Segmentation of Environment and Formation of Graph-based Maps in Mobile Robots

Muhammad Aziz Muslim

Supervisor: Prof. Masumi Ishikawa

A thesis submitted to the Kyushu Institute of Technology
in partial fulfillment of the requirement for
the degree of Doctor of Philosophy

Department of Brain Science and Engineering
Graduate School of Life Science and Systems Engineering
Kyushu Institute of Technology, Japan

March 2008

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Masumi Ishikawa, for his supports, supervisions and advices. Following his advices, complex thing becomes simpler and simple thing becomes scientifically sounds. My special thanks to Prof. Furukawa and Dr. Tokunaga who introduced me to mnSOM. Thanks also to Dr. Hong Zhang, Dr. Ali, Dr. Kamei and all Ishikawa Lab. members for togetherness, understanding and cooperations. Hope this will continue in the future.

Also I would like to express my thanks to my mother, father and sisters for their unbounded love and supports. Also to my beloved family, abundance support, love and patience of my wife, Titik, and my children, Filza, Haidar and Alya are reasons to stay on the hard way of pursuing Ph.D degree.

Last but not least, I would like to express my special thank to the Japanese government for supporting my Ph.D study through monbukagakusho scholarship. This research was also partially supported by the 21st Century COE (Center of Excellence) Program, and by Grant-in-Aid for Scientific Research(C)(18500175) and (C)(2)(15500140) from the Ministry of Education, Culture, Sports, Science and Technology(MEXT), Japan.

Muhammad Aziz Muslim
March 2008

Abstract

A new approach in Artificial Intelligence (AI), which focuses on agent's interaction with the world, is expected to solve difficulties in the classical AI. The interaction leads an agent to exhibit emergent behaviors, which are not preprogrammed by a designer. This is what biological agents (i.e. animals and humans) do in their daily life. This dissertation aims at finding mechanisms necessary for this. A mobile robot is used as a test bed for this purpose.

The real world is completely different from a virtual world frequently used in the classical AI. The real world is always subject to complexity, noise, and nonlinearity. Information on the real world is often spatio-temporal in nature, and is hard to do information processing in real time. Solving real world problems often leads to unsatisfactory results due to its inherent difficulties. One promising approach to this is to segment the spatio-temporal information into meaningful elements. The purpose of the present thesis is to segment the world and to form a graph-based map for efficient processing.

Task segmentation in navigation of a mobile robot based on sensory signals is important for realizing efficient navigation, hence attracted wide attention. In this research, a new approach to segmentation in a mobile robot by a modular network SOM (mnSOM) is proposed. In a mobile robot, the standard mnSOM is not applicable as it is, because it is based on an 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. The resulting mnSOM demonstrates segmentation performance of 94.05% for a novel dataset based on an unrealistic assumption that winner modules corresponding to subsequences in the same class share the same label. Since this is not at all practical, the current study proposes segmentation without this unrealistic assumption.

Firstly, the conventional hierarchical clustering is applied to the resulting mnSOM. Without the above unrealistic assumption, its segmentation performance deteriorates by only 1.2%. Hierarchical clustering assumes that the distances between any pair of modules are provided with precision, but this is not the case in mnSOM. Accordingly, this is followed by a clustering based on only the distance between spatially adjacent modules with modification by their temporal contiguity. This clustering with spatio-temporal contiguity provides superior performance to the conventional hierarchical clustering.

Based on the resulting mnSOM, a graph-based map is formed. Due to stochastic character of sensory-motor information, I propose to use Hidden Markov models (HMMs) instead of a deterministic method. Given a sequence of data, mnSOM produces sequence of labels, which may includes erroneous ones due to noise. HMMs are employed for better estimates of labels. Finally, from the resulting sequence of labels, L-junctions and T-junctions are located, and are used as nodes for constructing a graph-based map. For comparative study, vector quantization of sensory-motor signals is also tried. The resulting HMMs based on the quantized data also generate a graph-based map.

The resulting graph-based map also contributes to goal seeking. Simulation result shows that the resulting graph-based map is efficient for goal seeking, since it is not necessary to construct a new map every time the environment changes.

Chapter 1

Introduction

Classical artificial intelligence (AI) has faced many difficulties such as the symbol grounding problem (i.e. providing relationship between symbols and the real world), the embodiment problem (i.e. intelligence requires a physical body) and the situatedness problem (i.e., an agent embedded in an environment). To solve these problems, a new approach in AI, (some references refer it to "embodied cognitive science") has been extensively explored since 1990s. In contrast to the classical AI and the conventional paradigm of information processing, this new approach capitalizes agent's interaction with the real world.

An agent is situated if it exists in a dynamic environment, which it can manipulate or change through its actions, and which it can sense or perceive [11]. Situatedness is expected to make an agent exhibit emergent behaviors: behaviors not programmed into agents by a designer. To have this capability an agent must be equipped with an appropriate mechanism. The present thesis aims to find the mechanism. Mobile robots are used as a testbed for finding and demonstrating this mechanism.

There have been many approaches towards this mechanism such as neural networks, fuzzy logic, and control theory. Humans and animals naturally exhibit emergent behaviors in their daily life owing to a mechanism in their brains. This suggests the importance of finding a brain-inspired mechanism for this emergence. We believe neural networks inspired by natural brains, contribute to finding the mechanism and to understanding naturalintelligence.

The real world is very different from a virtual world in the classical AI. The real world is always subject to noise and disturbances, and has nonlinearity. More importantly, information in the real world is huge and spatiotemporally continuous, hence information processing based on this tends to be quite complex and time consuming. We believe segmenting it into mean-

ingful elements is one of the origins of intelligence, and solves difficult tasks just as we humans do.

Segmentation, here, is decomposing the world into meaningful elements based on continuous spatio-temporal sensory information. Segmentation can also be found in humans. The central issue in this thesis is to realize segmentation based on continuous spatio-temporal information, and efficient information processing based on the resulting segmentation. Concerning the former, there have been various studies so far. Taking into consideration their advantages and disadvantages, I propose an idea of combining modular network self-organizing map(mnSOM) and clustering for effective segmentation. Concerning the latter, I propose an idea of using Hidden Markov models (HMM) for internal symbolic representation of the world based on probabilistic interpretation of the resulting segmentation.

Segmentation is a process of creating symbols from continuous data, hence needs a mechanism of competition such as winner-take-all among elements. The conventional approach to this is the use of competitive learning among modular networks. In contrast to this, I propose to use a modular network self-organizing map due to its characteristic of topology preservation: similar modules being located nearby. This enables generation of interpolated modules, provided the number of modules is sufficiently large.

Capability of interpolation among modules varies according to the architecture of modular networks. In the conventional competitive learning, modules or units are in isolation; there is exists no notion of similarity between them. There are two aspects in "interpolation". The one is creating an output interpolated by outputs from multiple modules. This could be done by a combination of multiple modules with weights determined by the soft-max function. 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 soft-max [46] is an improvement over the conventional competitive learning in that the output interpolation is possible. Similarity between modules, however, is not explicitly represented. Furthermore, interpolation and segmentation generally do not coexist; interpolation is possible at the sacrifice of segmentation using the soft-max function, and only when the soft-max function asymptotically becomes winner-take-all, segmentation is possible at the sacrifice of interpolation.

Tani et al. proposed a recurrent neural network with a parametric bias [38]. It has the ability of the output interpolation, but has no longer the capability of segmentation. Furthermore, the performance of interpolation is not satisfactory due to inherent nonlinear characteristics of neural networks.

Self Organizing Maps (SOM)[15] are popular for classifying data with topology preservation. The resulting topological maps represent the so called "unit" interpolation. Martinez [20] proposed Neural Gas (NG) to alleviate a difficulty in SOM: mapping of high dimensional input space onto a fixed lattice. Walter et al. did a comparative study of SOM and NG for positioning task of industrial robot system [44]. They found that for a given control problem both algorithm performed similarly. NG performs better when topology of sub-manifold in input space is unknown or inhomogenous.

Modular network SOM (mnSOM) [7][8][34] 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 competitive 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.

The thesis focuses on segmentation tasks in mobile robots. Having long history, many approaches have been proposed so far to do segmentation in mobile robots. Mataric [21] proposed landmark-based navigation. In this approach, the movement of a robot is temporally segmented by landmarks. Using known landmarks, the robot maintains a map of the environment and uses it for path planning. Although this approach is successfull in segmentation, it is based on known landmarks rather than robot dynamics. Venery et.al. [43] used the concept of memorizing and recalling of sensory-motor association. The behaviors of mobile robots are based on learned association between sensory inputs and motor actions. Here, robot dynamics has been explored, but the corresponding segmentation is not clear.

Extensive researches to modularize a complex task from dynamical systems perspective were done by Tani and his colleagues. In Tani [36], given the current sensory state, a recurrent neural network predicts the next sensory input. When the robot encounters a fork on the road, it decides a route by a predefined branching mechanism. Nolfi and Tani [26] proposed to use 2-level prediction networks arranged hierarchically. The activation state of each output unit of the first level prediction network is fed into a winner-take-all layer, serving as a segmentation network. The output of the segmentation network then fed into the second level prediction layer. The final result showed that the network was able to extract low level regularities, such as 'walls', 'corner', and 'corridors'. Tani and Nolfi [37] further proposed to use a mixture of recurrent experts (MRE). Multiple RNN modules compete each other to become an expert in predicting sensory-motor

flow for a specific behavior through learning. The switching among winner RNN modules, which was performed through learning of gate opening dynamics, corresponds to temporal segmentation of sensory motor flow. The above study by Tani et.al. was successfull in segmentation in mobile robots. However, interpolation among modules was not possible. Furthermore, how to use the resulting segmentation in mobile robot was not discussed.

Aim of the present thesis is to segment the world based on sensory-motor signals using mnSOM, and to develop a graph-based map. To cope with nonlinear dynamics of a mobile robot in the environment, a modular network with good segmentation and interpolation capability is needed, and mnSOM is a good candidate for this. However, standard mnSOM is based on the assumption that class labels are known a priori. In case of a mobile robot, however, only an unsegmented sequence of data is available. I propose to divide a sequence into many subsequences, assuming that a class label does not change within a subsequence.

In constructing graph-based maps, care must be taken because IR sensors are corrupted by noise, hence a deterministic approach is not appropriate. I propose to use Hidden Markov models (HMMs) to handle stochastic sensory signals properly. I propose two alternatives. The former is to use the resulting labels by mnSOM as inputs to HMMs. The latter is to use quantized sensory signals as inputs to HMMs.

In chapter 2, a modular network SOM (mnSOM) and related studies are briefly surveyed to make clear our standpoint. First is a pioneering work of mixture of local experts. This is followed by mixture of recurrent experts which is an extension of mixture of local experts. A brief explanation of self organizing maps follows. Finally, a generalization of SOM called modular network SOM (mnSOM)is presented. In chapter 3, modification of the standard mnSOM applicable to mobile robots is given. This is followed by task segmentation in mobile robots by modified mnSOM. To solve difficulties in this, task segmentation by clustering of the resulting mnSOM is proposed. The one is by hierarchical clustering, followed by a novel clustering with spatial contiguity, and the other is clustering with spatio-temporal contiguity. In chapter 4, two methods for formation of graph-based maps of the environment are proposed. The one is Hidden Markov Models (HMMs) based on the resulting mnSOM, and the other is HMMs based on quantized sensory-motor signals are given. As a case study, simple goal seeking in mobile robots is demonstrated. In chapter 5, experimental results of task segmentation are presented in detail. Chapter 6 explains experimental results on graph-based map formation using Hidden Markov Models. Chapter 7 concludes this thesis and discuss future research direction.

Chapter 2

Overview of Related Studies

This chapter briefly presents adaptive mixture of local experts, mixture of recurrent experts, SOM and modular network SOM(mnSOM) as its extension.

2.1 Adaptive Mixture of Local Experts

Adaptive Mixture of Local Expert is a pioneer work in modular network. This approach can be viewed as a modular version of multilayer neural network or as an associative version of competitive learning [12]. Figure 2.1. depicts

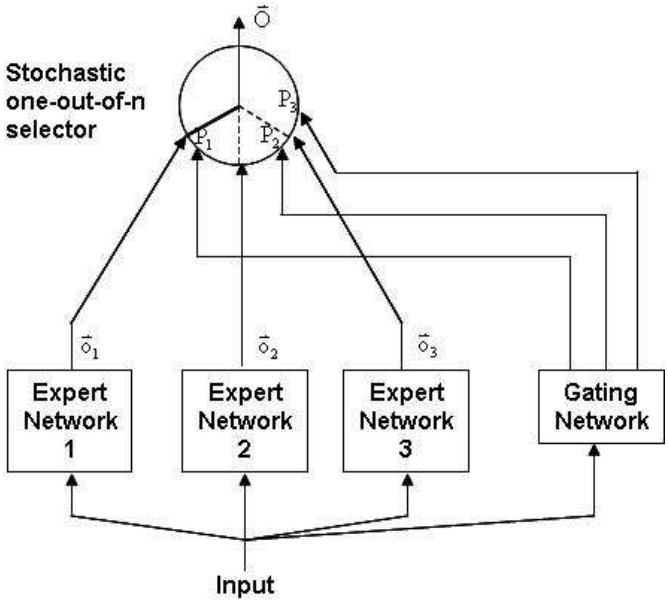


Figure 2.1: Adaptive Mixture of Local Experts.

the architecture of adaptive mixture of local experts. It is based on the competition among local experts in the form of multilayer networks doing supervised learning. Instead of linearly combining the outputs of separate experts, the gating network makes a stochastic decision on which expert to use at each instant. The output of gating network j is

$$p_j = \exp(x_j) / \sum_i \exp(x_i) \tag{2.1}$$

where x_j is the total weighted inputs provided to output unit j of the gating network. Error function defined as,

$$E^c = -\log \sum_i p_i^c e^{-\frac{1}{2}\left\|\mathbf{d}^c - \mathbf{o}_i^c\right\|^2} \tag{2.2}$$

where $\mathbf{d}$ is a target of the network and $\mathbf{o}$ is the corresponding output of the network. Finally, connection weights of the network are adapted according to

$$\frac{\partial E^c}{\partial o_i^c} = -\left[\frac{p_i^c e^{-\frac{1}{2}\left\|\mathbf{d}^c - \mathbf{o}_i^c\right\|^2}}{\sum_j p_j^c e^{-\frac{1}{2}\left\|\mathbf{d}^c - \mathbf{o}_j^c\right\|^2}}\right](\mathbf{d}^c - \mathbf{o}_i^c) \tag{2.3}$$

In this study, each expert modules are in isolation. In contrast to this, we proposed to use mnSOM [23][24], which allows cooperation among modules.

2.2 Mixture of Recurrent Experts

Tani and Nolfi [37] proposed a neural network structure inspired by mixture of local experts [12]. They used multiple-module Recurrent Neural Networks (RNNs), each of which competes with each other to become a winner as expert at predicting the sensory-motor flow for a specific behavior. The experts improve their performance through learning processes. The switching between the winner RNN modules can be regarded as temporal segmentation of the sensory-motor flow.

The architecture of mixture of recurrent experts is shown in Figure 2.2. In this figure , x_t, y_{t+1}^i, y_{t+1}^*, and g_t^i are the input, output, target and gate opening of the i-th RNN modules, respectively. In contrast to the architecture of Adaptive Mixture of Local Experts, gate opening function is derived from the internal value rather than using output of each modules directly. The gate opening of the i-th RNN modules is defined by

$$g_t^i = \frac{s_i^k}{\sum_{j=1}^n s_j^k} \tag{2.4}$$

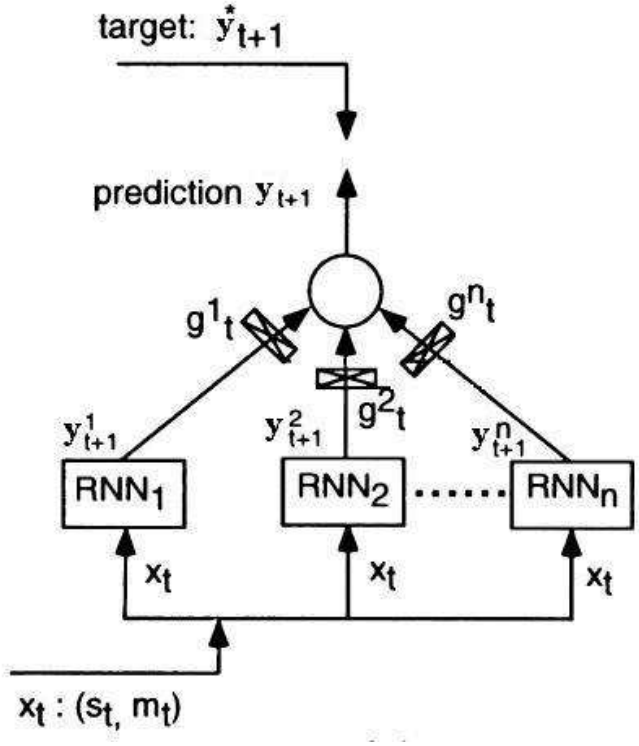


Figure 2.2: Mixture of Recurrent Experts.

where s_i^k is internal value of i-th gate opening at k-th step in a sliding window. The total output of the network is

$$y_{t+1} = \sum_{i=1}^{n} g_t^i y_{t+1}^i \tag{2.5}$$

Connection weights in each RNN and the gate opening are updated such that the following likelihood function is maximized

$$\ln L = \ln \sum_{i=1}^{n} g_t^i . e^{(-1/2\sigma^2)\left\| y_{t+1}^* - y_{t+1}^i \right\|^2} \tag{2.6}$$

where σ denotes a scaling parameter.

The direction of update for each internal gate opening value is determined by

$$\frac{\partial \ln L}{\partial s_t^i} = g(i|x_t, y_{t+1}^*) - g_t^i \tag{2.7}$$

where $g(i|x_t, y^*_{t+1})$ is explicitly given by

$$g(i|x_t, y^*_{t+1}) = \frac{g^i_t . e^{(-1/2\sigma^2)\left\|y^*_{t+1} - y^i_{t+1}\right\|^2}}{\sum_{j=1}^{n} g^j_t . e^{(-1/2\sigma^2)\left\|y^*_{t+1} - y^j_{t+1}\right\|^2}} \tag{2.8}$$

The error in each RNN module is calculated by

$$\text{error}^i_{t+1} = g(i|x_t, y^*_{t+1}).(y^*_{t+1} - y^i_{t+1}) \tag{2.9}$$

The update of connection weights and gate opening are computed through the use of BPTT [45] on a sliding window. The update of the gate internal state in sliding window at k-th step is obtained as

$$\Delta s^i_k = \epsilon_g . \frac{\partial \ln L}{\partial s^i_k} - \eta_g . (s^i_k - s^i_{k-1}) \tag{2.10}$$

The scheme proposed above was evaluated by simulation of navigation task. In their experiment, a mobile robot equipped with 20 laser range sensors maneuvering in collision-free using a variant of the potential method as in J. Tani [36] is moved around in two rooms. The robot moves around one room three times, then enters another room and moves around three times. Task segmentation result was evaluated based on the recorded gate opening and sensory-motor signals in its robotics field.

Generally speaking, performance of this study strongly depends on proper selection of parameters, such as a scaling parameter, which is determined by a designer. Unfortunately, it is very difficult to find a proper parameter value in each cases. Furthermore, improper parameter setting leads to computational instability. On the contrary to this, we use mnSOM for our study [23][24], which is proved computationally stable due to careful learning rate assignment by the algorithm during learning.

2.3 Self Organizing Maps (SOM)

The most popular unsupervised learning algorithm is self organizing maps (SOM) proposed by Kohonen [14]. Figure 2.3 depicts the architecture of SOM. The principal aim of self-organizing maps is to transform a set of signal patterns of arbitrary dimension into a set of lower dimensional vectors, and to locate them on a map in a topologically ordered fashion.

There are two training methods, i.e., on-line learning and batch learning. On-line learning makes weight adaptation whenever an input is provided. On the other hand, batch learning SOM makes weight adaptation after all input data are presented to the network.

SOM algorithm consists of the following 4 processes:

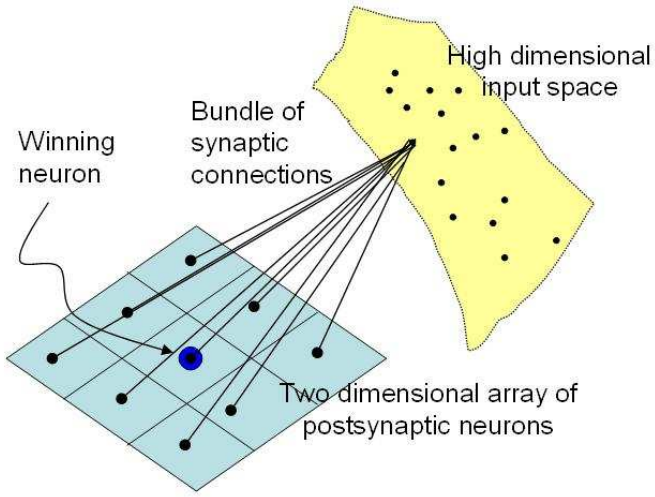


Figure 2.3: Kohonen model of Self Organizing Maps.

1. Evaluative Process

   Initializing synaptic weights in the network, neurons in the network repeatedly compute their distance to each input data.

$$E_i^k = \frac{1}{2} \left\| x_i - w^k \right\|^2 \text{for all} k \tag{2.11}$$

   In this equation, i and k are input number and node number, respectively. While x_i and w^k are input pattern and synaptic weight, respectively.

2. Competitive Process

   Using the distance function in Eq.(2.11) as the basis for competition among neurons, the particular neuron with the smallest distance is declared as the winner of the competition. In case Euclidean criterion is used, the winner of the competition is neuron with the minimum Euclidean distance to the particular input pattern.

$$k_i^* = \arg_k \min E_i^k \tag{2.12}$$

3. Cooperative Process

   Topological neighborhood function centered in the winning neuron is determined based on

- On-line learning:

$$\phi_i^k = h(k, k_i^*) \tag{2.13}$$

- Batch learning:

$$\psi_i^k = \frac{h(k, k_i^*)}{\sum_{i'} h(k, k_{i'}^*)} \tag{2.14}$$

if Gaussian neighborhood function is used:

$$h(k, k_i^*) = \exp\left[-\frac{l^2(k, k_i^*)}{2\sigma^2}\right] \tag{2.15}$$

where $l^2(k, k_i^*)$ is lateral distance to the winning neuron. The parameter σ is the effective width of the topological neighborhood. Usually the size of topological neighborhood shrinks with time. In this case, σ can be chosen in the form of:

$$\sigma(t) = \sigma_{min} + (\sigma_{max} - \sigma_{min}) \exp(-t/\tau) \tag{2.16}$$

where τ is time constant.

4. Adaptive Process

In adaptive process each synaptic weight of the network were updated based on neighborhood functions:

- On-line learning:

$$\Delta w^k = \eta \sum_i \phi_i^k (x_i - w^k) \tag{2.17}$$

η is learning rate parameter, and this parameter can be decreasing with time such as

$$\eta(t) = \eta_{min} + (\eta_{max} - \eta_{min}) \exp(-t/\tau_1) \tag{2.18}$$

- Batch learning:

$$\Delta w^k = \sum_i \psi_i^k x_i \tag{2.19}$$

2.4 The mnSOM

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 a similarity measure between modules[34]. 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 [7].

Figure 2.4 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 a class data. 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) [45]. Details of mnSOM learning algorithms are as follows.

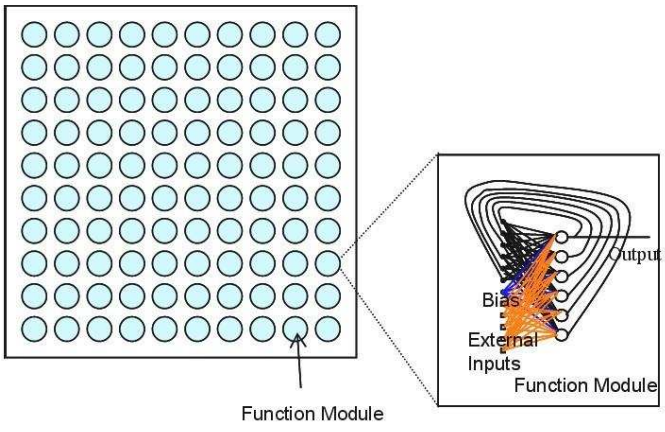


Figure 2.4: Array of modules in mnSOM and the function module as its element. The function module here is a fully connected RNN.

A learning algorithm of mnSOM is conceptually similar to that of the batch learning SOM. It consists of 4 processes [8]: evaluative, competitive, cooperative and adaptive processes. Let a set of input-output signals of a system be $\{x_{ij}, y_{ij}\}(i = 1, ..., M; j = 1, ..., L)$, where M and L are the number of classes and the number of data in each class, respectively.

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

$$E_i^{(k)} = \frac{1}{L}\sum_{j=1}^{L} \|\tilde{y}_{ij}^{(k)} - y_{ij}\|^2$$
$$k = 1, ..., K; i = 1, ..., M; j = 1, ..., L \tag{2.20}$$

where k stands for the module number, K stands for the number of modules, i stands for the class number, and j stands for the data number in each class labels.
2) Competitive process
The module with the minimum $E_i^{(k)}$ with respect to k is the winner for classes i.

$$v_i^* = \arg_k \min E_i^{(k)} \tag{2.21}$$

3) Cooperative process
A Learning rate of the module is determined by the following normalized neighborhood function:

$$\psi_i^{(k)}(t) = \frac{\phi(r(k, v_i^*); t)}{\sum_{i=1}^{M} \phi(r(k, v_i^*); t)} \tag{2.22}$$

$$\phi(r; t) = \exp[-\frac{r^2}{2\sigma^2(t)}] \tag{2.23}$$

$$\sigma(t) = \sigma_{min} + (\sigma_{max} - \sigma_{min})e^{-\frac{t}{\tau}} \tag{2.24}$$

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

$$\Delta\mathbf{w}^{(k)} = \sum_{i=1}^{M} \psi_i^{(k)}(t)(-\eta\frac{\partial E_i^{(k)}}{\partial \mathbf{w}^{(k)}}) \tag{2.25}$$

At each mnSOM iteration, repeat this back-propagation 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.

Chapter 3

Task Segmentation using mnSOM and Clustering

3.1 Introduction

Task segmentation, here, is to partition the entire movement of a robot from the start position to the end position into a sequence of primitive movements such as a forward movement or a right turn movement. Supposing a mobile robot moves in an ideal environment without obstacles and noise in sensory-motor signals, only motor signals are enough for task segmentation, and learning of dynamics would not be required. In real situations, however, a mobile robot makes slight turning movements even in a straight corridor due to obstacles and noise in sensory-motor signals. This would lead to wrong segmentation, provided only motor signals are taken into account. To cope with this difficulty, it is necessary to adopt learning of nonlinear dynamics of robot movement. One might ask, is SOM insufficient for this purpose? Generally speaking, SOM is easily affected by noise in data. In contrast to this, since mnSOM learns nonlinear dynamics of a robot movement, it is expected to be more robust than SOM. This is an advantage of mnSOM with RNN modules over SOM.

3.2 Khepera II Mobile Robot

Experiments are carried out using a Khepera II mobile robot. It has 8 infrared (IR) proximity sensors and 2 separately controlled DC motors. Both DC motors can be controlled by a PID controller executed in an interrupt routine of the robot's main processor. Figure 3.1. shows the structures of motor controllers. Two control modes can be used: the speed and position

modes. In our studies, speed mode is chosen. In speed mode, the controller has as input a speed value of the wheels, and controls the motor to keep this wheel speed. The speed modification is made as quick as possible. No limitation in acceleration is considered in this mode. The robot is controlled by a PC via serial connection. Each time step, motor commands are given to the robot. The unit is the pulse/10 ms that correspond to 8 mm/s. The maximum speed is 127 pulses/10 ms that correspond to 1 m/s. For our purpose, speed between 1 to 8 (8 mm/s up to 64 mm/s) is used.

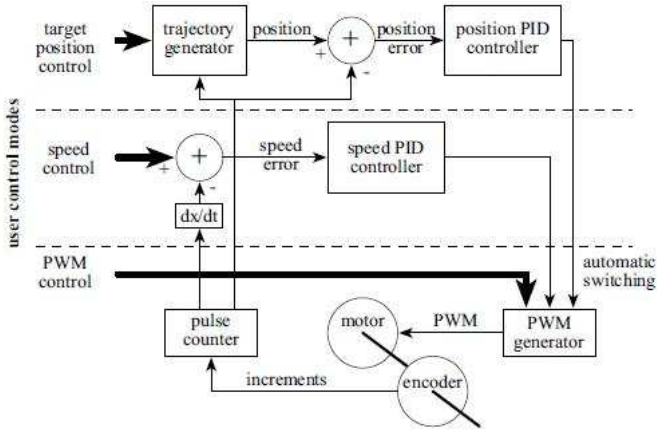


Figure 3.1: Structure of the motor Controller of Khepera II.

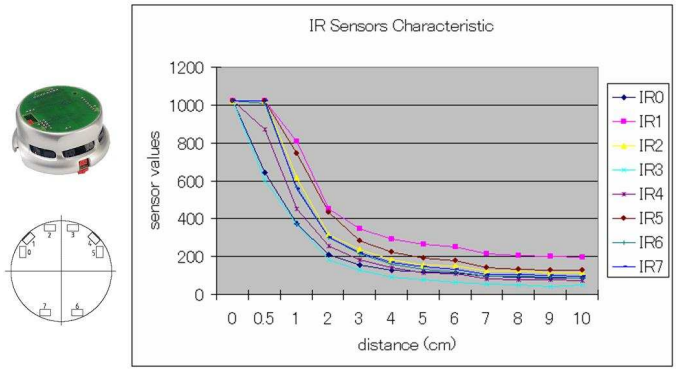


Figure 3.2: Khepera II mobile robot and its IR sensor characteristics.

Figure 3.2 shows the IR sensor characteristic of the robot. It illustrates that sensors can effectively detect an obstacle within 5 cm. Figure 3.4 depicts Khepera II mobile robot in its robotic field.

3.3 Khepera Simulator Webots

To alleviate the difficulty in working with real robot, a robot simulator, Webots 5.1.10, are used in our experiment. Webots is a three-dimensional mobile robot simulator developed by cyberbotics ltd. It provides rapid prototyping and simulation of mobile robot. Moreover, for a certain robots, direct control of the robot from webots is possible.

Webots 5 provides model of Khepera II mobile robots in its standard library. However, it is necessary to adjust their IR sensors characteristics in order to be as precise as possible to our real Khepera II. Since IR sensor in simulator is based on look up table, adjusting IR sensors characteristics are simply by inserting values of IR sensors of real robot to the table in the software. To simulate noisy environment, a noise can be added to each IR sensors. Figure 3.3. shows Khepera II robot model in Webots 5.1.10 and its sensor settings in simulator.

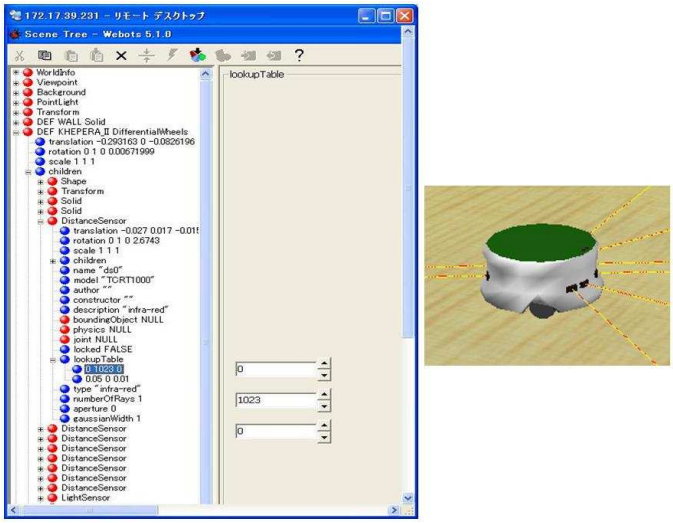


Figure 3.3: Model of Khepera II mobile robot and its IR sensor settings.

3.4 Modification on Standard mnSOM Algorithm and Data Preparation

In case of a mobile robot, 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, decompose sequence of data 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 modified mnSOM algorithm follows.

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,

$$E_i^{(k)} = \frac{1}{L}\sum_{j=1}^{L} \|\tilde{y}_{ij}^{(k)} - y_{ij}\|^2$$
$$k = 1, ..., K; i = 1, ..., M; j = 1, ..., L \tag{3.1}$$

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 subsequences.

2) Competitive process

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

$$v_i^* = \arg_k \min E_i^{(k)} \tag{3.2}$$

3) Cooperative process

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

$$\psi_i^{(k)}(t) = \frac{\phi(r(k, v_i^*); t)}{\sum_{i=1}^{M} \phi(r(k, v_i^*); t)} \tag{3.3}$$

$$\phi(r; t) = \exp[-\frac{r^2}{2\sigma^2(t)}] \tag{3.4}$$

$$\sigma(t) = \sigma_{min} + (\sigma_{max} - \sigma_{min})e^{-\frac{t}{\lambda}} \tag{3.5}$$

where $r(k, v_i^*)$ stands for the distance between module k and the winner module, v_i^*, t is the iteration number in mnSOM, σ_{min} is the minimum neighborhood size, σ_{max} is the maximum neighborhood size, and λ 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,

$$\Delta \mathbf{w}^{(k)} = \sum_{i=1}^{M} \psi_i^{(k)}(t)(-\eta \frac{\partial E_i^{(k)}}{\partial \mathbf{w}^{(k)}}) \tag{3.6}$$

At each mnSOM iteration, 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.

The robot moves from the start position to the end position by wall following on the robot field in Figure 3.4. 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.

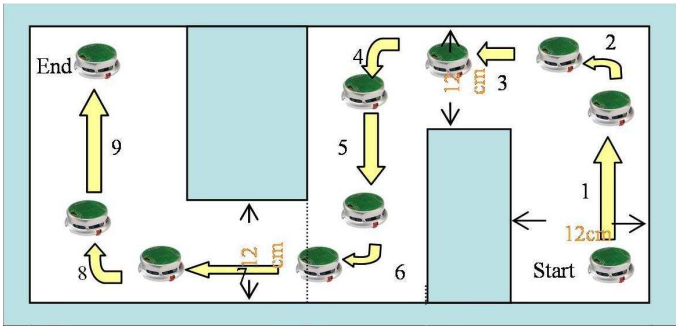


Figure 3.4: Robotic Field.

For later evaluation of training and test results, the path in Figure 3.4 is manually segmented into 9 sequences based on motor commands as in Figure 3.4. 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 Figure 3.4, 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 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 3.1, 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.

Table 3.1: 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

3.5 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 3.2 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.

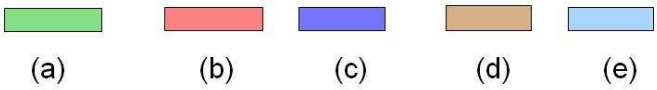


Figure 3.5: Color of module used in the resulting mnSOM : (a) forward movement (b) left turn (c) right turn (d) transition between forward movement and left turn (e) transition between forward movement and right turn

For evaluating the performance of classification, mnSOM is trained using subsequences with varying length, i.e., 10, 15, 20, 25, and 30. Each mnSOM module compete to become a winner of subsequences. Because the number

Table 3.2: Parameters in mnSOM

#x stands for the number of x.

mnSOM	
map size	10x10(100 modules)
neighborhood size (Eq.(2.24))	$\sigma_{max} = 10; \sigma_{min} = 1;$ $\tau = 80$
# 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

of subsequence is large the corresponding number of winner module, which is can be considered as the number of segmentation, also tends to be in big number. This tendency leads segmentation results to be meaningless. To alleviate this tendency, it is assumed that winner modules corresponding to subsequences in the same class share the same label.

Using the adopted assumption, the resulting mnSOM only has five labels. Five colors in Figure 3.5 are used to emphasis the corresponding five labels, i.e. "F,", "L,", "R,", "L/F," and "R/F." A module with one of the colors in Figure 3.5 indicates that it becomes a winner for a subsequence with the corresponding movement. For later examination, let a green module be called a "stationary" module, a blue module or red module be called "turning" module, and a light blue or light brown module be called "transition" module.

The resulting mnSOM is evaluated by the number 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, define the following degree of badness of misclassification.

1. The degree of badness of misclassification 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 misclassification 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.

3.6 Introducing temporal continuity into mnSOM

To prevent too rapid switching of winner modules, a threshold is introduced to the competitive process in mnSOM. The threshold for temporal continuity makes a winner module for the current subsequence tend to be the same as the previous time step, it is expected that the temporal continuity of the resulting mnSOM increases. Two types of thresholds are introduced by modifying Eq. (2.21) as follows:

1. Constant MSE Threshold.

$$v_i^* = \begin{cases} v^*_{i-1\, i\neq 1}, & \text{if } E_i^{(k)} \geq E_i^{(v^*_{i-1})} - \theta \\ \arg_k \min E_i^{(k)}, & \text{else} \end{cases} \quad (3.7)$$

2. Time Varying MSE Threshold. Since MSE tends to decrease as iteration in mnSOM proceeds, a time varying MSE threshold is considered to be more appropriate than the constant MSE threshold. An exponentially decaying MSE threshold θ is represented as,

$$\theta(t) = \theta_{min} + (\theta_{max} - \theta_{min}) e^{-\frac{t}{\mu}} \quad (3.8)$$

where θ_{min}, θ_{max}, and μ are the minimum threshold, the maximum threshold, and a decay time constant, respectively. Determination of parameters θ_{min}, θ_{max}, and μ is essential for obtaining a good result, but their appropriate determination is not easy. An alternative one is the following proportional MSE threshold

$$\beta(t) = \beta \min_k E_i^{(k)}(t) \quad (3.9)$$

where β is a proportional parameter. The value of the threshold at mnSOM iteration t is proportional to the corresponding minimum MSE.

3.7 Clustering of the Resulting mnSOM

In section 3.1-3.6, task segmentation was done by mnSOM, using prior information that winner modules corresponding to subsequences in the same

class share the same label. Since this prior information is not available in real situation, segmentation thus obtained should be regarded as the upper bound for the performance, not as a candidate for performance comparison. This section proposes to do segmentation using clustering methods based on the distance between modules in the resulting mnSOM, without using the above prior information. Firstly, conventional hierarchical clustering is used. It assumes that the distances between any pair of modules are provided with precision, but this is not the case in mnSOM. Secondly, a clustering method based on only the distance between spatially adjacent modules with modification by their temporal contiguity is proposed.

3.7.1 Hierarchical Clustering

A procedure of hierarchical clustering [5] is the following.

1. Let each module form a separate cluster.
2. Merge two clusters with the minimum distance.
3. Recalculate the distance between clusters.
4. Repeat steps 2 and 3 until the minimum distance between clusters exceeds a given threshold or the number of clusters reaches a given number of clusters.

In contrast to SOM, the definition of the distance between modules is problematic, because the distance depends on input to these modules. In addition, an essential issue in clustering is how we define the distance between modules. Suppose that

$$m_i = \arg\min_k MSE(k, i) \tag{3.10}$$

where $MSE(k, i)$ stands for the mean square error of module i given input subsequence k. The distance between modules i and j here, is defined by:

$$d_{ij} = \sqrt{(MSE(m_i, j) - MSE(m_i, i))^2 + (MSE(m_j, i) - MSE(m_j, j))^2)} \tag{3.11}$$

The inclusion of only the subsequences m and n in the definition is to prevent the distance from being blurred by many less relevant subsequences. Definition of the distance between clusters I and J follows. Suppose that the cluster I is composed of modules, $M_{I1}...M_{IR_I}$, and the cluster J is composed

of modules, $M_{J1}...M_{JR_J}$. The distance between these two clusters is defined by,

$$D_{IJ} = \frac{1}{R_I R_J} \sum_{i=1}^{R_I} \sum_{j=1}^{R_J} d_{ij} \tag{3.12}$$

where d_{ij} is the distance between two individual modules i and j as in Eq.(3.11).

3.7.2 Clustering with spatial contiguity

In mnSOM the neighboring area shrinks as learning proceeds. This suggests that the distance between modules are meaningful only within neighboring modules. On the other hand, hierarchical clustering assumes that the distance between any pair of modules is meaningfully given. Considering this issue, the following clustering method with spatial contiguity is proposed.

1. Calculate the distance between any pair adjacent modules. For module (i,j), adjacent modules are (i,j-1), (i,j+1), (i-1,j) and (i+1,j).
2. Rank order adjacent distances in increasing order.
3. Merge a pair of adjacent modules with the minimum distance.
4. Calculate the number of clusters formed by the merger.
5. Repeat steps 3 and 4 until the predefined number of clusters is obtained.

3.7.3 Clustering with spatio-temporal contiguity

In mobile robot data, temporally contiguous subsequences tend to have the same label. Accordingly, winner modules corresponding to temporally contiguous subsequences tend to have the same label. To take the temporal contiguity into account, Eq.(3.11) is proposed to be modified as follows

$$d_{ij} = \sqrt{(MSE(m_i, j) - MSE(m_i, i))^2 + (MSE(m_j, i) - MSE(m_j, j))^2} \left(1 - \exp\left(-\frac{|m_i - m_j|}{\tau}\right)\right) \tag{3.13}$$

where τ is a time constant for temporal contiguity, and m_i and m_j are subsequence numbers. In contrast to Eq. (3.11), the second term of the right-hand-side of Eq. (3.13) reduces the distance between winner modules by taking into consideration the temporal contiguity of subsequences. This modified definition of the distance is expected to have the tendency that these temporally contiguous modules have the same label.

Chapter 4

Formation of Graph-based Maps

4.1 Introduction

To acquire information from the environment, a mobile robot needs to model its environment. The resulting map is particularly useful for robot navigation. Generally speaking, there are three existing approaches to robotic mapping. A first is a grid-based approach [3], a second is a topological approach [22], and a third is a combination of a grid-based approach and a topological approach [40][41]. While the third approach claims to combine the advantages of the first two approaches, i.e. consistent and unambiguous modeling of the world and easiness of construction, it suffers from large memory.

A probabilistic approach has attracted wide attention in robotic mapping due to its effectiveness [41][42][2]. Thrun [41][42] uses a neural network model to map sensory information to probability of occupancy in each grid. He uses a laser range finder to measure the distance to nearby objects with high spatial resolution. A grid-based map approach is applied first using a probabilistic approach. A topological map is constructed on top of the resulting grid-based map.

Contrary to Thrun [41], which employs probability of occupancy for mapping, Boada et.al [2] uses hidden Markov models (HMMs) to recognize inherently distinctive places such as corridor, door opening, and so forth. Based on the recognition of places with high accuracy, the robot can do localization, path planning and navigation. However, the recognition is done using the preprocessed information obtained from a Voronoi map. Accordingly, it cannot detect small irregularities such as closed doors.

Graph-based approaches is a general approach to state relation among

entity in a set of data. If this entity corresponds to spatial information, the corresponding approach formed graph-based map. Graph-based maps are important in decreasing the computational cost. Nodes in such map correspond to "places" in the environment and the edges are paths between two places. Graph-based maps provide a concise description of the navigability of a space [1][39][40][33][6]. In our study, node represents junctions, while edge represents a corridor connecting 2 junctions.

This chapter proposes a graph-based approach to mapping a robotic field using Hidden Markov Models. Two methods are proposed: the former is to estimate HMMs based on a sequence of labels obtained by modular network SOM (mnSOM) and the latter is to estimate HMMs based on quantized sensory-motor signals. Segmentation of the environment using mnSOM is discussed in Chapter 3 and Chapter 5. Although mnSOM learns nonlinear dynamics of sensory-motor signals, it still generates labels from each subsequence separately. This might not be robust, because resulting sequence of labels may rapidly change, which rarely occurs in the real world. The combination of mnSOM and HMM is hoped to provide more robust segmentation of the environment. The resulting HMMs can be converted into a graph-based map in a straightforward way.

4.2 Overview of Hidden Markov Models (HMMs)

This subsection provides brief overview of Hidden Markov Models (HMMs). For comprehensive foundation on HMMs, refer to [30] and [31].

An HMM estimates an unobservable (hence hidden) state at each time step based on a sequence of observed simbols [30][31]. Parameters in HMM are represented as $\lambda = (A, B, \pi)$, while the notations are described in Table 1.

There are 3 basic problems of interest to be solved in HMMs:

1. Model Evaluation: Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model $\lambda = (A, B, \pi)$, efficiently compute probability of the observation sequence, $P(O|\lambda)$

2. Path Analysis: Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model λ, choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$, in some meaningful sense

3. Training problem: Adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$

Solutions to those problems are summarized as follows [31]:

Table 4.1: HMM notations

Notation	Description	
T		the length of observed sequence
N		the number of states in the model
M		the number of the observation symbols-per-state
Q	$= \{q_1, q_2, \ldots, q_N\}$	states
V	$= \{v_1, v_2, \ldots, v_M\}$	discrete set of possible symbol-observation
A	$= \{a_{ij}\}, a_{ij}$	state transition probability
	$= \Pr(q_i \text{ at } t+1 \mid q_i \text{ at } t)$	
B	$= \{b_k\}, b_j(k)$	probability distribution of observed-output
	$= \Pr(v_k \text{ at } t \mid q_j \text{ at } t)$	
π	$= \{\pi_i\}, \pi_i$	probability distribution of initial state
	$= \Pr(q_i \; at \; t = 1)$	

1. Model evaluation by "The Forward-Backward procedure"

   Consider the forward variable

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda) \tag{4.1}$$

   We can estimate $\alpha_t(i)$ as follows:

   (a) Initialization:

$$\alpha_t(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \tag{4.2}$$

   (b) Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=t}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}),$$

$$1 \leq t \leq T-1$$
$$1 \leq j \leq N \tag{4.3}$$

   (c) Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4.4}$$

2. Path analysis by "Viterbi algorithm"

   Suppose that

$$\delta_t(i) = \max_{q_1,q_2,\cdots,q_{t-1}} P\left[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t | \lambda\right] \tag{4.5}$$

   is the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state S_i. By induction we have

$$\delta_{t+1}(j) = \left[max_i \delta_t(i) a_{ij}\right] . b_j(O_{t+1}) \tag{4.6}$$

   To keep track of the argument which maximizes the above equation, new variable $\psi_t(j)$ is defined. The complete procedure then follows

   (a) Initialization

$$\delta_1(i) = \pi_i b_i(O1), \quad 1 \le i \le N \tag{4.7}$$

$$\psi_t(j) = 0 \tag{4.8}$$

   (b) Recursion

$$\begin{aligned} \delta_t(j) = \max_{1\le i\le N} \left[\delta_{t-1}(i) a_{ij}\right] b_i(Ot), \\ 2 \le t \le T \\ 1 \le j \le N \end{aligned} \tag{4.9}$$

$$\begin{aligned} psi_t(j) = \arg\max_{1\le i\le N} \left[\delta_{t-1}(i) a_{ij}\right], \\ 2 \le t \le T \\ 1 \le j \le N \end{aligned} \tag{4.10}$$

   (c) Termination

$$\begin{aligned} P^* &= \max_{1\le i\le N} \left[\delta_T(i)\right] \\ q_T^* &= \arg\max_{1\le i\le N} \left[\delta_T(i)\right] \end{aligned} \tag{4.11}$$

   (d) Path (state sequence) back tracking

$$q_t^* = \psi_{t+1}, \quad t = T-1, T-2, \cdots, 1 \tag{4.12}$$

3. HMM training by using Baum-Welch algorithm

   Using α and β calculation in forward-backward algorithm, calculate the expected number of transitions from S_i to S_j as follows

$$\sum_{t=1}^{T-1} \zeta_t(i,j) = \frac{\alpha_t(i) a_{ij} b_i(O_{t+1}) \beta_{t+1}^{(} j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_i(O_{t+1}) \beta_{t+1}^{(} j)} \tag{4.13}$$

and calculate the expected number of transitions from S_i from

$$\gamma_t(i) = \sum_{j=1}^{N} \zeta_t(i,j) \tag{4.14}$$

finally, HMM parameters are reestimated using the following formulas

$$\overline{\pi_i} = \gamma_1(i) \tag{4.15}$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \zeta_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{4.16}$$

$$\overline{b_j(k)} = \frac{\sum_{t=1 \mathrm{s.t} O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{4.17}$$

$$\tag{4.18}$$

4.3 Formation of Graph-based Maps

In estimating HMMs, observed symbols are necessary, which are discrete in nature. Two cases are considered in this chapter. The first one is to use the resulting labels obtained from mnSOM. The second one is to use quantized sensory-motor signals by vector quantization(VQ) or k-means algorithm. The resulting HMMs can be interpreted as the segmentation of the environment. Figure 4.1 illustrates the framework for recognition of the environment adopted in this paper. Block (a) in Figure 4.1 represents the recognition of the environment using HMMs based on a sequence of observed symbols obtained from k-means, named "k-means-HMM." Block (b) in Figure 4.1 represents the recognition of the environment using HMMs based on a sequence of labels obtained from mnSOM, named "mnSOM-HMMs." HMMs with left to right model is used in current study.

First step in Figure 4.1 is the determination of HMMs parameters. This is performed by Baum-Welch algorithm [30][31]. The best fit HMM is determined by model evaluation using forward algorithm [30][31]. The maximum likelihood model is chosen as the best fit model.

4.3.1 Environment segmentation using mnSOM

mnSOM provides segmentation of the robotic environment, i.e. the robot recognizes its surrounding through segmentation, based on sensory-motor signals during movement. The resulting mnSOM provides discrete environment required for formation of a graph-based map.

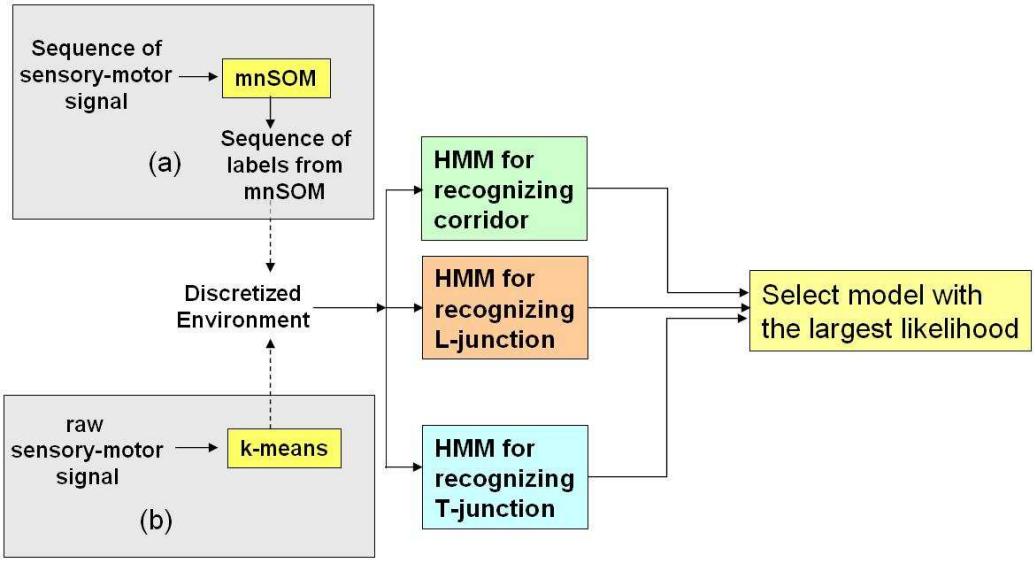


Figure 4.1: Framework of environment recognition using HMMs. (a) Discretized environment is provided by k-means.(b) Discretized environment is provided by mnSOM.

Supposing that location and orientation of a mobile robot at every time step are known, manual data segmentation into sequences of L-junction, T-junction, corridor and so on is possible. This manual segmentation is used for evaluation of the resulting segmentation. Figure 4.2. illustrate the robotic fields employed in the present study and the corresponding manual segmentation of the environment.

Dataset are divided into many subsequences with the uniform length (currently, each subsequece comprises 20 samples). For reference, subsequence label is determined based on robot location in the robotic field for the corresponding subsequence as depicted in Figure 4.2. As a consequence of uniform length of subsequences, several subsequences are stretched between two labels, hence it is necessary to generate a transient label, i.e. "L/C" for subsequences stretched between L-junction and straight corridor and "T/C" for subsequences stretched between T-junction and straight corridor.

After training, the resulting mnSOM provides a label to each module as in [24]. Given a training subsequence or a novel one, one of the modules becomes a winner. The corresponding label estimates the type of the environment (i.e. a corridor, an L-junction or a T-junction). Given a data set, mnSOM produces a sequence of labels, from which graphical representation of the environment is expected to generate.

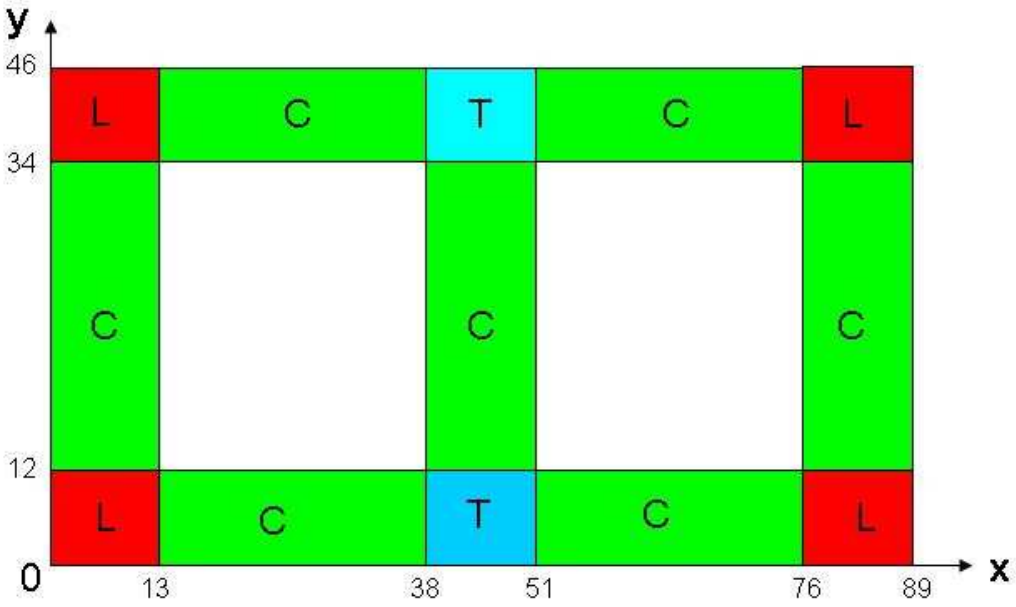


Figure 4.2: Manual Segmentation of Robotic field. Reference label of subsequence is determined based on robot location at the corresponding subsequence. Here, "C" corresponds to straight corridor, "L" corresponds to L-junction, and "T" corresponds to T-junction.

4.3.2 k-means clustering

The k-means algorithm is an algorithm to cluster objects based on attributes into k partitions.

$$c = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{4.19}$$

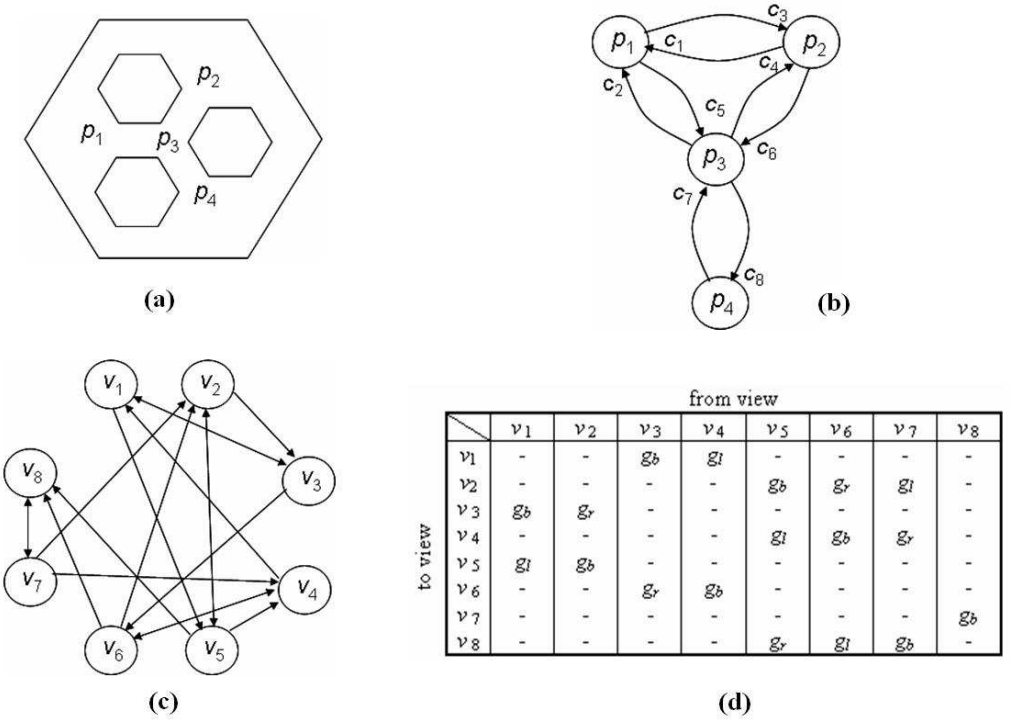
μ_i is centroid or mean of all the points $x_j \in S_i$ In this study, a k-means algorithm is employed for quantization,i.e. given input signal, k-means exhibit the closest mean as the codebook vectors for corresponding input signal.

4.4 Utilizing the Resulting Graph-based Map

Scholkopf and Mallot propose a learning view graph for robot navigation [33][6]. In their proposal, a view-graph map is learned from a sequence of views during exploration. This view-graph provides a discretized version of the environment. The graph-based maps in this chapter are similar to directed place graph in [33]. Figure 4.3. explains how to construct a use the resulting graph for navigation in [33].

In contrast the above mentioned approach, in this thesis a connectivity matrix is introduced directly from the graph-based map, showing the connectivity between two nodes. Hence, it can avoid using dual maps representation. Accordingly the procedures are as follow:

1. Find all possible paths, store nodes and the corresponding robot movements toward the goal
2. Find the shortest path
3. Execute optimum path



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1	-	-	g_b	g_l	-	-	-	-
v_2	-	-	-	-	g_b	g_r	g_l	-
v_3	g_b	g_r	-	-	-	-	-	-
v_4	-	-	-	-	g_l	g_b	g_r	-
v_5	g_l	g_b	-	-	-	-	-	-
v_6	-	-	g_r	g_b	-	-	-	-
v_7	-	-	-	-	-	-	-	g_b
v_8	-	-	-	-	g_r	g_l	g_b	-

Figure 4.3: Construction of a view graphs (adopted from [19]) (a). A simple maze (b). A graph-based map corresponds to (a). Here p_i and c_j are places (junctions) and corridors, respectively (c). Interchange graph of (b), where each node v_i corresponds to one directed connection in the graph-based map (d). Adjacency matrix of the view-graph, with labels indicating the movement leading from one view to another. Here, g_l, g_r and g_b correspond to go left, go right, and go backward, respectively

Chapter 5

Experimental Results on Task Segmentation and Clustering

5.1 Task Segmentation using mnSOM

5.1.1 Difficulty in the standard mnSOM

Figure 5.1.(a) illustrates the resulting map by the standard mnSOM. No information is available on the relation between modules with different colors. It indicates that segmentation based on Figure 5.1.(a) generates 23 classes, which is meaningless. Because of this, our previous studies [23][24] assumed availability of prior information that winner modules corresponding to subsequences in the same class share the same label. However, the prior class information is unavailable in real situation and is unrealistic.

5.1.2 Experiments with a Single Path

mnSOM modules learn internal models of nonlinear dynamics of robot-environment interaction by minimizing mean prediction error of sensory-motor signals at the next time step, given the past sensory-motor signals. After training, the resulting mnSOM provides a label to each module by a procedure in Section 3.5. taking advantage of prior information, that winner modules corresponding to subsequences in the same class share the same label. Given a subsequence, either experienced or novel, one of the modules becomes a winner. The label of the winner module provides task segmentation for each subsequence.

mnSOM is trained using subsequences with varying length. Figure 5.1 illustrates the training result using subsequences with the length 20. White color modules are those which never win the competition for any subsequence.

Figure 5.1: The resulting mnSOM with each color representing a data class. (a)Segmentation by the standard mnSOM. The resulting number of classes is 23, and is too large. (b) Segmentation by mnSOM with the assumption that winner modules corresponding to subsequences in the same class share the same label.

Table 5.1 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, a moderate length of a subsequence might be exists. Taking this characteristic into account, the best length of a subsequence is empirically found based on the criterion of the classification rate.

Figure 5.1 indicate that modules in white color have never become a winner for any subsequence, hence remain unlabeled. For labeling unlabeled modules, a similar labeling method as the conventional SOM is adopted; label an unlabeled module by the label of the subsequence with the minimum Mean Square Error (MSE). Figure 5.2 illustrates the resulting fully labeled task map. Because most of the robot motion is forward movement, many unlabeled modules in Figure 5.1 are labeled by "F" in Figure 5.2. The resulting task map is then evaluated using a novel dataset. Figure 5.3 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.

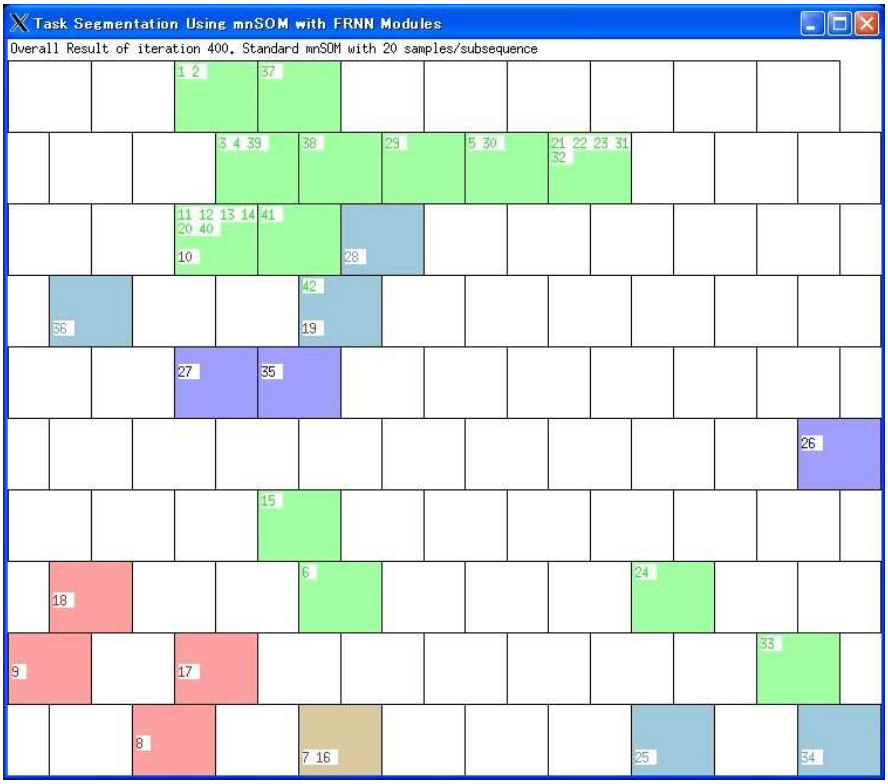


Figure 5.2: Resulting mnSOM trained by subsequences of the length 20 in a single path. The numbers in each module shown in the winner modules represent subsequences which become a winner at the corresponding module.

Table 5.1: 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

Figure 5.3: Fully labeled task map based on a single path composed of subsequences with the length 20.

Figure 5.4: Test result of the task map based on a single path composed of subsequences with the length 20.

Table 5.2: 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

5.1.3 Experiments with Multiple Paths

For better generalization, a task map is proposed to build based on multiple paths. In this study, four paths obtained from the same environment but with slightly different routes are used. The detailed procedure is:

- Generate four paths by moving a Khepera II robot in the forward direction twice as in Figure 3.4, and in the reverse direction twice.
- 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, the majority voting is adopted. When it does not provide a solution, decide the label based on rules in Table 5.2. In case of more than 2 different labels, repeatedly apply rules in Table 5.2 starting from the first two labels. If "X" in Table 5.2 occurs, the corresponding module is left unlabeled, but it is considered to be unlikely. Figure 5.4 illustrates the resulting task map. Again, not all modules become a winner. Hence, it is necessary to assign labels to those modules.

A labeling method as in a single path is assigned to labels the unlabeled modules. Figure 5.5 shows the resulting fully labeled task map based on multiple paths. Each module is an expert representing the nonlinear dynamics of the corresponding subsequences.

Task Map Comparisons: with or without temporal continuity?

Figure 5.6 illustrates the resulting mnSOM of single path by three threshold methods in Subsection 3.6. After labeling the unlabeled modules, test these

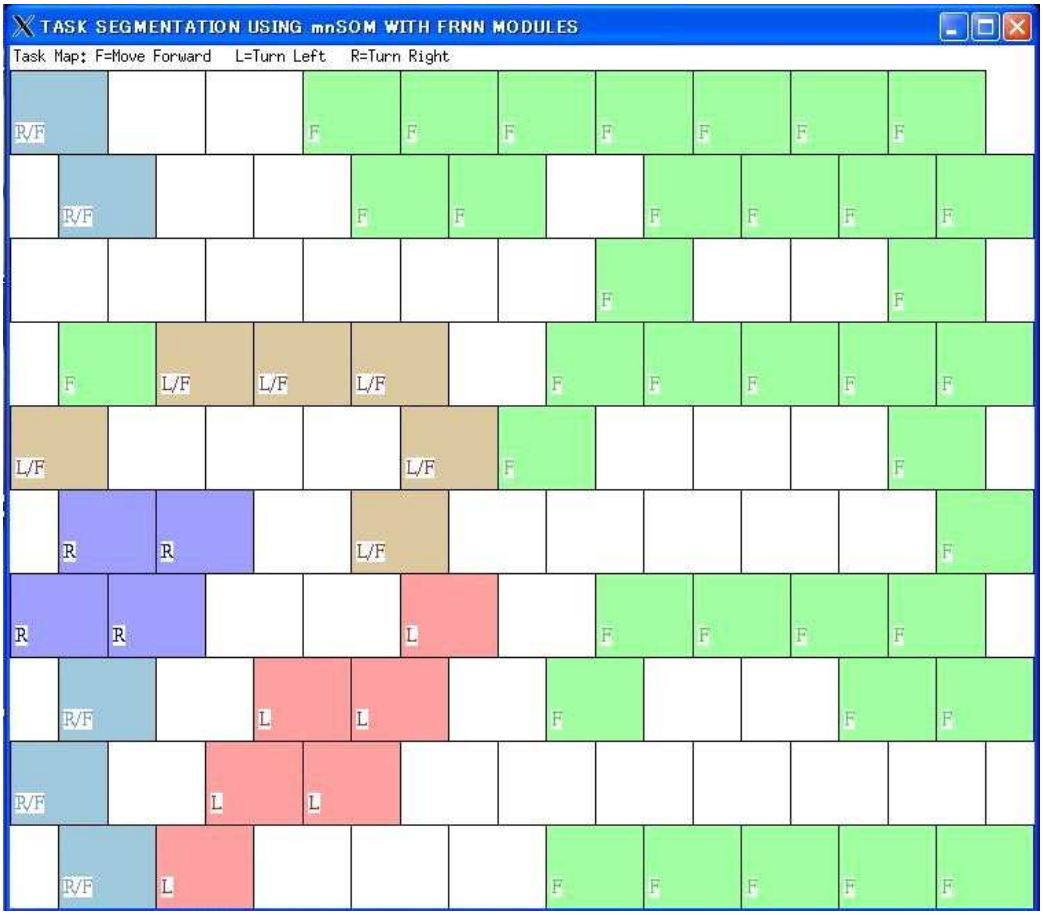


Figure 5.5: Resulting task map based on multiple paths.

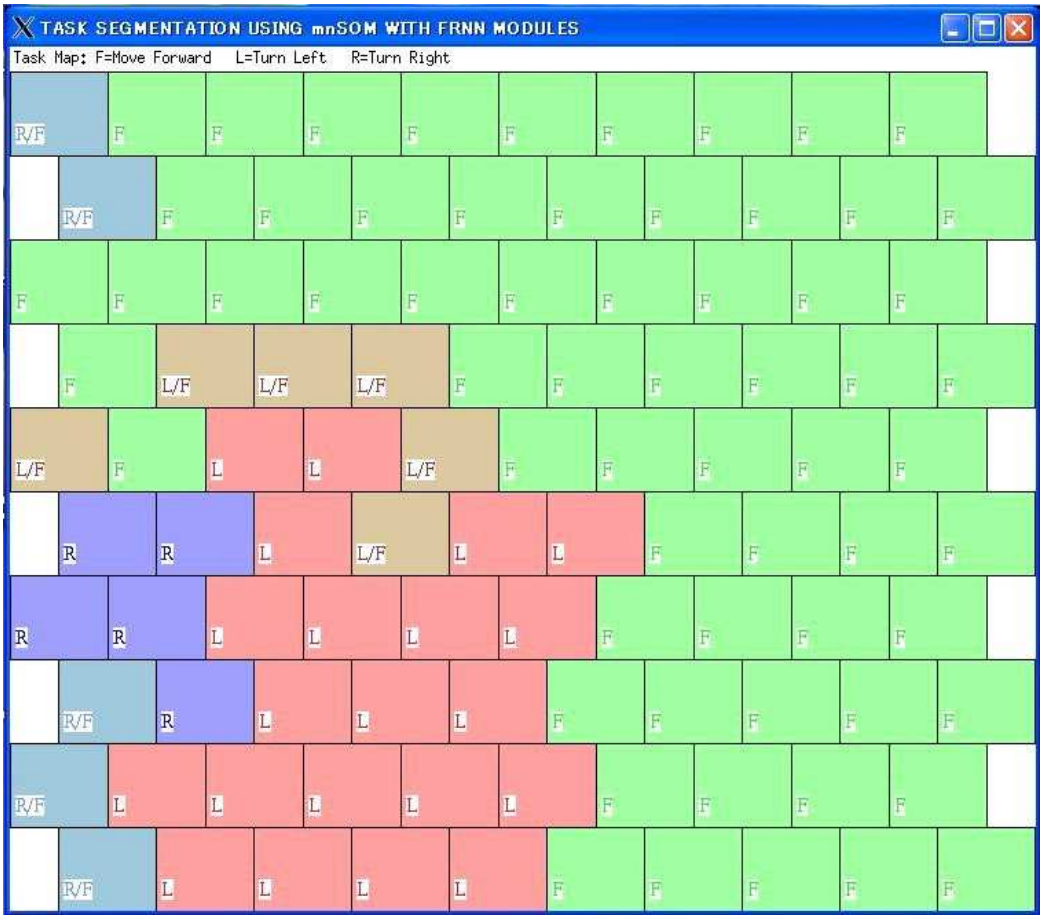


Figure 5.6: Fully Labeled Task Map based on multiple paths.

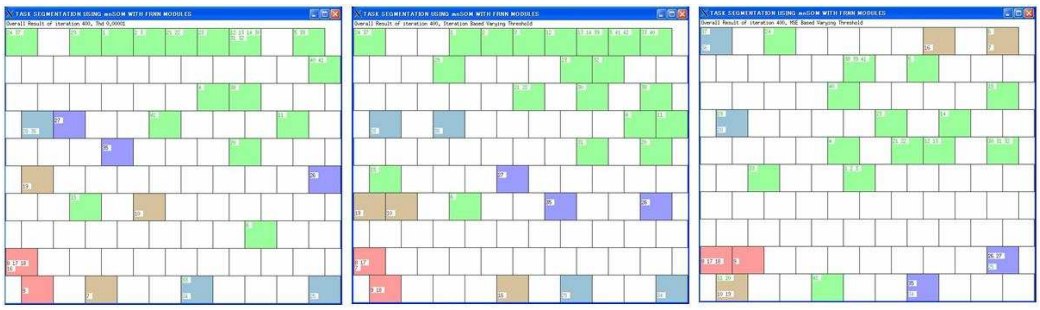


Figure 5.7: Resulting mnSOM for various Threshold in Section 3.6. (a) Constant Threshold. (b) Time Varying MSE Threshold. (c)Proportional MSE Threshold.)

three task maps. Time varying MSE threshold gives the best result with no misclassification during test. Further, elaborate mnSOM training of multiple path by using time varying MSE threshold. Figure 5.7 illustrates the resulting fully labeled task map based on multiple paths as in [23].

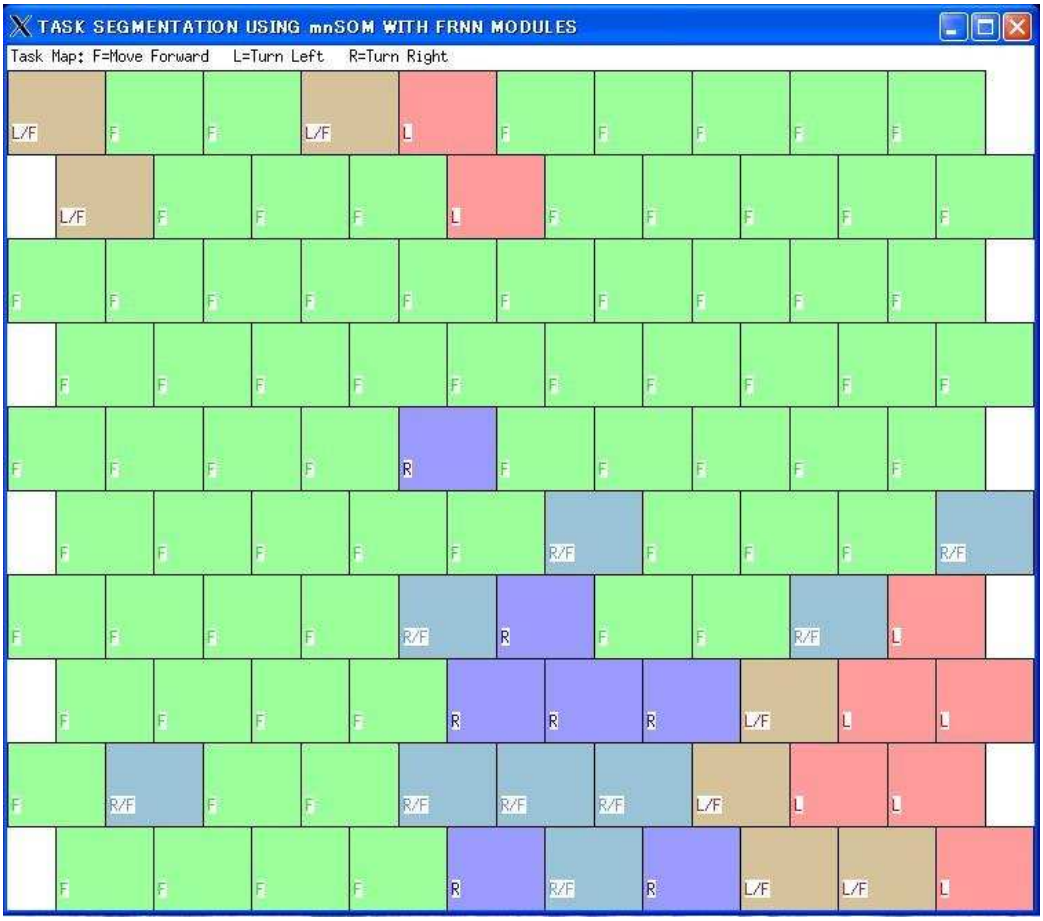


Figure 5.8: Fully labeled task map based on multiple paths for mnSOM with temporal continuity.

Tables 5.3 and 5.4 provide within- and between- class distances using approach without temporal continuity and with temporal continuity, respectively. The tables indicate that without temporal continuity is superior to with temporal continuity. Without using temporal continuity produces 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 its superiority.

Table 5.3: Between- and within- class distance matrix corresponding to Figure 5.5

	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 5.4: Between- and within- class distance matrix corresponding to Figure 5.7

	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					

Figure 5.9: Test result for a novel dataset.

Table 5.5: Classification and segmentation performance of the resulting task map

NS*stands for length of a subsequence. † stands for the best result using temporal continuity

NS*	The number of misclassifications					Correct Segmentation rate (%)	
	Datasets					training	novel
	1	2	3	4	novel	datasets	dataset
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

A novel path is used to evaluate the resulting task map as shows in Figure 5.8. Each module in Figure 5.8 represents subsequence numbers for which the corresponding module becomes a winner. Comparison between Figure 5.8 and Figure 5.4 indicates that 11 winner modules out of 27 winner modules for a novel path (40.74%) have not become a winner during training. It is interpreted that mnSOM takes advantage of its module interpolation capability to find an appropriate expert module for a novel subsequence. Table 5.5 summarize the overall performance for training and test. As before, subsequences with the length 20 is the best.

5.2 Comparative Studies

5.2.1 Task Segmentation using SOM

A SOM with 100 units is used for task segmentation using the same dataset as in the previous section. As the inputs are 8 IR sensor values and 2 motor commands. Figure 5.9. depicts the resulting task map. The resulting task map shows very clear border among tasks. A novel dataset then used to evaluate the resulting map. As the result, 107 samples are misclassified, which is equal to have 87.3% correct classification rate. This result indicates the superiority of mnSOM over SOM for task segmentation.

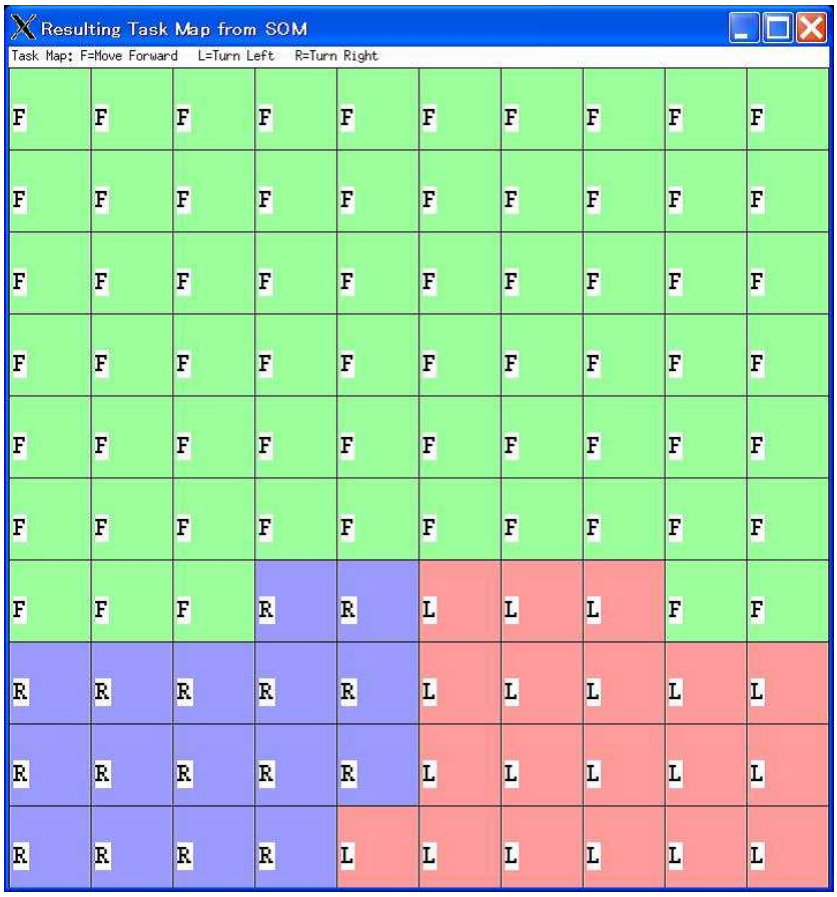


Figure 5.10: A Task Map generated using SOM.

Table 5.6: Parameters in Mixture of Recurrent Experts (MRE)

#x stands for the number of x.

MRE	
learning rate of gate internal state (ϵ_g)	0.07
momentum of gate internal state(η_g)	0.02
scaling factor(σ)	5
# iterations	1000
# BPTT iterations for each MRE iteration	30
MRE Module: FRNN	
# external input units	10
# visible (output) units	10
# hidden units	27
learning rate (ϵ)	0.002

5.2.2 Task Segmentatiom using Mixtures of Recurrent Experts

A mixtures of recurrent experts with 5 RNN modules is employed for this purpose. The same modules architecture are employed in MRE. Detail parameters are in Table 5.5. Figure 5.10 shows training result at iteration 1000. Although MSE are small enough, a certain module (module 5) became a winner for the whole data, hence segmentation is unsuccessfull. It is difficult to find proper parameters for successful learning. I have tried to vary scaling factor from 0.5 upto 10 with unsuccessful results. Another problem might come from the use of fully connected recurrent network which has high degree of freedom, hence one module is enough to handle the whole dataset.

5.3 Task Segmentation using Clustering Methods

This section explains experimental results of segmentation using clustering methods based on the distance between modules in the resulting mnSOM, without using prior information that winner modules corresponding to subsequences in the same class share the same label. The experiments are done using simple robotic field ("robotic field 1") and more complex robotic field("robotic field 2"). For convenient, segmentation results for a novel data

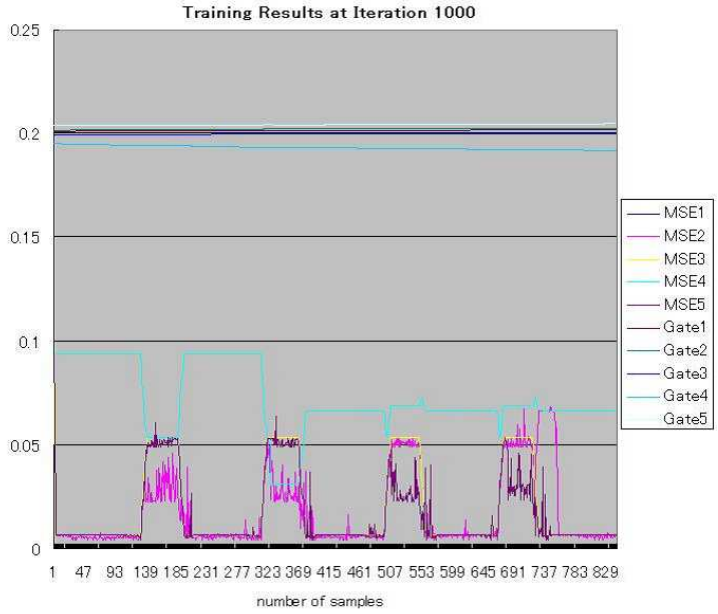


Figure 5.11: Training result of a MRE at iteration 1000

set for robotic field 1 and robotic field 2 are redrawn in Figure 5.11. It is to be noted that the result should not be regarded as a candidate for performance comparison, because it uses unrealistic prior information which is not available in real situation. The result, therefore, should be regarded as the upper bound for segmentation performance.

Figure 5.12 illustrates the resulting segmentation of a novel dataset by hierarchical clustering for robotic field 1 and robotic field 2. The task maps in Figure 5.12 are similar to those by mnSOM in Figure 5.11 to some extent.

Figures 5.13 and 5.14 indicate that proper value of τ shifts some winner modules corresponding to adjacent subsequences (e.g. subsequence 16 and 17 in Figure 8) into the same cluster, and changes cluster boundary.

Table 5.6 gives summary of segmentation performance by various clustering methods in addition to the upper bound for the segmentation performance. It is the correct segmentation rate by mnSOM using prior information. Since this prior information is unavailable in real situation, this should be regarded as the upper bound for the segmentation performance, not as a candidate for performance comparison.

In clustering with spatio-temporal contiguity, the performance of clustering depends on the time constant parameter, τ, in Eq. (9). τ=0 corresponds to clustering with spatial contiguity and positive values of τ correspond to clustering with spatio-temporal contiguity. Table 1 indicates that the per-

(a) (b)

Figure 5.12: Resulting labels for novel subsequences based on mnSOM (a) for robotic field 1, (b) for robotic field 2.

(a) (b)

Figure 5.13: The Resulting Segmentation by Hierarchical Clustering: (a) for robotic field 1, (b) for robotic field 2.

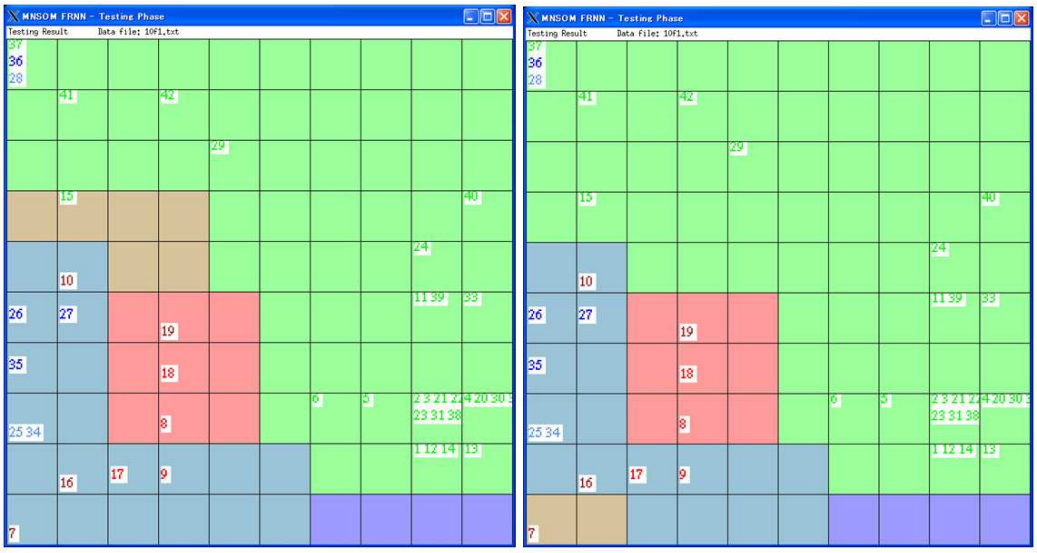


Figure 5.14: Resulting Segmentation by Clustering with Spatio-temporal Contiguity for Robotic Field 1, (a) τ=2, (b) τ=7

Figure 5.15: Resulting Segmentation by Clustering with Spatio-temporal Contiguity for Robotic Field 2, (a) τ=2, (b) τ=19. Subsequences 16 and 17 (circled) which are lied on separated cluster in (a) became on one cluster in (b)

formance is the best at τ=7 for the robotic field 1, while the performance is the best at τ=19 for robotic field 2. In robotic field 1, the performance of the hierarchical clustering is superior to that of clustering with spatio-temporal contiguity. In robotic field 2, the performance of clustering with spatio-temporal contiguity is superior to that of the hierarchical clustering. The reason for this is left for future study.

Table 5.7: Correct Segmentation rate (%) by various Clustering Methods. "upper bound" stands for the correct segmentation rate by mnSOM with prior information. "Tr1", "Tr2", "Tr3", "Tr4" stand for training dataset 1, 2, 3 and 4, respectively. "Ave" stands for the average over 4 datasets. "Novel" stands for novel dataset.

Robotic Field	Data-set	upper bound	Hierar-chical	Spatio-temporal contiguity					
				$\tau \approx 0$	τ=2	τ=7	τ=11	τ=15	τ=19
1	Tr1	94.4	85.71	86.9	86.9	88.1	78.6	67.9	67.9
	Tr2	96.4	85.71	82.1	82.1	84.5	67.9	66.7	52.4
	Tr3	94.0	91.67	78.6	78.6	83.3	71.4	75.0	54.8
	Tr4	100	90.48	80.9	80.9	83.3	63.1	65.5	53.6
	Ave	96.2	88.4	82.1	82.1	84.8	70.3	68.8	57.1
	Novel	94.0	92.9	83.3	83.3	86.9	82.1	70.2	67.9
2	Tr1	97.6	88.7	86.3	86.3	94.4	91.1	91.1	93.6
	Tr2	96.0	88.7	83.1	83.1	86.3	86.3	86.3	91.1
	Tr3	99.2	85.5	91.1	91.1	92.3	92.7	92.7	90.3
	Tr4	98.4	91.1	87.1	87.1	89.5	89.5	89.5	89.5
	Ave	97.8	88.5	86.9	86.9	90.6	89.9	89.9	91.1
	Novel	95.2	92.7	80.6	80.6	87.9	87.9	87.9	93.6

5.4 Conclusions and Discussions

Subsequences with varying lengths are explored. 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, it is 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.

Introduction of temporal continuity to the winner modules is also proposed. The results, however, have less performance than without introducing temporal continuity.

It is possible to do task segmentation using SOM, but mnSOM shows superiority over SOM for current application. Using provided data in this study, MRE learning is unsatisfactory due to difficulty to find its proper parameters. The problem also arise from the using of fully connected recurrent network as MRE's module, since a certain module became a winner for the whole data, hence segmentation is unsuccessful.

One might consider that proposed study is symbolic, i.e., discretized, control, since modules are proposed and one of them becomes a winner for a subsequence. However, in contrast to the so-called symbolic approach, module interpolation is possible in present approach. It can be interpreted that mnSOM takes advantage of its module interpolation capability to find an appropriate expert module for a new subsequence.

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, 10x10 modules are used in mnSOM.

A PC with Pentium 4 (3.2GHz, 1GB RAM) is used. mnSOM training based on a single path takes 8.5 hours 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, which is small enough for real application.

Segmentation based only on mnSOM assumes the availability of prior information that winner modules corresponding to subsequences in the same class share the same label. Since this prior information is not available in real situation, segmentation using clustering methods based on the distance between modules in the resulting mnSOM is proposed. The resulting segmentation by mnSOM is then regarded as the upper bound for the segmentation performance, not as a candidate for performance comparison.

Firstly, the conventional hierarchical clustering is proposed. This supposes that the distances between any pairs of modules are provided with precision, but this is not the case in mnSOM. Secondly, a clustering method based on the distance between only the spatially adjacent modules with modification by their temporal contiguity is proposed.

In the robotic field 1, the segmentation performance by the hierarchical clustering is very close to the upper bound for novel data. In the robotic field 2, the segmentation performance by clustering with the spatio-temporal

contiguity is very close to the upper bound for novel data. Therefore, the proposed methods demonstrated their effectiveness of segmentation. However, segmentation performance for training data is significantly lower than the upper bound. The improvement of this is left for future study.

Chapter 6

Experimental Results on the Formation of Graph based Maps

6.1 Environment Segmentation

Figure 6.1 illustrates robotic path in a training dataset. Numbers in figure are the starting and ending subsequence number of T- or L-junctions. Location of a number corresponds to robot location in the robotic field. Label of a subsequence is determined according to current robot location (refers to Figure 4.2). This dataset is used for HMMs and mnSOM training. Experimental results presented in this section is based on a novel dataset depicted in Figure 6.2. Figure 6.3 illustrates the resulting environment map in mnSOM module using the same procedure as in Chapter 3. It is to be note that the color of a module contains different information comparing to that in Chapter 5. Legend in the lower part of the graph explains meaning of each color of modules. Given a subsequence of data, known data or a novel one, mnSOM will exhibit prediction of the type of environment experienced by the robot in the current subsequence. Given a dataset, mnSOM will generate sequence of module labels from which the current data originally came.

6.2 Environment Recognition using mnSOM-HMMs

Using the resulting mnSOM in Figure 6.3, given a subsequence of data, known data or a novel one, mnSOM provides estimation of the type of the environ-

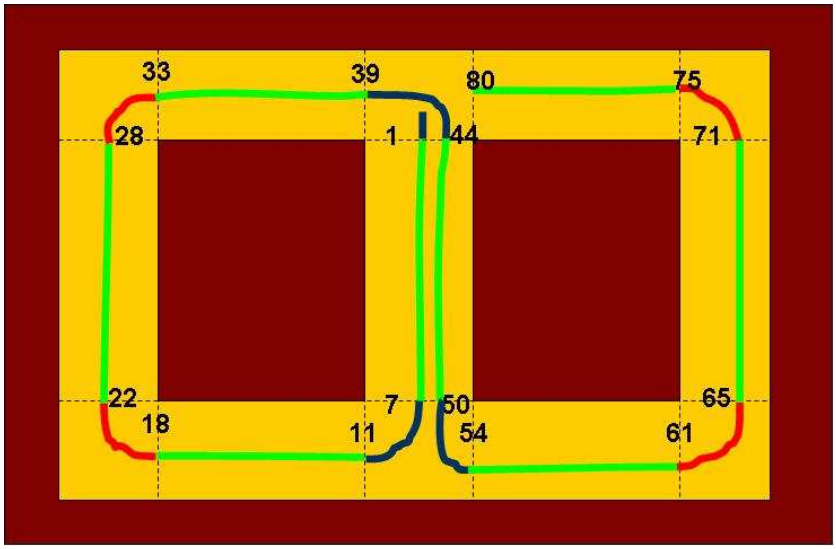


Figure 6.1: Robotic path in a training dataset.

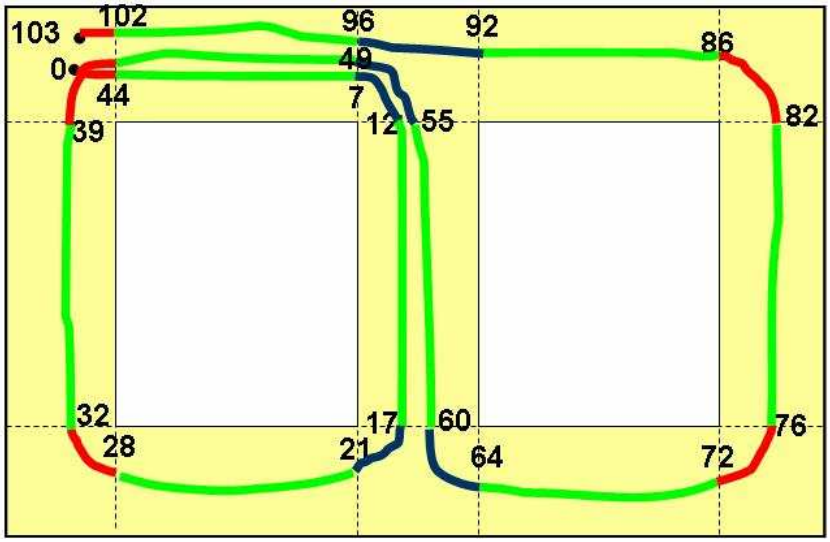


Figure 6.2: Robotic path in a novel dataset

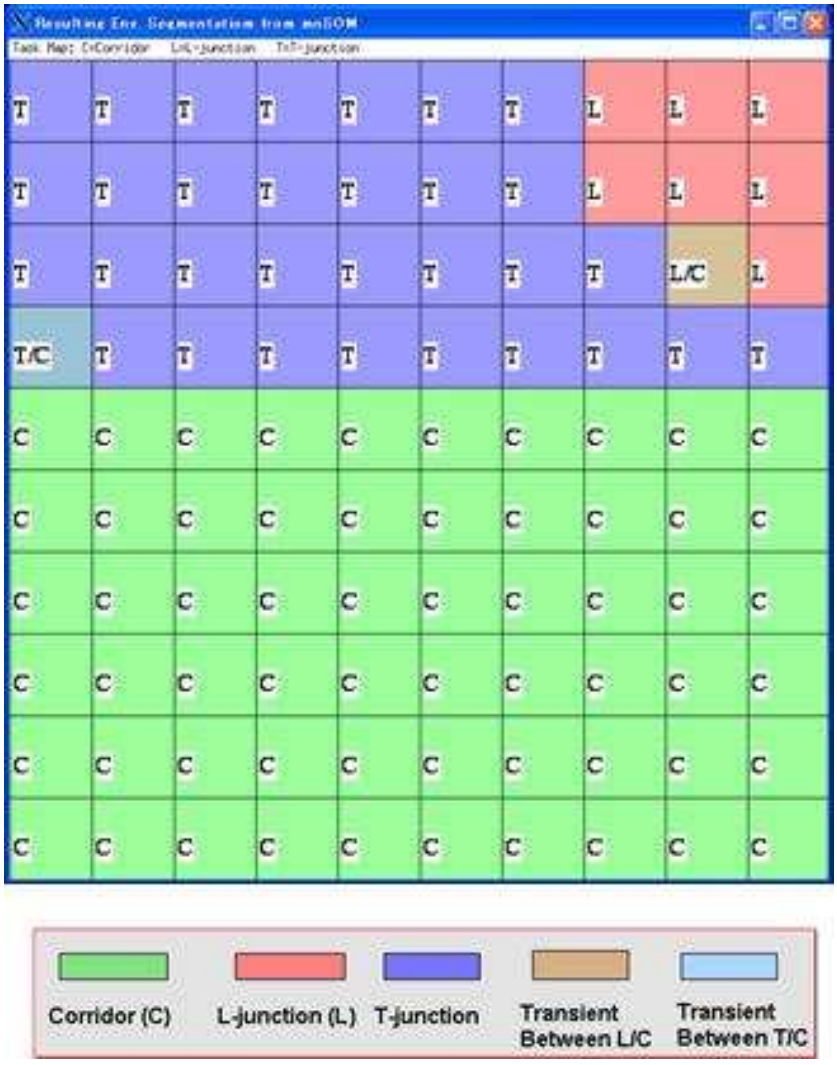


Figure 6.3: Resulting Map from mnSOM.

Table 6.1: Parameters of mnSOM-HMMs

	Parameter	Value
T	the length of the observed sequence	80
N	the number of states in the model	3
M	the number of observation symbols per state	5
V	discrete set of possible symbol observation	{"C", "L", "T", "L/C", "T/C"}
π	probability distribution of initial state	{1, 0, 0}

ment experienced by the robot corresponding to current subsequence. Given a dataset, mnSOM provides sequence of labels. Parameters in mnSOM-HMMs are given in Table 6.1 and Table 6.2.

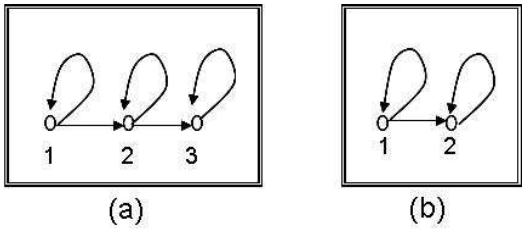


Figure 6.4: Left-right type HMM employed in the current study (a) with 3 states (b) with 2 states

Three HMMs are trained, namely HMM for corridor, HMM for L-junction and HMM for T-junction. An important issue is how to determine the starting and end points of each HMM. To solve this problem, a normalized likelihood in Eq. (6.1) is calculated by taking into consideration of the length of the period, k, of each HMM.

$$P_{norm_k} = (\prod_{i=1}^{i=k} P(O_i|\lambda_i))^{1/k} \tag{6.1}$$

In Figure 6.5, normalized likelihood value begins to decrease after time step 6. It suggests that the HMM corresponding to corridor with the length of

Table 6.2: Estimated Parameters in the mnSOM-HMMs

Parameter	HMM	Values				
A	HMM for corridor	0.7675	0.2325	0.0		
		0.0	0.8295	0.1705		
		0.0	0.0	1.0		
	HMM for L-junction	0.2732	0.7268	0.0		
		0.0	0.6739	0.3261		
		0.0	0.0	1.0		
	HMM for T-junction	0.848	0.152	0.0		
		0.0	0.0004	0.9996		
		0.0	0.0	1.0		
B		"C"	"L"	"T"	"L/C"	"T/C"
	HMM for corridor	0.7822	0.0187	0.0181	0.1064	0.0746
		0.947	0.0472	0.0	0.004	0.0017
		0.0001	0.0002	0.4947	0.0	0.505
	HMM for L-junction	0.4817	0.3995	0.0	0.0628	0.0559
		0.0768	0.4958	0.2002	0.2272	0.0
		0.8794	0.0	0.0	0.1206	0.000
	HMM for T-junction	0.1443	0.0	0.4945	0.0	0.3612
		0.0	0.0	0.8897	0.0	0.1103
		0.5184	0.0	0.0008	0.1482	0.3326

6 is the best from the viewpoint of likelihood. The subsequent decisions are made in the same way by starting from the following time step of the end point of the previous HMM.

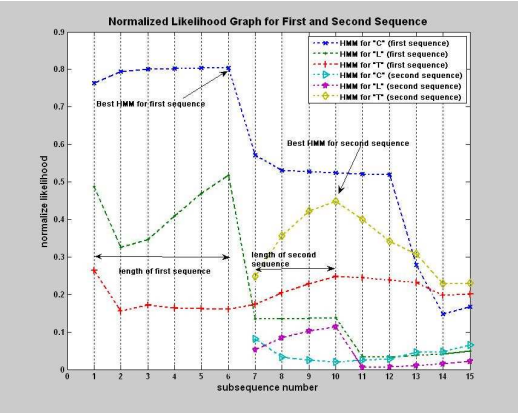


Figure 6.5: Graph of normalized likelihood. (Here, normalized likelihood values are available only at markers on the curve)

Table 6.3 gives the resulting recognitions. The fourth column of this table provides the middle of each segment. In case of T and L labels, they correspond to the center of junctions.

The correct segmentation rate is calculated from comparison of the length of label sequences in Table 6.3 to the original length of sequences in Figure 6.2 The correct segmentation rate is 89.32 %. However, mis-recognition occurs as shown in Figure 6.6: a T-junction is mis-recognized as a combination of T-junction and L-junction.

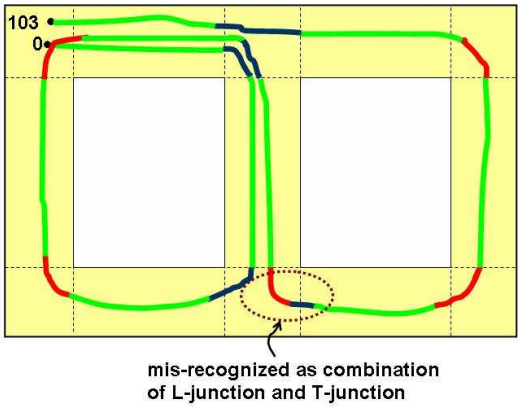


Figure 6.6: Junctions recognition using mnSOM-HMM for a Novel Data

Table 6.3: Summary of mnSOM-HMMs Recognition Results for a Novel Dataset

starting subsequence number	end subsequence number	best HMM	subsequence at the middle of sequence
1	7	C	3.5
8	11	T	9.5
12	17	C	14.5
18	20	T	19
21	26	C	23
27	31	L	29
32	38	C	35
39	43	L	41
44	48	C	46
49	54	T	51.5
55	60	C	57.5
61	63	L	62
64	65	T	64.5
66	71	C	68.5
72	76	L	74
77	82	C	79.5
83	84	L	83.5
85	92	C	88.5
93	97	L	94.5
98	103	C	100.5

Table 6.4: Parameters in k-means-HMMs

Parameter	Value
T	1600
N	2 or 3
M	5
V	{"1", "2", "3", "4", "5"}
π	{1, 0, 0}

6.3 Environment Recognition using k-means-HMM

Using k-means quantization, a set of 5 codebook vectors with 10-dimension are created. Ten dimension corresponds to 8 IR sensors and 2 motor commands. Accordingly, at every time step, one of 5 codebook vectors is observed. These code vectors are used for estimating parameters in HMMs. Figure 6.7 shows comparison of log likelihood of HMMs with various number of codebook vectors.

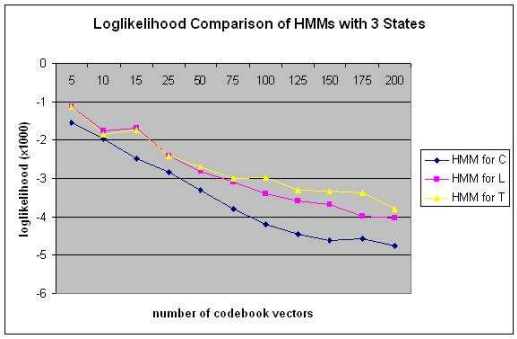


Figure 6.7: Likelihood Comparison of HMMs with Quantized Raw Input Data using k-means Clustering with Various Numbers of Codebook vectors.

Three HMMs are trained, namely HMM for corridor, HMM for L-junction and HMM for T-junction. Two cases are compared in this subsection, junctions recognition using HMMs with 3 states and that with 2 states. Tables 6.4 shows parameters in a k-means-HMMs

k-means-HMMs with 3 States

Tables 6.5 shows estimated parameters of k-means-HMMs with 3 states.

Table 6.5: Estimated Parameters of k-means-HMMs with 3 States

Parameter	HMM	Values				
A	HMM for	0.9749	0.0251	0.0		
	corridor	0.0	0.9158	0.0842		
		0.0	0.0	1.0		
	HMM for	0.9933	0.0067	0.0		
	L-junction	0.0	0.9094	0.0906		
		0.0	0.0	1.0		
	HMM for	0.9225	0.0775	0.0		
	T-junction	0.0	0.9921	0.0079		
		0.0	0.0	1.0		
B		"1"	"2"	"3"	"4"	"5"
	HMM for	0.0084	0.9404	0.0487	0.0025	0.0000
	corridor	0.1907	0.0355	0.7618	0.012	0.0000
		0.6384	0.0013	0.0042	0.356	0.0000
	HMM for	0.0167	0.5347	0.4486	0.0000	0.0000
	L-junction	0.9500	0.0000	0.0500	0.0000	0.0000
		0.000	0.0654	0.9346	0.0000	0.0000
	HMM for	0.9141	0.0775	0.0084	0.0000	0.0000
	T-junction	0.0093	0.0000	0.9907	0.0000	0.0000
		0.1680	0.6348	0.0198	0.0000	0.1774

Table 6.6: Recognition Results Summary of k-means-HMMs with 3 States

starting sample number	end sample number	best HMM	the middle of segment (sample)	the middle of segment (subsequence)
1	17	L	9	1
18	148	C	83	6
149	238	T	152	7
239	345	C	193	9
346	448	T	392	18
449	578	C	513	25
579	657	L	621	31
658	779	C	718	35
780	894	L	837	41
895	1015	C	955	47
1016	1099	T	1057	52
1100	1207	C	1153	57
1208	1274	T	1241	62
1275	1384	C	1329	66
1385	1490	L	1437	74
1491	1658	C	1574	78
1659	1702	L	1680	84
1703	1877	C	1790	89
1878	1962	T	1919	95
1963	2060	C	2010	100

While recognition results of a novel data is depicted in Figure 6.8 and Table 6.6. The fourth column of Table 6.6 provides the middle of each segment in term of sample number while fifth column provides the middle of each segment in the term of equivalent number of subsequence. Suppose that the middle of each T or L sequence corresponds to an edge of a graph, and each edge connected by sequence of "C" labels, a graph-based map can be generated if a pair of edges are connected by a sequence "C." Figure 6.8 depicts the resulting map, which is similar to some extent to the original path in Figure 6.2. The correct segmentation rate is calculated from comparison of the length of label sequences to the original length of sequences in Figure 6.2. The correct segmentation rate is 83.74 %. In addition, no mis-recognition occurs.

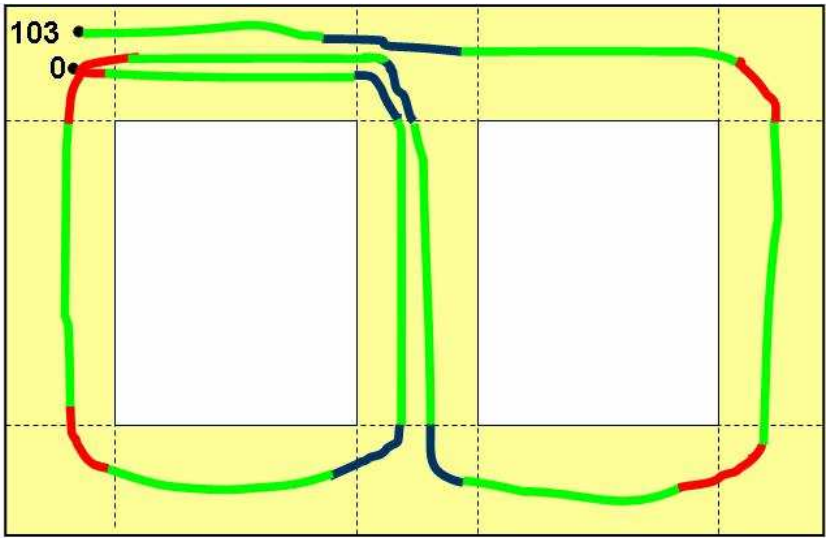


Figure 6.8: Junction recognition using k-means-HMMs with 3 states for a novel dataset

k-means-HMMs with 2 States

Tables 6.7 shows estimated parameters of k-means-HMMs with 2 states. While recognition results of a novel data is depicted in Table 6.8 and Figure 6.9. The correct segmentation rate is 81.26 %. However, a mis-recognition occurs here, i.e. a T-junction is recognized as combination of L-junction and T-junction.

Table 6.7: Estimated Parameters of k-means-HMMs with 2 States

Parameter	HMM	Values				
A	HMM for corridor	0.9871	0.0129			
		0.0	1.0			
	HMM for L-junction	0.8581	0.1419			
		0.0	1.0			
	HMM for T-junction	0.9427	0.0573			
		0.0	1.0			
B		"1"	"2"	"3"	"4"	"5"
	HMM for corridor	0.6842	0.0059	0.2269	0.0533	0.0298
		0.0153	0.0609	0.0	0.485	0.4388
	HMM for L-junction	0.8919	0.0164	0.0000	0.0491	0.0426
		0.0217	0.6625	0.0000	0.3158	0.000
	HMM for T-junction	0.9676	0.0192	0.0000	0.0132	0.0000
		0.0376	0.8683	0.0000	0.0941	0.0000

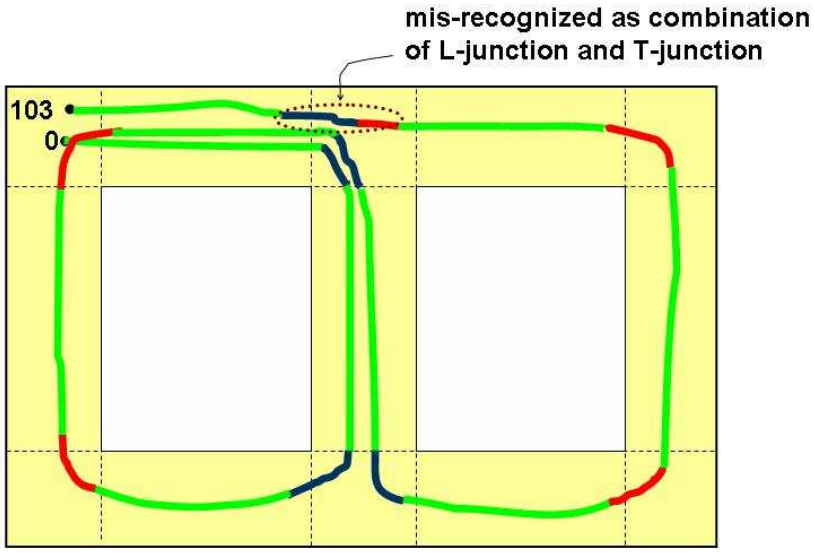


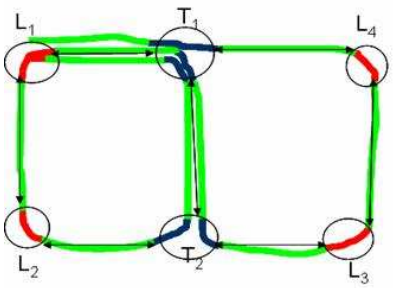
Figure 6.9: Junction recognition using k-means-HMMs with 2 states for a novel dataset

Table 6.8: Recognition Results Summary of k-means-HMMs with 2 States

starting sample number	end sample number	best HMM	the middle of segment (sample)	the middle of segment (subsequence)
1	162	C	81	4
163	238	T	200	10
239	345	C	291	15
346	415	T	380	19
416	586	C	500	26
587	655	L	620	31
656	779	C	717	36
780	907	L	843	43
908	1030	C	968	49
1031	1100	T	1065	54
1101	1207	C	1153	58
1208	1271	T	1239	62
1272	1431	C	1351	68
1432	1490	L	1460	73
1491	1658	C	1574	79
1659	1739	L	1698	85
1740	1890	C	1814	91
1891	1910	L	1900	95
1911	1962	T	1936	97
1963	2060	C	2011	101

6.4 Formation of a Graph-based Map and Case Study

In the previous section, a graph-based map has been derived. This section provides a method to use the resulting graph for mobile robot navigation. The basic idea has been explained in chapter 4.



(a)

	L_1	L_2	L_3	L_4	T_1	T_2
L_1	1	1	0	0	1	0
L_2	1	1	0	0	0	1
L_3	0	0	1	1	0	1
L_4	0	0	1	1	1	0
T_1	1	0	0	1	1	1
T_2	0	1	1	0	1	1

(b)

Figure 6.10: Constructing the graph-based maps from the resulting sequence of HMMs (a). The resulting environment recognition and its corresponding graph-based map. Here, L_i corresponds to L-junction and T_j corresponds to T-junction (b). Connectivity matrix. '1' corresponds to connected, '0' corresponds to not connected

Figure 6.10.(a) depicts the resulting graph-based map from the corresponding environment recognition. Two arcs are shown here to represent bi-directional movement of a mobile robot between 2 nodes. Figure 6.10(b) shows the corresponding connectivity matrix.

Suppose that an obstacle is present in the corridor as in Figure 6.11.(a). The robot is at the start position and seeking the goal with the red circle on the robotic field. Two lines shown in this figure are the corresponding possible robotic paths toward the goal.

Following the procedure in Chapter 4:

1. Possible paths

   Since the goal is between L_3 and L_4, the following paths are possible

   - $L_1 - T_1 - T_2 - L_3 - L_4$, robot movement at a node are "straight forward - turn right - turn left - turn left"
   - $L_1 - L_2 - T_2 - L_3 - L_4$, robot movement at a node are "turn right - turn left - straight forward - turn left"

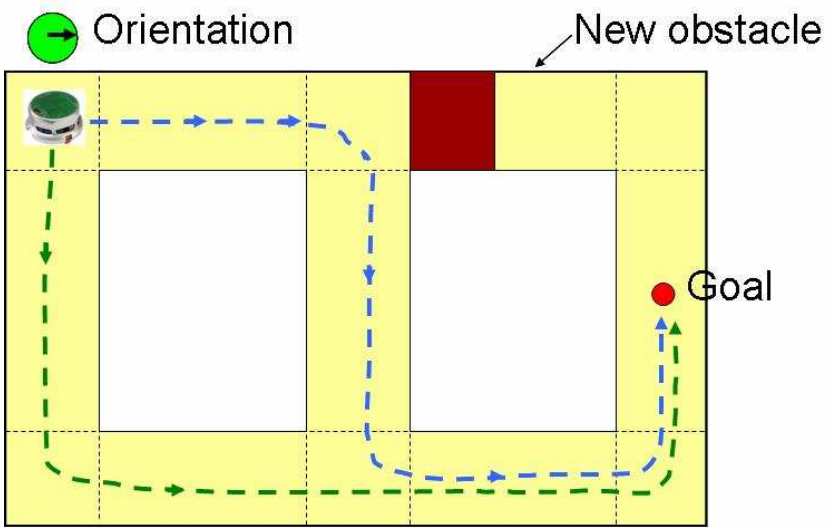


Figure 6.11: Goal Seeking Problem

2. Find the shortest path

   It is assumed that $L_1 - T_1 - T_2 - L_3 - L_4$ has the shortest path toward the goal

3. Execute optimum path

Figure 6.12 shows robot movements in Webots simulation.

6.5 Conclusions and Discussions

Ambiguous interpretation of sensory-motor signals due mainly to noise and fluctuation often makes deterministic approach in mobile robots unsatisfactory. In this paper, a stochastic approach based-on estimation of Hidden Markov Models (HMMs) was proposed to recognize environment of a mobile robot. From this recognition a graph-based map is formed. Graph-based maps are important in decreasing the computational cost.

Two methods are proposed in this paper. The former is to estimate HMMs based on a sequence of labels obtained by modular network SOM (mnSOM). The resulting maps are quite similar to the actual map and the correct segmentation rate is good enough (89.32 % for a novel dataset), however a label mis-recognition occurs for a novel dataset.

The latter is to estimate HMMs based on quantized sensory-motor signals as observed symbols. The resulting sequence of HMMs provide a sequence

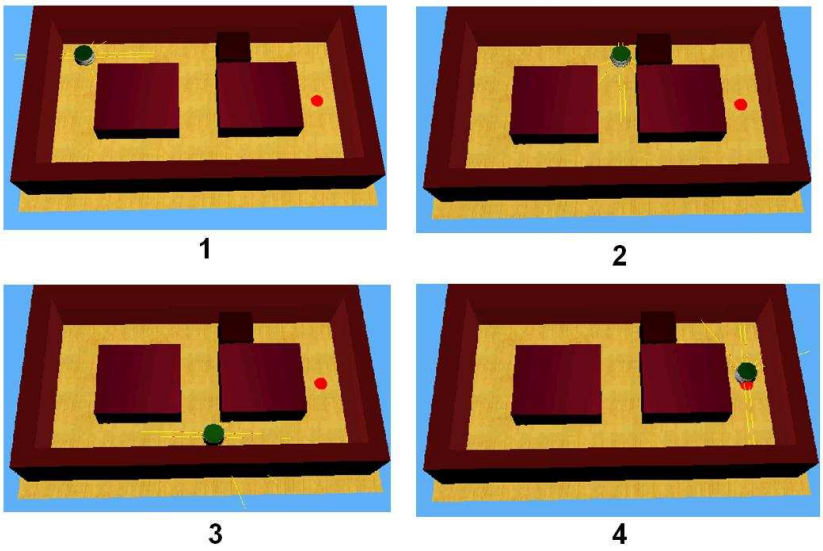


Figure 6.12: Webots simulation of goal seeking. Numbers shown in the figures are the corresponding order of robot movements to reach the goal

of labels describing segmentation of the environment, i.e., L-junction, T-junction, and corridor. It is straightforward to generate a graph-based based on them. A k-means-HMMs with 3 states gives better segmentation rate, 83.74 % for a novel dataset, than a k-means-HMMs with 2 states (81.26 % for a novel dataset). In addition, the model with 3-state has no label misrecognition. This result also suggests that k-means-HMMs with 3 states is more reliable than mnSOM-HMMs with 3 states.

The resulting graph-based map obtained from HMMs also contributes to goal seeking. Simulation results indicate that the proposed method can perform goal seeking efficiently, since it is not necessary to introduce new map for changing environment.

Chapter 7

Conclusions and Discussions

7.1 Conclusions

To segment the world based on sensory-motor signals using mnSOM, and to develop a graph-based map, I proposed the followings:

1. Segmentation by clustering based on the resulting mnSOM

   Task segmentation by mnSOM alone in mobile robots requires unrealistic prior information, hence the results should be regarded as the upper bound for the performance of segmentation. To avoid this unrealistic prior information, clustering methods are applied to the resulting mnSOM. Firstly, I proposed to use the conventional hierarchical clustering. It assumes that the distance between any pairs of modules are provided with precision, but this is not the case in mnSOM. Secondly, I proposed to used a clustering method based on the distance between only the spatially adjacent modules with modification by their temporal contiguity. The experimental results showed that the performance of proposed clustering methods are very close to the upper bound for novel data. Therefore, the proposed methods demonstrated their effectiveness in segmentation.

2. Formation of graph-based maps by Hidden Markov models (HMMs)

   I proposed two approaches for this, the former is to use the resulting labels by mnSOM as observed symbols in HMMs. The latter is to use quantized sensory signals provided by k-means clustering as observed symbols in HMMs. Experimental results indicated that HMMs based on quantized sensory signals by k-means provided good recognitionperformance, since the selection of HMMs was always correct. The result-

ing graph-based map is also effective in goal seeking, because memory and computational cost is much smaller than those in grid-based maps.

7.2 Discussions

The present thesis proposed segmentation using clustering based on the resulting mnSOM. The clustering results using spatio-temporal contiguity provided good performance by proper adjustment of parameters. However, the how to adjust parameter is not clear. Solution to this issue is left for further study.

I proposed two methods for formation of graph-based maps: HMMs based on the resulting mnSOM, and HMMs based on quantized sensory data. The later was more robust than the former. Introduced prior probability to the former for better performance with no avail. How to improve the performance of recognition of the environment is also left for future study.

HMMs in this study worked off-line, restricting its applicability. In application to mobile robots, there has been a strong needs to enhance the ability of mnSOM. Tokunaga [35] recently proposed on-line learning of mnSOM. This algorithm is expected to be more suitable for mobile robots, since it can handle on-line data.

Segmentation of visual data in mobile robots is one of important issues to be explored. Having segmented sequence of images collected during exploration of the environment, a technique such as homing [6][33] can be employed for path planning and goal seeking. Recent proposal on SOM^n, a variant of mnSOM which is used SOM as a function module, is expected to enhance capability of segmentation of image data, since it demonstrated its effectiveness in face recognition [13][9].

Segmentation is, I believe, one of the origins of intelligence, hence has applicability to diverse fields such as marketing, social, politics and engineering field. Segmentation in other fields has not yet been properly studied, and waits for development as new endeavors.

Bibliography

[1] Beevers, R. K., Huang, W. H.: Loop closing in topological maps, Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA 2005), Barcelona, Spain, (2005) 4378-4383

[2] Boada, B.L., Blanco, D., and Moreno, L.: Symbolic place recognition in voronoi-based maps by using hidden markov models, Journal of Intelligent and Robotic Systems, 39, (2004) 173-197

[3] Borenstein, J. and Koren Y.: The vector field histogram - Fast obstacle avoidance for mobile robots,IEEE Transactions on Robotics and Automation, 7 (3), (1991), 278-288

[4] Cyberbotics Ltd.: Webots User Guide, www.cyberbotics.com, (2006)

[5] Duda, R.O., Hart, P.E., and Stork, D.G.: Pattern Classification, Wiley-Interscience, (2001)

[6] Franz, M. O., Scholkopf, B., Mallot, H. A., Bulthoff, H. H.: Learning View Graphs for Robot Navigation, Autonomous Robots 5, (1998) 111-125

[7] 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

[8] 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)

[9] Furukawa, T.: Modular Network SOM and Self-Organizing Homotopy Network as a Foundation for Brain-like Intelligence, WSOM 2007, Bielefeld, Germany, (2007) (Keynote Speech)

[10] Haykin, S.: Neural Networks A Comprehensive Foundation, Prentice Ahll International Inc., 2nd ed, Canada (1999)

[11] http://en.wikipedia.org/wiki/Situated

[12] Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G.: Adaptive Mixtures of Local Experts, Neural Computation, **3** (1991) 79-87

[13] Jiang, J., Zhang, L., and Furukawa, T.: Improving the Generalization of Fisherface by Training Class Selection Using SOM2, Lecture Notes in Computer Science(Edited book of 13th International Conference of Neural Information Processing (ICONIP2006)) , 4233 (2006), 278-285

[14] Kohonen,T.: Self organized formation of topologically correct feature maps, Biological Cybernetics, 43, (1982) 59-69.

[15] Kohonen, T.: Self-Organizing Maps, Springer, 1995

[16] K-Team S.A: Khepera 2 user manual, Switzerland, (2002)

[17] Linaker, F.: Unsupervised On-line Data Reduction for Memorisation and Learning in Mobile Robotics, Ph.D Thesis, Dept. Computer Science, University of Sheffield , UK, 2003

[18] Low, K. H., Leow, W. K., Ang, M. H. Jr,: An Ensemble of Cooperative Extended Kohonen Maps for Complex Robot Motion Tasks, Neural Computation, 17, (2005) 1411-1445

[19] Mallot,H.A, Bulthoff, H.H., Georg, P., Scholkopf, B., Yasuhara, K.: View-based cognitive map learning by an autonomous robot, in Proc. ICANN'95, vol.2., Paris, (1995) 381-386

[20] 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

[21] Mataric, M. J.: Integration of Representation into Goal-driven Behavior-based Robot, IEEE Trans. on Robotic and Automation, 8(3), (1992) 304-312

[22] Mataric, M. J.: A Distributed Model for Mobile Robot Environment-Learning and Navigation, Technical Report: AITR-1228, MIT, (1990)

[23] 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

[24] M. Aziz Muslim, Masumi Ishikawa, and Tetsuo Furukawa: Task Segmentation in a Mobile Robot by mnSOM : A New Approach To Training Expert Modules, Neural Computing and Application (To appear), Springer, (2007)

[25] Newel,A., Simon, H.A., Computer Science as Empirical Inquiry:Symbols and Search, Communication of the ACM, 19, (1976) 113-126

[26] 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

[27] Pariyapong, V., Parnichkun, M.: Ensemble structure of multiple local sensor fusion machine using evolutional pruning technique [an application to heading and rate of turn estimation], IEEE ICIT '02, vol 1., (2002) 421-426

[28] Pawelzik, K., Kohlmorgen, K., and Muller, K. R.: Annealed Competition of Expert for a Segmentation and Classification of Switching Dynamic, Neural Computation, 8(2), pp. 340-356, 1999

[29] Pfeifer, R., Scheier, C.: Understanding Intelligence, MIT Press, 1999

[30] Rabiner,R. L. and Juang, B.H.: An Introduction to Hidden Markov Models, IEEE ASSP Magazine, 3(1), (1986) 4-16

[31] Rabiner,R. L.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proc. IEEE, 77(2), (1989) 257-286

[32] Schapire, R.E.: The strength of Weak Learnability, Machine Learning, 5, (1990), 197-227

[33] Scholkopf, B., Mallot, H. A.:View-based cognitive mapping and path planning, Adaptive Behavior 3(3), (1995) 311-348

[34] 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

[35] Tokunaga, K. and Furukawa, T.: The online algorithm for generation of stability mapping in generalized modular network SOM, Brain-Inspired Information Technology, Kitakyushu, Japan, (2006) 70

[36] 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)

[37] 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.

[38] 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

[39] Remolina, E., Kuipers, B.: Towards a General Theory of Topographical Maps, Artificial Intelligence, 152, (2004) 47-104

[40] Thrun, S., Bucken,A.: Integrating Grid-based and Topological Maps for Mobile Robot Navigations, AAAI, Portland, Oregon, (1996)

[41] Thrun, S., Fox, D. and Burgard, W.: A probabilistic approach to concurrent mapping and localization for mobile robots, Machine Learning, 31, (1998) 29-53

[42] Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Ha"hnel, D., Rosenberg, D., Roy, N., Schulte, J., and Schulz,D.: Probabilistic algorithms and the interactive museum tour-guide robot Minerva, International Journal of Robotics Research, 19(11), (2000) 972-999

[43] Veneri, N.,Messelodi, S., Crespi, B.: A Memory-based Approach to Sensory-Motor Coordination, Machine Vision and Applications, 10, Springer-Verlag, (1998), 269-279

[44] 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

[45] 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

[46] Wolpert, D.M., and Kawato, M.: Multiple paired forward and inverse models for motor control, Neural Networks, Vol.11, (1998) 1317-1329