed to train mnSOM based on multiple paths. The combined task map provides a map of expert modules. The resulting mnSOM using subsequences with the length of 20 produces the best segmentation performance of 96.73% for training data and 94.05% for a novel data.

One might consider that our proposal is symbolic, i.e., discretized, control, since we propose modules and one of them becomes a winner for a subsequence. However, in contrast to the so-called symbolic approach, module interpolation is possible in our approach. We can interpret that mnSOM takes advantage of its module interpolation capability to find an appropriate expert module for a new subsequence.

Although we clarified advantages and disadvantages of our proposal and [7], we haven't done comparative study yet, which is left for further research.

In the current study sensory-motor signals are obtained from a real mobile robot and task segmentation is done successfully based on them. The original purpose of mnSOM, however, was to provide desirable control of a mobile robot based on the resulting task segmentation. Since the outputs of the winner module provide sensory signals and motor commands at the next time step, the winner module can in principle provide motor commands for a mobile robot. However, if we use the same algorithm for determining the winner module during training and test, control for the latter is not satisfactory due to a big time delay. To deal with this issue, we are currently doing research to incorporate previous sensory-motor informations and current sensor informations to generate appropriate motor commands at the next time step. We will report the results in the near future.

Acknowledgments.

This research was partially supported by the 21st Century COE (Center of Excellence) Program and by Grant-in-Aid for Scientific Research(C)(18500175) both from the Ministry of Education, Culture, Sports, Science and Technology(MEXT), Japan.

References

1. Furukawa, T., Tokunaga, K., Kaneko, S. , Kimotsuki, K., and Yasui, S.: Generalized self-organizing maps (mnSOM) for dealing with dynamical systems, in Proc. of 2004 International Symposium on Nonlinear Theory and Its Application (NOLTA2004), Fukuoka, Japan (2004) 231-234
2. Furukawa, T., Tokunaga, K., Morishita, K., and Yasui, S.: Modular Network SOM (mnSOM): From Vector Space to Function Space, in Proc. of IJCNN2005, Canada (2005)
3. Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G.: Adaptive Mixtures of Local Experts, Neural Computation, **3** (1991) 79-87
4. Tokunaga, K., Furukawa, T., and Yasui, S., Modular Network SOM: Extension of SOM to the realm of function space, in Proc. of Workshop on Self Organizing Maps (WSOM2003), (2003) 173-178
5. Nolfi, S., and Tani, J.: Extracting Regularities in Space and Time Through a Cascade of Prediction Networks : The Case of a Mobile Robot Navigating in a Structural Environtment, Connection Science, (11)2, (1999) 129-152
6. Tani, J. : Model-based Learning for Mobile Robot Navigation From Dynamical System Perspective, IEEE Transactions on System, Man and Cybernetics Part B, 26(3), 421-436, (1996)
7. Tani, J., and Nolfi, S.: Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems, Neural Networks, 12, (1999) 1131-1141.
8. Tani, J., Ito, M., and Sugita, Y.: Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB, Neural Networks, Vol.17, (2004) 1273-1289
9. Kohonen, T.: Self-Organizing Maps, Springer, 1995
10. Martinetz, T.: Neural-Gas network for vector quantization and its application to time-series prediction, IEEE Transactions on Neural Networks, Vol.4(4), (1993) 558-569
11. Walter,J.A., and Schulten, K.J.: Implementation of self-organizing neural networks for visuo-motor control of an industrial robot, IEEE Transactions on Neural Networks, Vol.4(1), (1993) 86-95
12. Williams, R.J., and Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity, In Y. Chauvin and D. Rumelhart, editors, Backpropagation: Theory, Architectures and Applications. Erlbaum, (1992) 433-486
13. Wolpert, D.M., and Kawato, M.: Multiple paired forward and inverse models for motor control, Neural Networks, Vol.11, (1998) 1317-1329
14. M. Aziz Muslim, Masumi Ishikawa, and Tetsuo Furukawa: A New Approach to Task Segmentation in Mobile Robots by mnSOM, Proc. of 2006 IEEE World Congress on Computational Intelligence (IJCNN2006 Section), Vancouver, Canada (2006) 6542-6549