SOM of SOMs

Tetsuo Furukawa

Department of Brain Science and Engineering,

Graduate School of Life Science and Systems Engineering,

Kyushu Institute of Technology, Kitakyushu, Japan

Acknowledgments

This work was partially supported by a Center of Excellence Program (Center #J19) grant in 2003 to the Department of Brain Science and Engineering of the Kyushu Institute of Technology by the Japanese Ministry of Education, Culture, Sports, Science and Technology. Partial support was also provided by the Japanese Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), 17500193, 2005. The facial images in this paper have been used with permission of Softopia Japan, Research and Development Division, HOIP Laboratory.

Corresponding Author

Department of Brain Science and Engineering, Kyushu Institute of Technology,

2-4 Hibikino, Wakamatsu-ku, Kitakyushu 808-0196, Japan

Tel: +81–93–695–6124

Fax: +81–93–695–6134

E-mail: furukawa@brain.kyutech.ac.jp (T. Furukawa)

SOM of SOMs

Abstract

This paper proposes an extension of the self-organizing map (SOM), in which the mapping objects themselves are self-organizing maps. Thus a "SOM of SOMs" is presented, which we refer to as a SOM^2. A SOM^2 has a hierarchical structure consisting of a single parent SOM and a set of child SOMs. Each child SOM is trained to represent the distribution of a data class in a manifold, while the parent SOM generates a self-organizing map of the group of manifolds modeled by the child SOMs. Thus a SOM^2 is an architecture that organizes a product manifold represented as (child SOM)×(parent SOM). Such a product manifold is called a fiber bundle in terms of the topology. This extension of a SOM is easily generalized to any combination of SOM families, including cases of neural gas (NG) in which, for example, "NG^2(=NG×NG) as an NG of NGs" and "NG×SOM as a SOM of NGs" are possible. Furthermore, a SOM^2 can be extended to a SOM^n, such as SOM^3=SOM$\times$SOM$\times$SOM defined as a "SOM of SOM^2". In this paper, the algorithms for the SOM^2 and its variations are introduced, and some simulation results are reported.

Keywords: self-organizing map, modular network SOM, mnSOM, manifold learning, fiber bundle, homotopy

1 Introduction

The self-organizing map (SOM) as introduced by Kohonen has provided a powerful and useful tool for data mining, classification, analysis, visualization, and so on [13]. For a labeled dataset, a SOM is one of the best techniques available for visualizing the distribution of each class in the entire data distribution. In an application such as this, a SOM shows how the data vectors of each class are distributed in the high dimensional data space by transforming them to a low dimensional map space while preserving their topological relationships. Though this characteristic of SOMs is effective in many applications, some cases require the visualization of these relationships *between the distributions of classes*, i.e., to what degree two class distributions are similar or different. Alas, a SOM does not provide such information. A SOM provides a map of data vectors, but not a map of class distributions.

The aim of this paper is to propose a method of mapping classes that can represent the relationships between their distributions. In other words, the mapped objects of a SOM are no longer vectors, but class distributions that form manifolds in the data space. Because the distribution of a class, i.e., the manifold, can be represented by a basic SOM, all the classes can be modeled by a group of basic SOMs, which we call "child SOMs". Thus the method involves the generation of a self-organizing map of a group of self-organizing maps, that is, a "SOM of SOMs", abbreviated to "SOM^2" in this paper, because a SOM^2 represents a product manifold formed by SOM$\times$SOM. The architecture of the SOM^2 is simple: it is a hierarchical structure of a set of child SOMs and a single parent SOM, as shown in Fig. 1. Each child SOM learns to represent the corresponding class distribution in the data space, whereas the parent SOM is expected to generate a meta map of the set of maps represented by the child SOMs.

This expansion of the SOM is useful in cases where differences between class distributions are considered more essential than differences between data vectors. As an example, let us consider a classification task of 3D objects from sets of photographs, e.g. face images. Each class in this example consists of a set of 2D images of an object taken from various viewpoints. Even if some photographs of different objects look similar from a certain viewpoint, the entire distributions of data vectors, i.e., 2D images, should be different. In other words, each object corresponds to a unique distribution of data vectors of 2D images [1]. Therefore, it is necessary to generate a map of classes rather than a map of data vectors. In this way, similar items can be identified when a set of sample data from each system, object, or parameter set, etc. is measured.

Let us further clarify the framework of the situation dealt with by the SOM^2, using an example of face image classification. Suppose we have a face image of a person, that is represented by a vector in the high-dimensional data space. If the camera angle is changed continuously, the face images will show a corresponding continuous change. As a result, a one-dimensional trajectory is obtained, which is called the viewpoint manifold. It is also possible to define a continuous map from a one-dimensional camera angle space to a high-dimensional image data space. Such a manifold and map can be modeled using a conventional SOM, by providing a set of face images from different camera angles.

In the above scenario, a manifold is obtained by continuously changing the camera angle, but there is also another way of obtaining a manifold of face images. Suppose that the subject's facial expression changes, e.g., from a sad to a smiling face. The continuous change in facial expression also generates a continuous trajectory in the data space, which is called a face manifold. The set of face images taken from different camera angles and with varying facial expressions forms a product manifold, that consists of a group of viewpoint manifolds and a group of face manifolds. In terms of topology, this product manifold is called a 'fiber bundle'. The map from the camera angle space to the face image space also shows a continuous change when the facial expression changes and is called a homotopy. The fiber bundle and homotopy are the target concepts addressed by the SOM^2.

Let us suppose that we have a set of face images sampled randomly, and suppose further that no information about the camera angle or the facial expression is given. The only clue is the class information, which confirms that a set of data of a class belongs to the same viewpoint manifold. Thus face images belonging to the same class represent the same (yet unknown) facial expression, and are taken from various (unknown) camera angles. In this situation, we need to order the data within each class to show good continuity in the data space, whilst simultaneously ordering the classes. Since a conventional SOM can approximate a data distribution by a manifold, this task is expected to be achieved by a self-organizing map of a set of self-organizing maps, representing a fiber bundle. This is the purpose of the SOM^2, which is, therefore, the extension of a SOM from 'manifold learning' to 'fiber bundle learning'. The SOM^2 is also regarded as an extension from a 'self-organizing map' to a 'self-organizing homotopy'.

Another example where data distributions are more important than individual data points is photograph classification according to color. In this case, a three dimensional RGB histogram is usually used as a feature vector for each photograph. Thus each photograph is regarded as a class, consisting of pixel data in the RGB space. SOMs have also been used to represent color distributions instead of histograms [22, 28]. Therefore it is possible to generate a map of photographs using a SOM^2, by regarding the color SOMs as data vectors. A similar situation arises in texture classification tasks [20, 31].

By extending the concept, it is easy to generalize the SOM^2 to a SOM^n by regarding a group of SOM^{2}s as a set of data vectors to the next higher-level SOM, and so on. For example, a SOM^3 (SOM$\times$SOM$\times$SOM) is a "SOM of SOM^{2}s". In addition, other types of neural maps, e.g., those for neural gas (NG), can be employed as replacements for the parent and/or child SOMs. For example, the "NG of NGs", "SOM of NGs", and "NG of SOMs", which we abbreviate respectively as "NG^2=NG$\times$NG", "NG$\times$SOM", and "SOM$\times$NG", are all possible members of the SOM^n family. Consequently we can obtain numerous self-organizing architectures by further multiplication. In this sense, the real aim of this paper is to present the power of the SOM.

In this paper, the algorithm and some applications of SOM^n are presented. Some preliminary results have been published as conference proceedings [4, 5, 6].

2 Related work

Actually, the importance of the concept has been recognized in the field of the SOM. The adaptive subspace SOM (ASSOM) is an architecture designed for this very purpose [14]. The main difference between the ASSOM and SOM^2 is that the ASSOM represents the given data distributions by a set of infinite linear subspaces, whereas the SOM^2 approximates these by a set of finite nonlinear manifolds. Thus an ASSOM can be denoted as PCA$\times$SOM in our notation. The self-organizing operator map (SOOM) was proposed by Kohonen to represent a homotopy [12]. Although the original concept of the SOOM included a nonlinear case, the algorithm presented by Kohonen was applicable to linear operator cases only. A kernel-based ASSOM and modular network SOM (mnSOM) with multi-layer perceptrons (MLPs) have been proposed as nonlinear extensions of the ASSOM and SOOM, respectively [11, 7, 30, 29]. The common features of these algorithms, with the exception of the SOM^2, are that (i) every nodal unit of the SOM is replaced by a particular module such as PCA, kernel PCA, MLP and so on, and (ii) each nodal module is trained by a weighted mixed dataset so that the module represents an intermediate model of the given classes. These modules can however, only represent appropriate intermediate models when the degree of freedom is limited (low dimensional linear fitting is the typical case). If each module is powerful enough to represent a nonlinear manifold as well, then the module can model different classes at the same time by giving a mixed dataset [19]. It is possible to train such nonlinear modules in a supervised manner by giving enough information during the training. For example, the face image classification task becomes much easier if information about the camera angle is given, which is assumed to be unknown in our framework. These problems are solved in the SOM^2 algorithm, in which intrinsic variables such as the camera angle, are estimated iteratively. In addition, these architectures only have two levels, namely, the lower functional module level and upper SOM level, whereas the SOM^2 can be extended to SOM^n cases.

From the viewpoint of supervised learning, there are two major groups currently using the SOM and other vector quantization techniques. One of these uses a single SOM (or vector quantization architecture) to classify n-classes of data. The supervised SOM is a typical example, and many algorithms have been proposed [15, 8]. The other group uses n-SOMs for n-classes [34, 35]. In the former case, all labeled data are classified in the same map space, whereas each class distribution is modeled by a corresponding SOM in the latter case. Although the SOM^2 is fairly similar to the latter case, it does not belong to either group. The purpose of the SOM^2 is to model not only the given classes, but also any continuous change of the given classes. In another words, the purpose of the SOM^2 is not to segregate classes, but to find a continuous relationship between classes. It can also be stated that the mapping objects dealt with by a SOM^2 are data classes, whereas data vectors are the mapping objects in the conventional SOM.

One may consider hierarchical SOMs and tree structure SOMs [16, 24, 25] to be similar to the SOM^2 because both architectures have hierarchical structures. However the meaning of the hierarchy is completely different. It is worth emphasizing that the SOM^2 does not aim to organize a global map consisting of a set of local maps. Instead, in SOM^2 the child SOMs are global maps of different objects (sets of data vectors).

When using a SOM^2, all data are mapped into two spaces representing independent aspects of the data, namely, the best matching positions in both the parent SOM and child SOM. For example, in the case of face images, each image data is assigned to two coordinates given by the parent/child SOMs and corresponding to the facial expression and camera angle, respectively. Thus the SOM^2 is related to nonlinear independent component analysis (ICA), but is not exactly the same. A method for nonlinear ICA using a conventional SOM has already been proposed [9, 21]. The difference between this method and the SOM^2 is that the two independent axes are explicitly defined in the SOM^2, whereas only a square map space is given for the conventional SOM. Furthermore, an advantage of ICA is that it does not require class information, whereas an advantage of the SOM^2 is that it can solve more difficult cases by utilizing class information. Therefore ICA and the SOM^2 are applicable in different situations.

A SOM is categorized to a subspace method as well. In the pattern recognition field, a vast amount of literature has been published concerning subspace methods including the SOM. Object recognition, face image classification, human action recognition and scene recognition are all representative fields. Within these fields, one of the key points is to extract observation-independent features by representing data as product subspaces, such as (observation-dependent component)$\times$(observation-independent component). An example of this is the recognition of a human action, in which image data are represented by (viewpoint)$\times$(human action) [27]. In the robotics field, such a product manifold method is also useful. Ritter et al. applied SOMs with a functional hierarchy to a robot manipulator [26, 33]. Kurata and Oshiro proposed a method organizing a product space using SOMs, which they applied to the localization task of a mobile robot [17]. These methods are related to our framework, though this work aims to establish a more generalized scheme of the SOM. Thus these works are expected to be re-described from the viewpoint of the SOM^2.

3 Algorithm and architecture of the SOM^n family

3.1 Mathematical notation

In this paper, mathematical formulas are written according to the following rules. Variables i, j and k denote the indexes of the data or class, whereas n, m and l are the indexes of SOM units. Thus $\mathbf{x}^k$ and $\mathbf{w}^n$ represent the kth data vector and the nth reference vector, respectively. Upper case variables, such as I, J, K denote the upper limits of these indexes. Vectors are indicated by boldface, and D is the dimension of a data vector. Vectors denoted by upper case with asterisks depict joint vectors. For example, $\mathbf{W}^*$ represents a vector obtained by joining vectors $\mathbf{w}^1$ to $\mathbf{w}^N$, that is, $\mathbf{W}^* \triangleq (\mathbf{w}^1, \ldots, \mathbf{w}^N) = (\mathbf{w}^n)_{n=1}^N$ and $\mathbf{W}^* \in \mathbb{R}^{N \times D}$. By obeying this notation rule, $\mathbf{U}^{l**}$ represents a joint vector defined by $\mathbf{U}^{l**} \triangleq (\mathbf{u}^{lmn})_{m=1}^{M}{}_{n=1}^{N}$ and $\mathbf{U}^{l**} \in \mathbb{R}^{M \times N \times D}$. Indexes are written as super- or subscripts allowing formulas to obey the tensor notation rule because these joint vectors can be regarded as higher rank tensors. Indexes of best matching units (BMUs) are indicated with hat, so that $\hat{n}_k$, denotes the BMU of $\mathbf{x}^k$ within $\{\mathbf{w}^1, \ldots, \mathbf{w}^N\}$. Similarly, $\hat{n}_k^j$ is the index of the BMU of $\mathbf{x}^k$ within $\{\mathbf{w}^{j1}, \ldots, \mathbf{w}^{jN}\}$. An index with a check, as in $\check{j}_k$, denotes the class of the data. In this case it means that $\mathbf{x}^k$ belongs to the $\check{j}_k$th class.

3.2 Architecture of SOM^2

To begin with, let us first consider the dataset dealt with by the SOM^2. Because the goal is to map a group of class distributions, all data vectors are assumed to be classified and/or labeled in advance. Suppose that there are J classes and K labeled data vectors $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$, and $\mathbf{x}^k \in \mathbb{R}^D$. Let X^j denote the jth class dataset, each of which comprises K_j data points. In addition, it is assumed that the distribution of X^j is approximated by the manifold Φ^j, which can be modeled by a child SOM. The label information is represented by a classification matrix defined by $R_k^j = \delta(j, \check{j}_k)$. Here $\delta(\cdot, \cdot)$ is the Kronecker delta.

The architecture of the SOM^2 has a hierarchical structure as illustrated in Fig. 1. At the first level there are J basic SOMs, which are called 'child SOMs'. (Note that the concept of child SOMs is introduced merely for explanation. As described later, users can write a program for a SOM^2 without referring to child SOMs. Therefore the algorithm can also be applied when J is infinite). The task of the child SOMs is to represent the data distributions of given classes, i.e., to organize a set of 'class maps'. Thus the jth class map is expected to model the distribution of X^j. Each child SOM has the same structure and the same number of reference vectors, denoted by $\mathbf{W}^{j*} = (\mathbf{w}^{j1}, \ldots, \mathbf{w}^{jN})$. Here $\mathbf{w}^{jn} \in \mathbb{R}^D$ means the nth reference vector of the jth child SOM, and the joint reference vector $\mathbf{W}^{j*} \in \mathbb{R}^{N\times D}$ represents the entire jth class map.

At the second level, there is another basic SOM called a 'parent SOM'. The task of the parent SOM is to organize a self-organizing map of a set of self-organizing maps. Let us call the map produced by the parent SOM a '*meta-map*'. Then the parent SOM's task is to generate a meta map of the class maps by giving $\{\mathbf{W}^{j*}\}$ as input data vectors. To achieve this task, the parent SOM has a set of M reference vectors $\{\mathbf{V}^{1*}, \ldots, \mathbf{V}^{M*}\}$, the dimensions of which are equal to $\mathbf{W}^{j*}$. Thus $\mathbf{V}^{m*} \in \mathbb{R}^{N\times D}$ can also be regarded as a a joint vector such that $\mathbf{V}^{m*} = (\mathbf{v}^{m1}, \ldots, \mathbf{v}^{mN})$. Let us call $\mathbf{V}^{m*}$ a '*reference map*'. Finally, the entire meta-map is represented by the joint vector $\mathbf{V}^{**} = (\mathbf{V}^{1*}, \ldots, \mathbf{V}^{M*})$.

The tasks attributed to a SOM^2 are (i) to organize a set of class maps representing the manifold set $\{\Phi^j\}$ which is carried out by the child SOMs, and (ii) to generate a meta-map of the manifold set carried out by the parent SOM. Tasks (i) and (ii) are processed in parallel. Fig. 2 (a) shows the actual simulation results for artificial datasets. In this example, three class distributions X^1, X^2 and X^3, are modeled by the child SOMs $\mathbf{W}^{1*}$, $\mathbf{W}^{2*}$ and $\mathbf{W}^{3*}$, respectively, while the parent SOM has five reference maps $\mathbf{V}^{1*}, \ldots, \mathbf{V}^{5*}$. Thus, in this example, the meta-map space is one-dimensional, while class and reference maps have two-dimensional spaces. The parent SOM orders the given class maps so that the meta-map represents a continuous change in the reference maps. Note that the reference maps $\mathbf{V}^{2*}$ and $\mathbf{V}^{4*}$ in Fig. 2 (a) are created to represent intermediate manifolds by interpolation, and are formed where there are no data points. Such interpolation cannot be achieved by the conventional algorithm in which each module is trained by a mixed dataset. Fig. 2 (b) shows the results of another simulation, in which the number of classes is larger than the number of reference maps ($J > M$). The SOM^2 once again organizes a continuous change in the maps of the given classes similar to the case where $J < M$.

The continuous change in maps is represented by a set of strings connecting the reference maps. These strings represent the '*fibers*', which are defined by a joint vector $\mathbf{V}^{*n} = (\mathbf{v}^{1n}, \ldots, \mathbf{v}^{Mn})$. Some representative fibers are indicated by dotted lines in Figs. 2 (a) and (b). Thus the entire SOM^2 organizes a fiber bundle in the data space, and each reference map represents a homotopic manifold as a section of the bundle. (Note that the dimension of the fibers is equal to that of the meta-map space; thus if the parent SOM has a two dimensional meta-map space, the fibers are also two dimensional.) This means that every data vector is assigned to a pair of low-dimensional coordinates in the meta map, where one coordinate represents the section and the other the fiber to which the data vector belongs. The former is determined as the *best matching map* (BMM) while the latter is determined as the *best matching unit* (BMU) within the BMM. Therefore BMM and BMU mean ‘best matching section’ and ‘best matching fiber’ in the fiber bundle, respectively. This is illustrated in Fig. 3.

3.3 SOM^2 algorithm

To introduce the SOM^2 algorithm, it is convenient to define an operator $\mathcal{S}$, that represents the update algorithm of the conventional SOM. Suppose we have a set of data vectors $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$ and a conventional (basic) SOM, the reference vectors of which are denoted by $\{\mathbf{w}^1, \ldots, \mathbf{w}^N\}$. Thus the entire map organized by the SOM is represented by the joint vector $\mathbf{W}^* = (\mathbf{w}^1, \ldots, \mathbf{w}^N)$. The operator $\mathcal{S}$ is defined as the update process from $\mathbf{W}^*(t-1)$ to $\mathbf{W}^*(t)$ with respect to dataset X as follows.

$$\mathbf{W}^*(t) := \mathcal{S}\left[\mathbf{W}^*(t-1), X;\ \sigma(t)\right]. \tag{1}$$

Here $\sigma(t)$ denotes the neighborhood size. In the batch algorithm, $\mathcal{S}$ is defined by the following equations.

$$\hat{n}_k(t) = \arg\min_n \left\|\mathbf{x}^k - \mathbf{w}^n(t-1)\right\|^2 \tag{2}$$

$$A_k^n(t) = \frac{h\left(d(n, \hat{n}_k(t));\ \sigma(t)\right)}{\sum_{k'} h\left(d(n, \hat{n}_{k'}(t));\ \sigma(t)\right)} \tag{3}$$

$$\mathbf{w}^n(t) := (1-\eta)\,\mathbf{w}^n(t-1) + \eta \sum_{k=1}^{K} A_k^n(t)\,\mathbf{x}^k \tag{4}$$

Here $h(\cdot\,;\,\cdot)$ and $d(\cdot\,,\,\cdot)$ denote, respectively, the neighborhood function and distance between two units in the map space. Learning mass A_k^n represents how much $\mathbf{x}^k$ affects $\mathbf{w}^n$, which is updated to get close to the mass center of $\{\mathbf{x}^k\}$ with masses $\{A_k^n\}$. η is a coefficient determining the update rate, with $\eta = 1$ normally in the batch SOM case. (Hereafter η is assumed to be 1.) It is also possible to define an on-line version $\mathcal{S}$, in which the traditional on-line algorithm is executed for a subset of X. In the conventional case, the operator $\mathcal{S}$ is repeated with a reducing σ until $\mathbf{W}^*$ reaches a steady state. This is the SOM algorithm, i.e., SOM^1. The calculation flow of the SOM^1 is depicted in Fig. 4 (a).

Using operator $\mathcal{S}$, the SOM^2 algorithm can be described in terms of iterations of the following three steps.

Step 1 At the start of every iteration, each class map, i.e., each child SOM is initialized by the reference map of the parent SOM that approximates the class distribution best. The easiest way is to pass the BMM at the preceding iteration to the corresponding child SOM. (This BMM is computed at the previous iteration, by step 3 below.) Thus the initial state of the jth class map at calculation time t is given by

$$\tilde{\mathbf{W}}^{j*}(t) := \mathbf{V}^{\hat{m}_j *}(t-1). \tag{5}$$

If the BMM is not given in the preceding iteration (this sometimes happens in on-line cases), then the least quantization error map is tentatively chosen as the BMM.

$$\hat{m}'_j(t) = \arg\min_m \left[\sum_{k=1}^{K} R_k^j \min_n \left\| \mathbf{x}^k - \mathbf{v}^{mn}(t-1) \right\|^2 \right] \tag{6}$$

$$\tilde{\mathbf{W}}^{j*}(t) := \mathbf{V}^{\hat{m}'_j *}(t-1) \tag{7}$$

Step 2 In each child SOM, a class map $\mathbf{W}^{j*}$ is estimated by applying the SOM algorithm to dataset X^j, and by regarding $\tilde{\mathbf{W}}^{j*}$ as the initial state.

$$\mathbf{W}^{j*}(t) := \mathcal{S}\left[\tilde{\mathbf{W}}^{j*}(t), X^j;\ \sigma_1(t)\right] \tag{8}$$

Here $\sigma_1(t)$ is the neighborhood size for the child SOM update. Note that $\mathbf{W}^{j*}(t)$ is not updated directly from $\mathbf{W}^{j*}(t-1)$, as it is overwritten by its BMM in Step 1.

Step 3 By regarding $Y(t) = \{\mathbf{W}^{j*}(t)\}$ as a set of data vectors, the meta-map is updated by the parent SOM.

$$\mathbf{V}^{**}(t) := \mathcal{S}\left[\mathbf{V}^{**}(t-1), Y(t);\ \sigma_2(t)\right] \tag{9}$$

Here $\sigma_2(t)$ is the neighborhood size for the meta-map update. (The details of how to compute this update, including how to compute BMMs for the class maps, will be discussed later in this section.) After $\mathbf{V}^{**}(t)$ has been updated, the estimated class maps $\{\mathbf{W}^{j*}(t)\}$ are abandoned, and are replaced by the BMMs as the initial state for the next iteration.

These three steps are repeated with reducing neighborhood sizes. The above calculation flow is illustrated in Fig. 4 (b).

The essence of the algorithm is that the parent SOM deals with a set of class maps as if they were ordinary data vectors (Step 3). It is worth noting that each class map is estimated separately without mixing datasets (Step 2). If each class map were organized directly from a mixed dataset, the map would cover all areas of the mixed classes, and would not represent an appropriate, intermediate distribution between classes. Thus it is necessary to estimate class maps separately, although they are not updated independently, because the class maps are overwritten by the BMMs at every iteration (Step 1). This means that the class maps are estimated whilst indirectly affecting one another via the meta-map. If the class maps were estimated independently, every class map would be organized in its own way, and subsequently making a meta map of such class maps would make no sense. This is because there can be several alternative organizations for a good child map, so independently organized child maps of similar objecs could end up in nonsimilar organizations. Therefore, 'separately but not independently' is an important aspect of the SOM^2 algorithm.

Considering the points raised above, the batch algorithm for a SOM^2 is formulated as follows. In Step 1, the reference vectors of the child SOMs $\{\tilde{\mathbf{W}}^{j*}\}$ are overwritten by their BMMs, thus representing the initial state at time t.

$$\tilde{\mathbf{w}}^{jn}(t) := \mathbf{v}^{\hat{m}_j n}(t-1) \tag{10}$$

In Step 2, the class maps are estimated separately for each class.

$$\hat{n}^j_k(t) = \arg\min_n \left\| \mathbf{x}^k - \tilde{\mathbf{w}}^{jn}(t) \right\| \tag{11}$$

$$B^{jn}_k(t) = \frac{R^j_k\, h_1\left(d_1(n, \hat{n}^j_k(t));\, \sigma_1(t)\right)}{\sum_{k'=1}^{K} R^j_{k'}\, h_1\left(d_1(n, \hat{n}^j_{k'}(t));\, \sigma_1(t)\right)} \tag{12}$$

$$\mathbf{w}^{jn}(t) := \sum_{k=1}^{K} B^{jn}_k(t)\, \mathbf{x}^k \tag{13}$$

Here $h_1(\cdot\,;\cdot)$ and $d_1(\cdot\,,\cdot)$ are the neighborhood and distance functions for the class maps. The BMU $\hat{n}^j_k$ determines the best matching fiber of $\mathbf{x}^k$, if $\mathbf{x}^k$ belongs to the jth class.

Finally, in Step 3, the BMM is determined for each class map $\mathbf{W}^{j*}$ by regarding it as a data vector, and then the meta-map is updated by the batch SOM algorithm.

$$\hat{m}_j(t) = \arg\min_m \left\| \mathbf{W}^{j*}(t) - \mathbf{V}^{m*}(t-1) \right\|^2 \tag{14}$$

$$= \arg\min_m \sum_{n=1}^{N} \left\| \mathbf{w}^{jn}(t) - \mathbf{v}^{mn}(t-1) \right\|^2 \tag{15}$$

$$A^m_j(t) = \frac{h_2\left(d_2(m, \hat{m}_j(t));\, \sigma_2(t)\right)}{\sum_{j'=1}^{J} h_2\left(d_2(m, \hat{m}_{j'}(t));\, \sigma_2(t)\right)} \tag{16}$$

$$\mathbf{V}^{m*}(t) := \sum_{j=1}^{J} A^m_j(t)\, \mathbf{W}^{j*}(t) \tag{17}$$

Here $h_2(\cdot\,;\cdot)$ and $d_2(\cdot\,,\cdot)$ are the neighborhood and distance functions for the meta-map. By combining (13) and (17), the update algorithm is formulated as

$$\mathbf{v}^{mn}(t) := \sum_{j=1}^{J} \sum_{k=1}^{K} A_j^m(t)\, B_k^{jn}(t)\, \mathbf{x}^k. \tag{18}$$

Then returning to Step 1, the BMMs are copied to the corresponding child SOMs as the next initial states.

Since the estimated class maps $\{\mathbf{W}^{j*}\}$ are overwritten by the BMMs, there is no need to store the class maps for the next iteration. (18) also means that the meta-map can be updated directly without using the class maps. In other words, each class map is required only for the duration of determining the BMM. This means that the SOM^2 algorithm does not consume much memory even with a huge number of classes.

It is worth noting that the organized meta-map is not affected by the data number of each class, because all class maps are treated equally by the parent SOM. If users prefer a map reflecting the data density, (16) can be modified as follows.

$$A_j^m(t) = \frac{K_j\, h_2\left(d_2(m, \hat{m}_j(t));\ \sigma_2(t)\right)}{\sum_{j'=1}^{K} K_{j'}\, h_2\left(d_2(m, \hat{m}_{j'}(t));\ \sigma_2(t)\right)} \tag{19}$$

An on-line version of the SOM^2 is also available, by replacing the operator $\mathcal{S}$ by the on-line SOM algorithm. Four different combinations of batch and on-line algorithm are then possible, e.g., the child level is on-line while the parent level is batch, and so on. Users can choose any one of these depending on the task.

With regard to parameter settings, one important point needs to be clarified: the neighborhood size of the child SOMs must be reduced more slowly than that of the parent SOM. The reason is that the parent SOM should stabilize earlier than the child SOMs because all child SOMs are overwritten by the parent at every iteration. In this paper, the neighborhood size is reduced as follows,

$$\sigma(t) = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \exp\left[t/\tau\right]. \tag{20}$$

For the batch algorithm, $\tau_{\text{parent}} \simeq 50$ and $\tau_{\text{child}} \simeq 100$ are typical values. Other parameters of the SOM^2, such as the number of reference units, can be determined the same as in the conventional case. Since the SOM^2 is rather robust in terms of parameter settings, consistent results can be obtained as long as extreme parameters are not used.

3.4 Generalization from SOM^2 to SOM^n

By adding a higher level, it is easy to generalize the SOM^2 to SOM^n, such as SOM^3 = SOM $\times$ SOM $\times$ SOM. To explain the role of the SOM^3, let us consider a situation in which a user requires a map of face images.

Suppose that a class consists of a set of photographs taken from various viewpoints at any one time. Thus the photographs belonging to a single class represent the same person with the same facial expression, but from different view points. Suppose further that each image set is classified according to subject, e.g., 'the set of image sets of Mr. A' and so on. In this case, the SOM^2 is required to model a set of image sets of one person, while the SOM^3 can deal with a set of sets of image sets classified by subject.

Now suppose that there are K data points $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$ that are classified according to J classes $\{X^1, \ldots, X^J\}$, and these J class maps are classified according to I meta-classes $\{Y^1, \ldots, Y^I\}$. Let R^j_k denote the classification matrix of data to class as defined previously, and let Q^i_j denote the meta-classification matrix of class to meta-class, defined as $Q^i_j = \delta(i, \check{i}_j)$. Thus $Q^i_j = 1$ if the jth class belongs to the ith meta-class, otherwise $Q^i_j = 0$.

The structure of a SOM^3 is described below. In the first level there are J child SOMs as is the case for a SOM^2, but there are I parent SOMs in the second level, that organize the set of meta class maps. Finally in the third level, there is a grandparent SOM, which produces a meta-meta-map as the output of the SOM^3. Let $\mathbf{W}^{j*}$ and $\mathbf{V}^{i**}$ denote the jth class map (child SOM) and the ith meta map (parent SOM), respectively. Suppose further that the grandparent SOM has L reference meta-maps $\{\mathbf{U}^{1**}, \ldots, \mathbf{U}^{L**}\}$. Thus the joint vector $\mathbf{U}^{***}$ represents the entire meta-meta-map. Since the reference meta-map $\mathbf{U}^{l**}$ at the third level can be regarded as a tensor of rank 3, it would be natural to call the SOM^3 a 'SOM of rank 3'.

The SOM^3 algorithm is given by the following equation. At the start, all class and meta class maps are replaced by the BMMs determined at $(t-1)$. Thus,

$$\tilde{\mathbf{U}}^{***}(t) := \mathbf{U}^{***}(t-1) \tag{21}$$

$$\tilde{\mathbf{V}}^{i**}(t) := \mathbf{U}^{\hat{l}_i**}(t-1) \tag{22}$$

$$\tilde{\mathbf{W}}^{j*}(t) := \mathbf{U}^{\hat{l}_j\hat{m}_j*}(t-1). \tag{23}$$

Here $\hat{l}_j \triangleq \hat{l}_{\check{i}_j}$ means the best matching meta-map (BMMM) of the meta class to which the jth class belongs. Then the SOMs at each level are updated by the SOM algorithm as follows.

$$\mathbf{W}^{j*}(t) := \mathcal{S}\left[\tilde{\mathbf{W}}^{j*}(t), X^j;\ \sigma_1(t)\right] \tag{24}$$

$$\mathbf{V}^{i**}(t) := \mathcal{S}\left[\tilde{\mathbf{V}}^{i**}(t), Y^i(t);\ \sigma_2(t)\right] \tag{25}$$

Here $Y^i(t) = \{\mathbf{W}^{j*}(t)\,;\ \text{all } j \text{ for which } \check{i}_j = i\}$. Finally the grandparent SOM is updated by regarding $Z(t) = \{\mathbf{V}^{1**}(t), \ldots, \mathbf{V}^{I**}(t)\}$ as a set of data vectors.

$$\mathbf{U}^{***}(t) := \mathcal{S}\left[\tilde{\mathbf{U}}^{***}(t), Z(t);\ \sigma_3(t)\right] \tag{26}$$

Here σ_3 is the neighborhood size for the grandparent SOM. In the case of a batch-algorithm, the above algorithm becomes

$$C_k^{jn}(t) = \frac{R_k^j\, h_1\left(d_1(n, \hat{n}_k^j(t));\ \sigma_1(t)\right)}{\sum_{k'=1}^{K} R_{k'}^j\, h_1\left(d_1(n, \hat{n}_{k'}^j(t));\ \sigma_1(t)\right)} \tag{27}$$

$$\mathbf{w}^{jn}(t) := \sum_{k=1}^{K} C_k^{jn}(t)\,\mathbf{x}^k \tag{28}$$

$$B_j^{im}(t) = \frac{Q_j^i\, h_2\left(d_2(m, \hat{m}_j^i(t));\ \sigma_2(t)\right)}{\sum_{j'=1}^{J} Q_{j'}^i\, h_2\left(d_2(m, \hat{m}_{j'}^i(t));\ \sigma_2(t)\right)} \tag{29}$$

$$\mathbf{V}^{im*}(t) := \sum_{j=1}^{J} B_j^{im}(t)\,\mathbf{W}^{j*}(t) \tag{30}$$

$$A_i^l(t) = \frac{h_3\left(d_3(l, \hat{l}_i(t));\ \sigma_3(t)\right)}{\sum_{i'=1}^{I} h_3\left(d_3(l, \hat{l}_{i'}(t));\ \sigma_3(t)\right)} \tag{31}$$

$$\mathbf{U}^{l**}(t) := \sum_{i=1}^{I} A_i^l(t)\,\mathbf{V}^{i**}(t) \tag{32}$$

Here $h_3(\cdot\ \cdot)$ and $d_3(\cdot\,,\cdot)$ are the neighborhood and distance functions for the grandparent SOM. By combining (28), (30), and (32), the complete update algorithm can be formulated as

$$\mathbf{u}^{lmn}(t) := \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} A_i^l(t)\,B_j^{im}(t)\,C_k^{jn}(t)\,\mathbf{x}^k. \tag{33}$$

Equation (33) implies that A_i^l, B_j^{im}, and C_k^{jn} are vital for updating the SOM^3, whereas the class maps that consume vast amounts of memory are not indispensable in the actual programming. By applying further nesting, one can easily obtain an algorithm for a SOM^n.

3.5 Variations of the SOM^2 using the Neural Gas Algorithm

By adopting other vector quantization methods, many variations in the SOM^2 can be created. Here we introduce neural gas (NG) [18, 3], by first considering an “NG of NGs,” i.e., NG^2=NG×NG. The only difference between SOM and NG is the way in which learning masses are determined: by a neighborhood function in the SOM, and by a function of ‘order’ in the NG model. For each data vector, the order of the winning unit is 0, while the next winning unit has order 1. Therefore, the NG^2 algorithm can easily be obtained by replacing the neighborhood function by a function of order. Thus Eqs. (12) and (16) are replaced by

$$\beta_k^{jn}(t) = \frac{R_k^j \exp\left[-s(\tilde{\mathbf{w}}^{jn}(t), \mathbf{x}^k)/\lambda_2(t)\right]}{\sum_{k'=1}^{K} R_{k'}^j \exp\left[-s(\tilde{\mathbf{w}}^{jn}(t), \mathbf{x}^{k'})/\lambda_2(t)\right]} \tag{34}$$

$$\alpha_j^m(t) = \frac{\exp\left[-s(\mathbf{V}^{m*}(t), \mathbf{W}^{j*}(t))/\lambda_1(t)\right]}{\sum_{j'=1}^{J} \exp\left[-s(\mathbf{V}^{m*}(t), \mathbf{W}^{j'*}(t))/\lambda_1(t)\right]}. \tag{35}$$

Here, $s(\tilde{\mathbf{w}}^n, \mathbf{x}^k)$ is the function that gives the order of $\tilde{\mathbf{w}}^n$ to the data $\mathbf{x}^k$, while $\lambda_1(t)$ and $\lambda_2(t)$ give the rates of decay. Therefore the complete update algorithm (18) becomes

$$\mathbf{v}^{mn}(t) = (1-\eta)\,\mathbf{v}^{mn}(t-1) + \eta \sum_{j=1}^{J} \sum_{k=1}^{K} \alpha_j^m(t)\beta_k^{jn}(t)\,\mathbf{x}^k \tag{36}$$

In the NG^2 case, it is better to set η to be smaller than 1. If one needs a "SOM of NGs" (NG×SOM), it can be obtained by combining (16) and (34) as

$$\mathbf{v}^{mn}(t) = (1-\eta)\,\mathbf{v}^{mn}(t-1) + \eta \sum_{j=1}^{J} \sum_{k=1}^{K} A_j^m(t)\beta_k^{jn}(t)\,\mathbf{x}^k, \tag{37}$$

whereas the combination of (12) and (35) represents the "NG of SOMs" (SOM×NG). In addition, if one employs other types of mapping algorithms, the number of variations will increase further. Of these variations, one of the most promising architectures is the NG×SOM, i.e., the "SOM of NGs". This is because there are no restrictions on the dimensions of the target manifolds in the child NGs, and the meta SOM allows one to visualize the relationships of the classes. The NG×SOM therefore inherits the advantages of both algorithms.

It is also possible to employ other algorithms such as the Generative Topographic Map (GTM) and kernel-based maximum entropy learning rule (kMER) [2, 10]. In either case, the user would simply replace the operator $\mathcal{S}$ by the respective algorithm. The important point is that the concept of the SOM^2 includes all these variations as long as $\mathcal{S}$ is definable, and thus there is no need to be restricted to Kohonen's narrowly-defined SOM.

4 Simulations and Results

4.1 Maps of artificial manifolds

To validate the performance of the SOM^2, artificial manifold sets were used in the simulations. Fig. 5 (a) shows the first manifold set, which contains a small number of classes ($J = 9$), with each class having a large number of data vectors ($K_j = 400$) for random sampling. The shapes of the manifolds are all congruent triangles, the positions and orientations of which gradually change. There are $M = 5 \times 5$ reference maps, with $N = 6 \times 6$ reference vectors for each reference and class map.

Figs. 5 (b) and (c) show respectively, the reference maps and meta-map generated by the SOM^2. As shown in Fig. 5 (b), all reference maps adhere well to the triangular shape, while the positions and orientations vary in such a way that a continuous map of the manifolds is formed successfully as the meta-map. This result also shows that the unknown intermediate manifolds can be estimated correctly by interpolation. Furthermore, the reference maps are aligned so that reference vectors with the same index correspond to a congruent point on each manifold. For example, the reference vectors indicated by arrowheads in Fig. 5 (c) have the same index, i.e., these units form a fiber. Therefore it is possible to observe how the manifold gradually changes shape by tracing the fibers. This homologizing ability is one of the advantages of the SOM^2.

In contrast to the first example, the second set shown in Fig. 6 (a) has a large number of classes ($J = 400$), each of which contains a very small number of samples ($K_j = 4$). The shapes of the manifolds are all congruent triangles, the positions and orientations of which were changed in the same manner as in the first example. Nine out of 400 manifold shapes are shown in Fig. 6 (a) with the data vectors belonging to the class (indicated by larger markers). Unlike the first case, however, it is difficult to recognize the shapes of the manifolds due to deficiencies in the samples. Furthermore, the manifolds of the second set overlap each other in such a way that the sample vectors are distributed evenly over the area without forming clusters. The results are almost the same as in the first manifold set (Figs. 6 (b) and (c)). The meta-map is well organized with good continuity, i.e., the positions and orientations of the reference maps vary continuously. The SOM^2 also succeeds in estimating the distribution of every class, even though the number of samples per class is very small. One may notice that the reference maps appear smaller in size when compared with the original manifold. However, the reference maps still succeed in estimating the average data distributions because the data points are usually distributed in a smaller area than the triangle of the manifolds due to the limited data. In addition, the SOM^2 succeeds in homologizing the given manifolds as in the first case (refer to the reference vectors indicated by arrowheads).

Fig. 7 shows another situation in which data points are not distributed in homologous manifolds. In this experiment there are three classes, the data points of which are distributed in the shapes of the letters ‘S’, ‘O’ and ‘M’ in $\mathbb{R}^2$ space (Fig. 7 (a)). Even though the manifolds are not homologous, the SOM^2 still tries to ascertain good correspondences between the given distributions. Fig. 7 (b) illustrates the meta-map organized by the SOM^2. Some representative fibers are also indicated, showing how the SOM^2 represents a continuous meta-map.

For this third experiment, it would be preferable to use an NG×SOM instead of a SOM^2. Fig. 7 (c) depicts the meta-map organized by a NG×SOM. Here, the data distributions are well represented by the reference NGs. As a result, the NG×SOM depicts a morphing of letters from 'S' into 'O' and from 'O' into 'M'.

4.2 Maps of shapes

One of the application areas of the SOM^n family is shape classification. To show the ability of the SOM^2 for this purpose, three simulations were done. The first dataset is shown in Fig. 8 (a). In this simulation, 15 contours are used, each of which contains 400 dots. Thus each contour forms a 1-dimensional closed manifold and 400 data vectors in the *x*–*y* space were sampled from each manifold. Let us call this method the 'Dot Distribution Representation' (DDR). To represent the manifold shape, each child SOM has a 1-dimensional closed structure with 36 reference vectors. The results are shown in Fig. 8 (b). As illustrated in the figure, the reference maps fit the contours well, and the parent SOM successfully generates a map of these contours. Furthermore, the SOM^2 interpolated the given contours resulting in a meta-map with good continuity.

If these contours are classified a priori, then a SOM^3 can be used. In the second simulation, it was assumed that each row of Fig. 8 (a) constitutes a class, i.e., the 15 contours were classified into 3 classes in advance. Thus there are 3 contour classes (rectangles, ellipses, and diamonds), each of which has 5 contours with different sizes and orientations, with each contour containing 400 data vectors using the DDR method. In this simulation, the task of the grandparent SOM is to represent the map of the contour classes, while the task of each parent SOM is to represent a map of the contours belonging to each class, and finally the task of every child SOM is to represent a contour. Each child SOM has a 1-dimensional closed structure as in the previous simulation, whereas the parent and grandparent SOMs have 1-dimensional linear structures. The results are shown in Fig. 8 (c). In the figure, each box corresponds to a reference map in the 1st level, while each row represents a meta-map generated in the 2nd level. Finally the entire map represents the meta-meta-map generated by the grandparent SOM in the 3rd level. According to the results, the grandparent SOM successfully generates a continuous meta-meta-map of contour classes, morphing from diamond to rectangle (viewing the map from top to bottom), while every parent SOM generates a meta-map of a contour class, with gradual variations in size and orientation (viewing the map from left to right). In addition, every child SOM regenerates a contour well; the red dots in Fig. 8 (c) show the original data overlaid on the BMMs. In other words, one can see a class of contours varying in size and orientation in every row, i.e., in every section of the fiber bundle, and a set of contours with similar size and similar orientation in every column, i.e., in every fiber.

If the data vectors do not form clear manifolds, an NG would be a better solution. In the next simulation, the task is to generate a map of face contours as shown in Fig. 9. To enable the DDR method to be applied to photograph images, 15 face images were first decomposed into small dots using a Laplacian Gaussian filter, followed by binarization with a threshold. DDR is expected to result in a more natural data representation, because the continuous movement of an image is represented by a continuous movements of dots. In this case, 600 dots were sufficient to express a face image. To organize a self-organizing map of the DDR images, an NG$\times$SOM was used. Fig. 9 illustrates the results. Every reference NG represents a face image, while the meta map space successfully shows a continuous change in camera angle.

4.3 Map of 3D objects from 2D images

The task of the next simulation is to generate a map of 3D objects from 2D projected images. Put more clearly, the task is to make a self-organizing map of 3D objects from a set of image sets, each of which contains several image data of one object from various viewpoints. Since a set of 2D images of a 3D object observed from various angles forms a viewpoint manifold in the high dimensional data space, a SOM^2 is expected to organize a meta-map of classes, as has already been described. Note that the SOM^2 does not know how 3D objects can be reconstructed from their 2D images.

The first simulation represents an artificial situation, in which the 13 objects shown in Fig. 10 (a) are used as the dataset. Here the objects are assumed to be flexible grids (9$\times$9 nodes), and each data vector consists of a set of (x, y) coordinates of the lattice points on a 2D image. Thus each 2D image corresponds to a $9\times9\times2 = 162$ dimensional data vector. Each class consists of 9 data vectors observed from different viewpoints. Fig. 10 (b) is the meta-map of the 3D objects generated by the parent SOM, along with two representative reference maps. The meta-map was generated successfully, showing good continuity of varying 3D shapes from a flat shape (bottom left in Fig. 10 (b)) to a prominent peaked shape (top right).

In other words, the SOM^2 generated a map where the actual 13 input objects are interpolated into 49 objects, and object shapes change smoothly over the intermediate 3D objects. Each reference map generated a map of 2D images of an object consisting of 25 images. Thus every reference map also interpolated intermediate viewpoints of the objects. Furthermore, all the maps are aligned with one another in such a way that all reference vectors with the same index are assigned to the images taken from the same viewpoint. This means that every fiber corresponds to a viewpoint. This is also an effect of the homologizing ability of the SOM^2.

Fig. 11 shows a more practical case, in which sets of face images are given to a SOM^2. The images are 204 photographs of 12 people from various viewpoints. These face images were classified into 12 classes in advance, but they were not sorted in any way, thus providing no clue to identifying the camera angle. Each image consists of 75×75 pixels, which are regarded as the 5625-dimensional data vectors without any image-preprocessing. In this simulation, the parent SOM consists of a set of 1-dimensional reference maps arrayed in a 2-dimensional meta-map space. Therefore the entire SOM^2 has a cubic map space as shown in Fig. 11 (a). Fig. 11 (b) shows a meta-map and two representative reference maps organized by the SOM^2. Every reference map (i.e. section) represents the continuous change in camera angle for a person, while every fiber represents a map of faces taken from a certain viewpoint.

4.4 Application of SOM^3 to autonomous mobile robot

The last simulation is an application of the SOM^3 to an autonomous mobile robot. The task is to build a geographical image (i.e. the inner model) of the work field from a set of visual images, which have been obtained during exploration. This application of a SOM was also proposed by Kurata and Oshiro [17]. To achieve this task, the SOM^3 illustrated in Fig. 12 (a) is employed. As is shown in the figure, the SOM^3 consists of three levels, each of which codes different properties of the given information.

The task of the SOMs at level 1, i.e., the child SOMs, is to extract shift invariant information from each image. By regarding the color pixels of each image as a set of data, each child SOM is expected to represent a color distribution of the image. This image coding is expected to be robust with respect to the robot movement, because the color distribution changes gradually as the robot continues to move. In addition, this method can drastically reduce the dimension of the image data. Here we refer to these as ‘color maps’.

In level 2, there is a set of parent SOMs, each of which has a circular topology. The task of the parent SOMs is to code the orientation of the robot, by regarding the color maps as data vectors. To achieve this, a set of color maps obtained at a particular location is given to a parent SOM.

Finally in level 3, there is a grandparent SOM, with a two dimensional map space. The task of the grandparent SOM is to code the position of the robot in the work field, as in place cells. Therefore the entire SOM^3 represents a $2 + 1 + 2$ dimensional fiber bundle. It should be noted that this five dimensional fiber bundle is located in the three dimensional data space because the most primitive data, i.e., each color pixel, only carries RGB information. It is emphasized again that the data distribution, rather than each data point, carries the essential information in the SOM^n algorithm.

The details of the experiment are now given. A robot simulator “WEBOTS” was used. The virtual work field used in this experiment is shown in Fig. 12 (b). The robot had a color camera with 40×16 (W$\times$H) pixels, and the horizontal view angle was approximately 120°. Each image was divided into 5 subimages with 16×16 pixels. Each subimage was regarded as a class of color values (a color map). Therefore five successive color maps were obtained from each image. To determine the BMMs at level 2 of the SOM^3, a group of color maps were matched to a block of reference maps. Thus five color maps were matched to five successive reference maps. The robot observed the landscape at 225 points in the field, and 20 images were obtained at each point. During the exploration, the robot movement was controlled manually.

Though it is rather an unnatural assumption, the information about the sequential movement of the robot was assumed lost. Consequently, a huge number of unordered snapshots were given to the SOM^3. The only clue was the meta-class information, which confirmed that images belonging to the same meta-class were observed at the same location. In practice, we can utilize a priori knowledge, for example, that the robot never leaps in the field, etc. to simplify the task. However, because the aim of this experiment is to demonstrate the performance of a SOM^n, any assistance from a priori knowledge was disregarded.

Fig. 12 (c) illustrates the meta-map organized by the SOM^3, along with two representative reference meta-maps. In this figure, the camera images are indicated at the BMM. The geographical topology in the work field is preserved in the meta-meta-map space, while each reference meta map represents the orientation at a position. Furthermore, the reference maps with the same index represent the same orientation. Therefore the inner image of the work field geography is successfully built in the robot. By using the organized meta-meta-map, the robot can itself localize its position and orientation.

5 Discussion

5.1 Homologizing ability of SOM^2

Briefly, a SOM^2 is a learning machine that represents a fiber bundle, whereas a conventional SOM is a machine representing a manifold. Therefore, the homologizing ability is one of the central functions of the SOM^2. Interestingly, there is no explicit process for homologization in the SOM^2 algorithm.

Fig. 13 shows the process of how a SOM^2 homologizes two manifolds. Here, only two data manifolds were given to a SOM^2 with two reference maps. In the middle of the simulation, the indexes of one of the reference maps and the corresponding class map were explicitly renumbered. Fig. 13 (a) shows the situation just after the renumbering. In this case, the upper map was rotated 90°. As a result, the fibers were twisted 90°. Figs. 13 (b), (c), and (d) show the next three iterations. As shown in these figures, the two maps immediately turned so as to minimize the distance between the two reference maps.

The reason why the class maps turned is that the class maps are overwritten by their BMMs at every iteration. Since the reference maps are smoothed by the effect of the neighborhood function in the SOM algorithm, the difference between reference maps becomes smaller than the difference between class maps. By substituting these smoothed reference maps into the class maps, the organized class maps are gradually aligned and the total length of fibers decreases. In other words, the fibers work like elastic rubber bands connecting the reference maps. Therefore, the homologizing ability of the SOM^2 is built into the SOM algorithm by its very nature.

5.2 On determining BMMs

There are two different ways to determine the BMM for a dataset. One is to use the least quantization error map, as defined by (6), while the other is to measure the distance using class maps, as defined by (14). The latter method is used in the SOM^2 algorithm to determine the BMMs.

To compare the difference between the two methods, let us consider the situation in which we have a dataset X and two reference maps $\mathbf{V}_A^*$ and $\mathbf{V}_B^*$ as illustrated in Fig. 14. By using the former method, $\mathbf{V}_B^*$ becomes the BMM for X, because one of the reference vectors in $\mathbf{V}_B^*$ wins all data points of X (Fig. 14 (a)). In contrast, $\mathbf{W}_A^*$ becomes the BMM in the latter method, because the entire distribution is more similar than $\mathbf{V}_B^*$ (Fig. 14 (b)). In this method, the class map $\mathbf{W}_X^*$ is estimated using a child SOM, and then the distances from $\mathbf{W}_X^*$ to $\mathbf{V}_A^*$ and $\mathbf{V}_B^*$ are compared. This is an important aspect of the SOM^2 highlighting how it differs from other algorithms.

Ideally speaking, the class map should be estimated individually for each reference map, by letting each reference map be the initial state of the class map. Thus, to measure the distance between X and $\mathbf{V}_A^*$, the class map $\mathbf{W}_X^*$ is initialized by $\mathbf{V}_A^*$, while $\mathbf{V}_B^*$ becomes the initial state when the distance from X to $\mathbf{V}_B^*$ is measured. This means that the distance between a map $\mathbf{V}^*$ and a dataset X is defined by the distance that $\mathbf{W}_X^*$ travels from the initial state $\mathbf{V}^*$ to the dataset X. To estimate $\mathbf{W}_X^*$, it is sufficient to execute the batch-SOM algorithm one or more times. The distance measurement is formulated using the operator $\mathcal{S}$ as follows.

$$L^2(\mathbf{V}^*, X) \triangleq \left\|\mathbf{W}_X^* - \mathbf{V}^*\right\|^2 \tag{38}$$

$$\mathbf{W}_X^* = \mathcal{S}\left[\mathbf{V}^*, X;\, \sigma\right] \tag{39}$$

This distance measurement is the recommended one. Since BMUs need also be determined in the former method, the increased calculation cost of the latter method is not that much in comparison.

In the learning algorithm for the SOM^2, each class map is only estimated from the latest BMM. Rigorously, it is better to measure distances using the method described above. Thus the BMM should be determined by substituting all reference maps into the class map as the initial states. Fortunately there is no need to be too concerned about this issue, because the homologizing ability of the SOM^2 aligns the class maps gradually. In fact all simulation results presented in this paper were produced by the simple algorithm without estimating class maps for every reference map.

After a meta-map has been obtained using the training datasets, the organized meta-map can be used to classify new datasets. To do this classification, it is also necessary to determine BMMs for the datasets. For this the homologizing ability of SOM^2 is not available, because the meta-map has already been fixed. Therefore the distance must be measured by estimating class maps for each reference map.

5.3 Calculation cost

Since a SOM^2 has the ability to represent datasets using two different levels of maps, it allows us to deal with more complex data than the conventional SOM. A corresponding increase in the size and complexity of the calculation is expected. We now compare the calculation cost of a SOM^2 with M reference maps, each of which has N reference vectors, with that of a conventional batch SOM with MN reference vectors. Note that both architectures have an equal number (i.e. MN) of reference vectors.

Since most of the calculation time is consumed by the process determining BMUs and BMMs, we evaluate the calculation cost according to the number of times the D dimensional Euclidean distance is measured. In the conventional SOM, the Euclidean distances are measured KMN times for every iteration, whereas in the SOM^2, the Euclidean distances are measured KN times for the child SOMs and JMN times for the parent. Thus the calculation costs of the SOM and SOM^2 are $O(KMN)$ and $O(KN + JMN)$, respectively. Surprisingly, execution of the SOM^2 is much faster than that of the conventional SOM, because the number of classes J is usually far less than the data number K. This is a real advantage in using a SOM^2 in practical applications.

5.4 Generative model of the framework

Finally, let us revisit the viewpoint of geometry to consider the theoretical framework underlying the SOM^2. Suppose that the data vectors are distributed on a set of nonlinear homotopic manifolds $\{\Phi^j\}$ in $\mathbb{R}^D$, with the manifolds having d-dimensions ($d < D$). Therefore, the manifold Φ^j can be represented by a nonlinear map f_j from $\mathbb{R}^d$ to $\mathbb{R}^D$ as

$$f_j : \xi \to \mathbf{x} \qquad \xi \in \mathbb{R}^d,\ \mathbf{x} \in \Phi^j. \tag{40}$$

Here ξ is an intrinsic variable that cannot be observed. The nonlinear maps $\{f_j\}$ are assumed to vary continuously according to an intrinsic parameter $\theta \in \mathbb{R}^P$ as follows.

$$F(\xi, \theta_j) = f_j(\xi). \tag{41}$$

Thus F is the homotopy underlying this generative model, and $\{\Phi^j\}$ are the sections of a fiber bundle generated by F. Furthermore, let the probability densities of ξ and θ be given, respectively, by $p(\xi)$ and $p(\theta)$, which are assumed to be independent. Without loss of generality, $p(\xi)$ and $p(\theta)$ can be assumed to obey uniform distribution defined in $[0,1]^d$ and $[0,1]^P$, respectively. Under these conditions, the distance between two homotopic manifolds Φ^1 and Φ^2 can be defined as,

$$L^2(\Phi^1, \Phi^2) \triangleq \int_{\xi\in\mathbb{R}^n} \|f_1(\xi) - f_2(\xi)\|^2\, p(\xi)\, d\xi \tag{42}$$

$$= \int_{\xi\in[0,1]^d} \|F(\xi,\theta_1) - F(\xi,\theta_2)\|^2\ d\xi. \tag{43}$$

Thus the goal of the architecture is to solve the inverse problem, namely, to estimate the fiber bundle and homotopy F from the given data vectors, that are generated by two unknown independent random variables ξ and θ. The only clue is that a set of data belonging to the same class is produced by the same θ. To achieve this goal, the architecture is expected to model each dataset as a homotopic manifold, whilst simultaneously ordering these manifolds to form a natural fiber bundle.

In the above generative model, the distance between manifolds is defined by (43), while in the SOM^2 algorithm it is approximately measured by the distance between two maps obtained by the operator $\mathcal{S}$ as (14) (see also Fig. 14 (b)). To measure the manifold distance (43) more precisely, it is better to define $\mathcal{S}$ as an equiprobability mapping algorithm rather than a SOM, because this causes the density of fibers to be proportional to the data density (a SOM does not generate an equiprobability map due to the magnification problem [23, 32]). Thus it is expected that the precision of the meta-map of the SOM^2 will be improved by using an equiprobability mapping, such as GTM and kMER, in exchange for calculation cost [2, 10].

Finally, let me present several points that should be emphasized. (i) All advantages and disadvantages of the SOM are inherited by a SOM^n because theSOM^n is a straightforward generalization of the SOM. As the SOM is still a powerful and effective algorithm in many fields despite the lack of theoretical assurance, the SOM^n would be equally useful and effective. (ii) The concept of the SOM^n is not limited to Kohonen's SOM. By modifying the operator $\mathcal{S}$, the SOM^n algorithm can be improved. Thus plenty of siblings and modifications of the SOM are possible in the SOM^n. In this sense the algorithm presented in this paper based on Kohonen's SOM should be seen rather as an implementation of the widely-defined SOM^n. (iv) Calculation speed is very important in practical tasks. The concept of the SOM^n provides us huge scale architectures, that are able to deal with more complex data structures. The simulation of the autonomous mobile robot, in which sensor data are organized in terms of position and orientation, is an example of this, and proves that the SOM^2 is a good architecture with high speed calculations.

The issues concerning the equiprobability mapping and theoretical derivation from the generative model should be examined further. These have been set as tasks for the future.

6 Conclusion

The concept of the SOM^n, together with an algorithm, have been presented in this paper. The essence of the algorithm is to generate a higher-rank of data representation with class information as a clue, and the given datasets are modeled by fitting to a fiber bundle. The algorithm has a hierarchical structure of the operator $\mathcal{S}$, which is usually defined by Kohonen's SOM algorithm. Since the concept of the SOM^n allows us to multiply by SOM and other mapping algorithms, we can obtain fruitful variations. It can therefore be expected that the SOM^n will be useful within many application fields.

References

1. Ando, H., Suzuki, S., & Fujita, T. (1999). Unsupervised visual learning of three-dimensional objects using a modular network architecture. *Neural Networks*, **12**, 1037–1051.

2. Bishop, C. M., Svensén, M., & Williams, C. K. I. (1998). GTM: The generative topographic mapping. *Neural Computation*, **10**(1), 215–234.

3. Cottrell, M., Hammer, B., Hasenfuss, A., & Villmann, T. (2005). Batch neural gas. *Proceedings of the 5th Workshop on Self-Organizing Maps (WSOM2005)* (pp. 275–282). Paris

4. *Written by the author*

5. *Written by the author*

6. *Written by the author*

7. Furukawa, T., Tokunaga, K., Morishita, K., & Yasui, S. (2005). Modular network SOM (mnSOM): From vector space to function space. *Proceedings of the International Joint Conference on Neural Networks 2005 (IJCNN2005)* (pp. 1581–1586). Montreal

8. Hammer, B., Strickert, M., & Villmann, T. (2005). Supervised neural gas with general similarity measure. *Neural Processing Letters*, **21**, 21–44.

9. Haritopoulos, M., Yin, H., & Allinson, N. (2001). Nonlinear blind source separation using SOMs and applications to image denoising. In N. Allinson, H. Yin, L. Allinson, & J. Slack, *Advances in self-organizing maps* (pp. 275–282). London: Springer-Verlag.

10. Van Hulle, M. M. (2000). *Faithful representation and topographic maps: From distortion to information-based self-organization*. Willey-Interscience.

11. Kawano, H., Horio, K., & Yamakawa, T. (2005). Nonlinear adaptive manifold self-organizing map with reproducing kernels and its application to pose invariant face recognition. *IEEJ Transactions on EIS*, **125**, 948–955.

12. Kohonen, T. (1993). Generalization of the self-organizing map. *Proceedings of the International Joint Conference on Neural Networks 1993 (IJCNN93)* (pp. 457–462).

13. Kohonen, T. (2001). *Self-Organizing Maps* (3rd ed). Berlin: Springer-Verlag.

14. Kohonen, T., Kaski, S., & Lappalainen, H. (1997). Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM. *Neural Computation*, **9**, 1321–1344.

15. Kohonen, T., Mäkisara, K., & Saramäki, T. (1984). Phonotopic maps — insightful representation of phonological features for speech recognition. *Proceedings of the International Conference on Pattern Recognition* (pp.182–185). Los Alamitos

16. Koikkalainen, P. (1993). Fast organization of the self-organizing maps. *Finnish Artificial Intelligence Society*, 51–62.

17. Kurata, K., & Oshiro, N. (2004). Separating visual information into position and direction by SOM. *Proceeding of International Symposium on Artificial Life and Robotics* (pp. 5–8). Oita

18. Martinetz, T.M., Berkovich, S.G., & Schulten, K.J. (1993). Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, **4**, 558–569.

19. Ohkubo, T., Tokunaga, K., & Furukawa, T. (2007). Self-organizing homotopy networks: Comparisons among modular network SOM, SOM of SOMs and parametric bias method. *International Congress Series*, **1301**, 168–171.

20. Oja, E. & Valkealahti, K. (1996). Co-occurrence map: Quantizing multidimensional texture histograms. *Pattern Recognition Letters*, **17**, 723–730.

21. Pajunen, P., Hyvärinen, A., & Karhunen, J. (1996). Nonlinear blind source separation by self-organizing maps. *Proceedings of International Conference on Neural Information Processing (ICONIP 1996)*, **2**, (pp. 1207–1210).

22. Pei, S-C. & Lo, Y-S. (1998). Color image compression and limited display using self-organization Kohonen map, *IEEE Trans. on Circuits and Systems for Video Technology*, **8**(2), 191–204.

23. Ritter, H., & Schulten, K. (1986). On the stationary state of Kohonen's self-organizing sensory mapping. *Biological Cybernetics*, **54**, 99–106.

24. Samsonova, E.V., Kok, J.N., & IJzerman, A.P. (2006). TreeSOM: cluster analysis in the self-organizing map. *Neural Networks*, **19**, 935–949.

25. Sauvage, V. (1997). The T-SOM (Tree-SOM). *Lecture Notes in Computer Science*, **1342**, 389–197.

26. Ritter, H., Martinetz, T., & Schulten, K. (1992). Neural Computation and Self-Organizing Maps — An Introduction. Addison-Wesley.

27. Souvenir, R., & Babbs, J. (2008). Learning the viewpoint manifold for action recognition. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2008* (pp. 1–7). Anchorage

28. Tanaka, K., Hoshi, N., & Horiguchi, T. (2003). Color image compression algorithm using self-organizing feature map. *Interdisciplinary Information Science*, **9**(2), 201–208.

29. Tokunaga, K., & T. Furukawa (2009). Modular network SOM. *Neural Networks*, (*in press*).

30. Tokunaga, K., Furukawa, T., & Yasui, S. (2003). Modular network SOM: Extension of SOM to the realm of function space. *Proceedings of the 5th Workshop on Self-Organizing Maps (WSOM2003)* (pp. 173–178). Kitakyushu

31. Valkealahti, K., & Oja, E. (1998). Reduced multidimensional co-occurrence histograms in texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(1), 90–94.

32. Villmann, T., & Claussen, J.C. (2006). Magnification control in self-organizing maps and neural gas. *Neural Computation*, **18**(2), 446-469.

33. Walter, J., & Ritter, H. (1996). Rapid learning with parametrized self-organizing maps. *Neurocomputing*, **12**, 131-153.

34. Zhang, B., Fu, M., & Yan, H. (1998). Handwritten digit recognition by neural 'gas' model and population decoding. *Proceedings of the IEEE International Joint Conference on Neural Networks* (pp. 1727–1731). Alaska

35. Zhang, B., Fu, M., Yan, H., & Jabri, M.A. (1999). Handwritten digit recognition by adaptive-subspace self-organizing map (ASSOM). *IEEE Transactions on Neural Networks*, **10**(4), 939–945.

Figure Legends

Fig.1: The SOM^2 architecture.

Fig.2: Simulation results for the SOM^2 with artificial datasets.

Fig.3: The best matching map (BMM) and best matching unit (BMU) correspond, respectively, to the best matching section and best matching fiber in the fiber bundle.

Fig.4: Calculation flow of the SOM and SOM^2 algorithm.

Fig.5: Maps of the first artificial manifold set generated by a SOM^2. (a) The data vectors of 9 classes. (b) The reference maps generated by the SOM^2. The maps with thick lines are the BMMs of the 9 given classes. (c) The meta-map generated by the SOM^2. The arrowheads indicate reference vectors with the same index in each reference map, i.e., a fiber.

Fig.6: Maps of the second artificial manifold set generated by a SOM^2. (a) The data vectors of 400 classes. The data vectors of 9 of the 400 classes are indicated with larger markers and the manifold shapes. (b) The reference maps generated by the SOM^2. (c) The organized meta-map. The arrowheads indicate reference vectors with the same index in each reference map.

Fig.7: The maps of three datasets that are not homotopic. (a) Three datasets used in this simulation. (b) The meta-map organized by a SOM^2. (c) The meta-map organized by a NG$\times$SOM.

Fig.8: The results of the shape classification task. (a) 15 contours used in the simulation. (b) A map of the contours generated by a SOM^2. (c) A map of the contours generated by a SOM^3. The red dots show the original data overlaid on the best-matching reference maps.

Fig.9: A map of face images organized by a NG$\times$SOM using the DDR method. The one-dimensional meta-map space is indicated by the division into 3 lines. The BMMs of the input face images are indicated by circles.

Fig.10: Maps of 3D objects generated from 2D images by a SOM^2. (a) 2D images of 13 objects used in the simulation. (b) Meta-map of the objects generated by the SOM^2. For clarity, each box in the meta-map shows only one reference vector of the corresponding reference map. Two representative reference maps are also presented in full.

Fig.11: Maps of faces organized by a SOM^2. (a) Entire meta-map organized by a SOM^2. (b) Each fiber represents a map of faces, while each section, i.e., each reference map, represents the continuous change in viewpoints.

Fig.12: Maps of camera images of a mobile robot organized by a SOM^3. (a) Architecture of the SOM^3. (b) Work field built in the simulator WEBOTS. (c) Organized meta-meta-map created by the SOM^3 along with two representative reference meta-maps. The input images (16×16 images) are indicated at the best matching reference map in the meta-meta-map. For clarity, each image indicated in the meta-meta map corresponds to one of the reference maps of each meta-map with the same index; the two representative reference meta-maps are shown in full.

Fig.13: Homologization process of given manifolds by SOM^2.

Fig.14: Methods for determining the BMM. (a) $\mathbf{V}_B^*$ becomes the BMM for dataset X, using the least error criteria. (b) $\mathbf{V}_A^*$ becomes the BMM when the distance is measured by the class map.

Figures

The following captions and figures of reduced size are attached for the convenience of the reviewers. Due to the too large file size, the original figures are presented in `SOM2figs.pdf`.

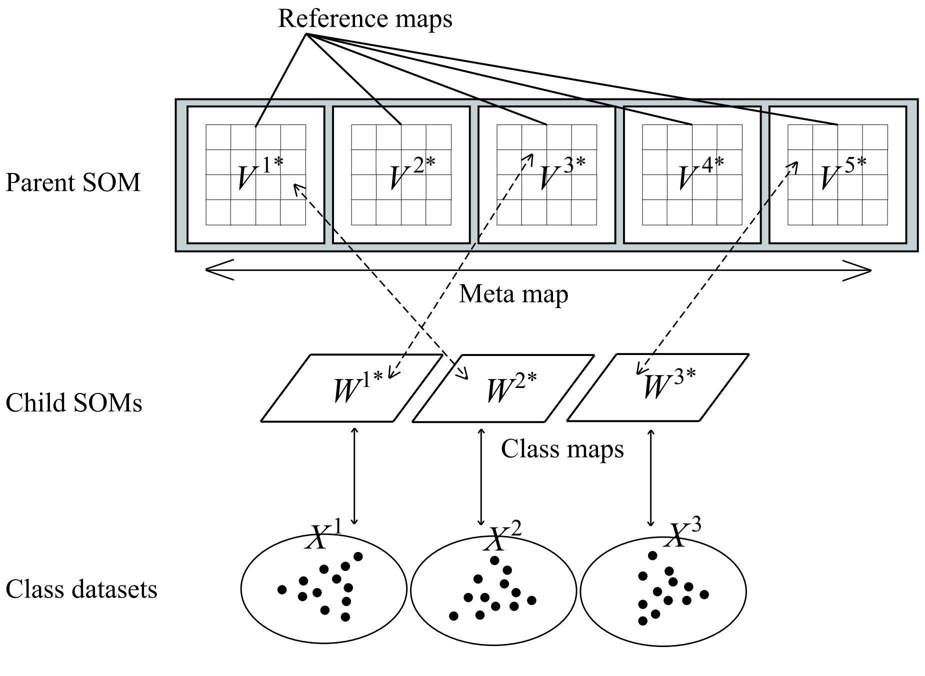


Fig. 1: The SOM^2 architecture.

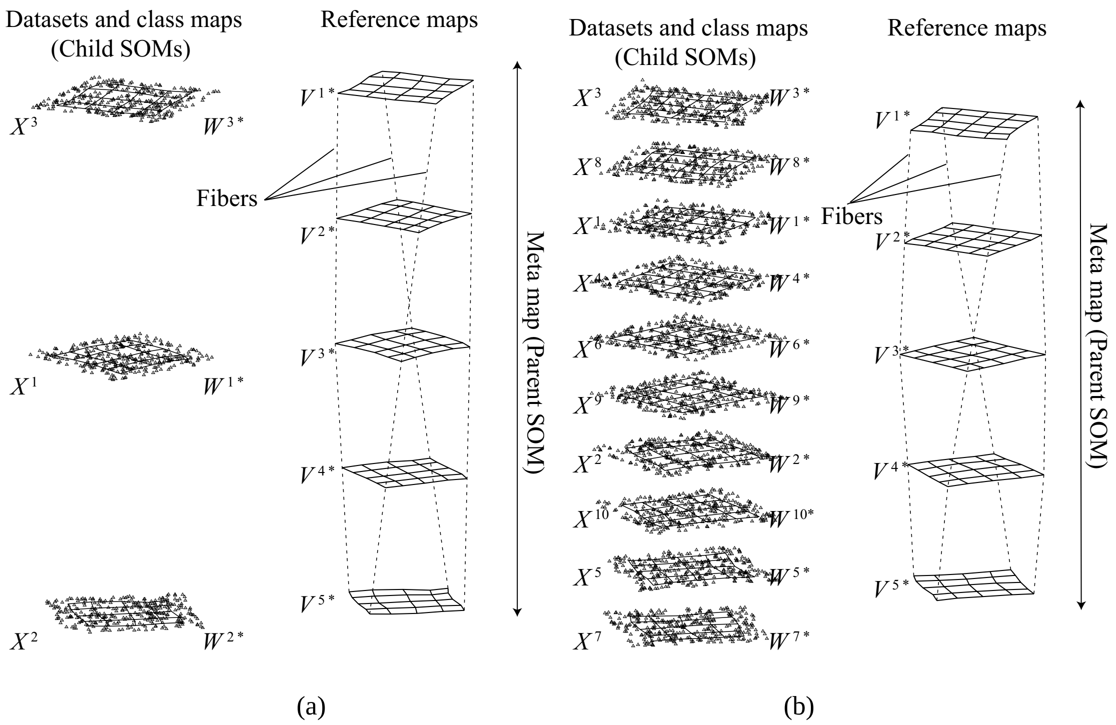


Fig. 2: Simulation results for the SOM^2 with artificial datasets.

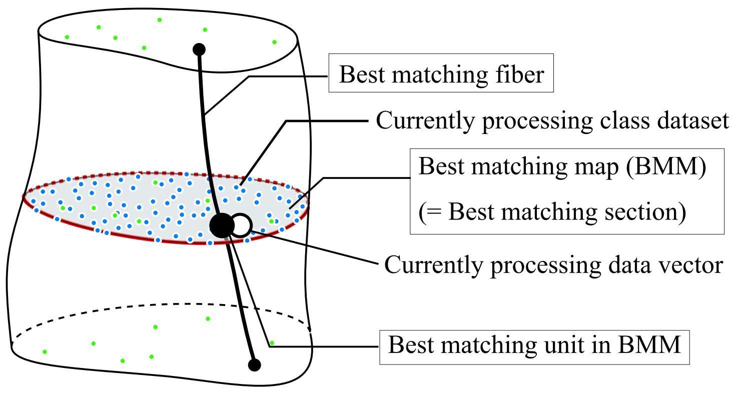

Best matching fiber
Currently processing class dataset
Best matching map (BMM)
(= Best matching section)
Currently processing data vector
Best matching unit in BMM

Fig. 3: The best matching map (BMM) and best matching unit (BMU) correspond, respectively, to the best matching section and best matching fiber in the fiber bundle.

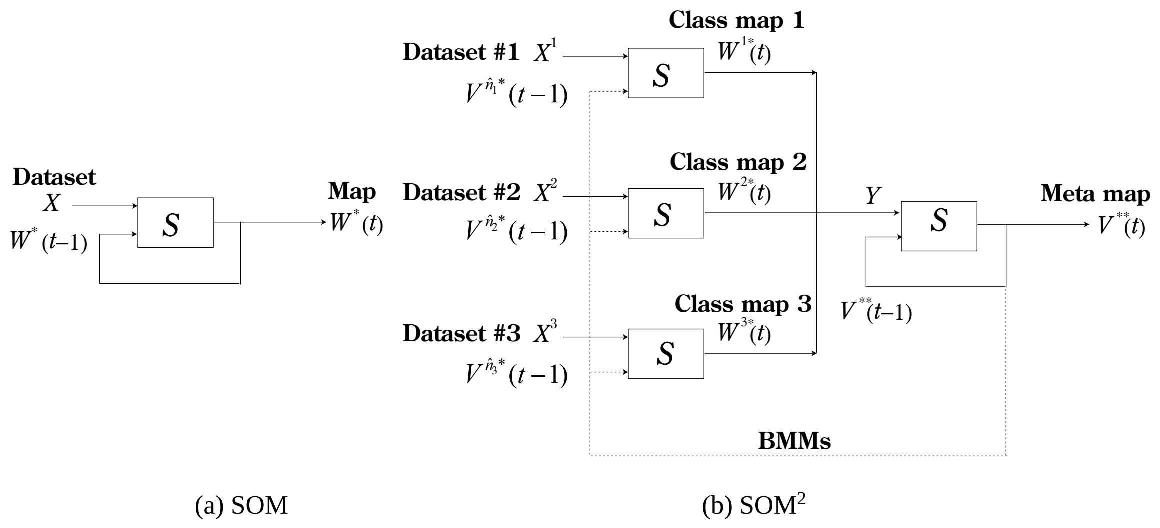
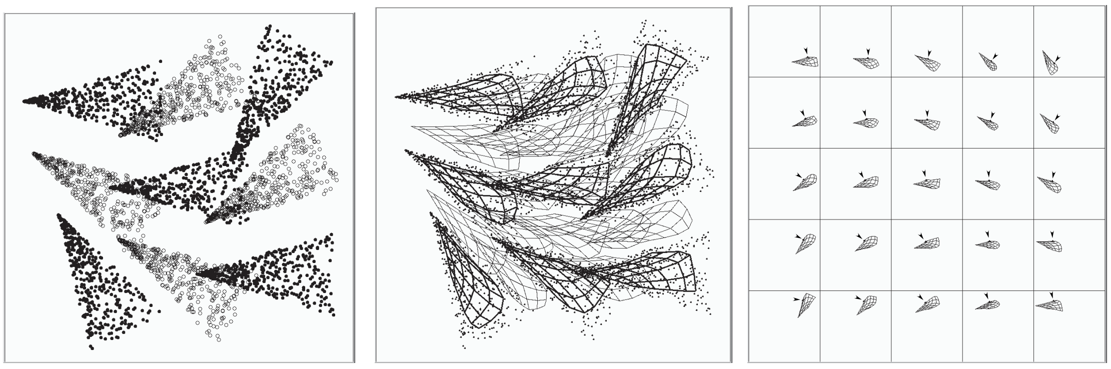
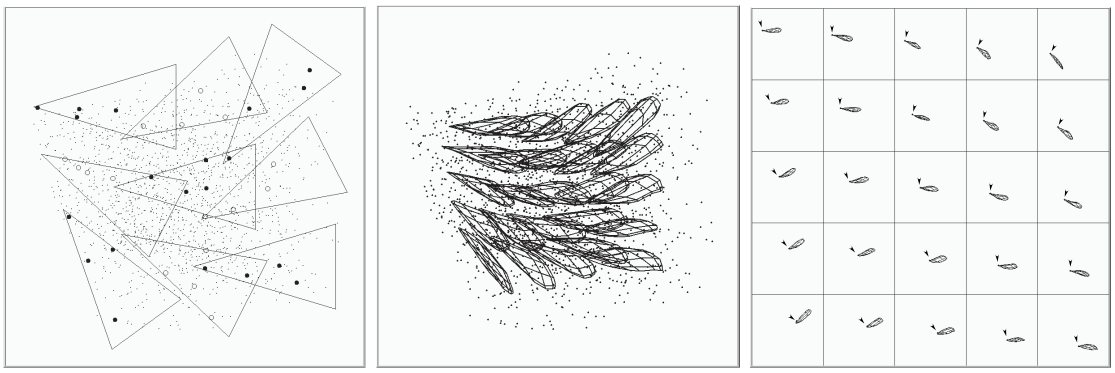


Fig. 4: Calculation flow of the SOM and SOM^2 algorithm.



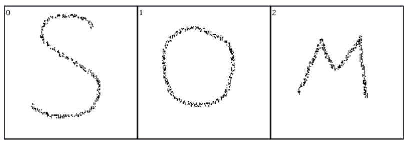
(a) (b) (c)

Fig. 5: Maps of the first artificial manifold set generated by a SOM^2. (a) The data vectors of 9 classes. (b) The reference maps generated by the SOM^2. The maps with thick lines are the BMMs of the 9 given classes. (c) The meta-map generated by the SOM^2. The arrowheads indicate reference vectors with the same index in each reference map, i.e., a fiber.

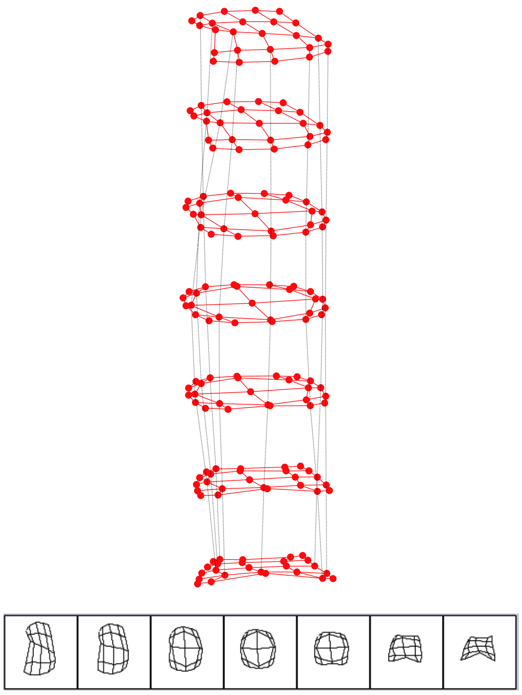


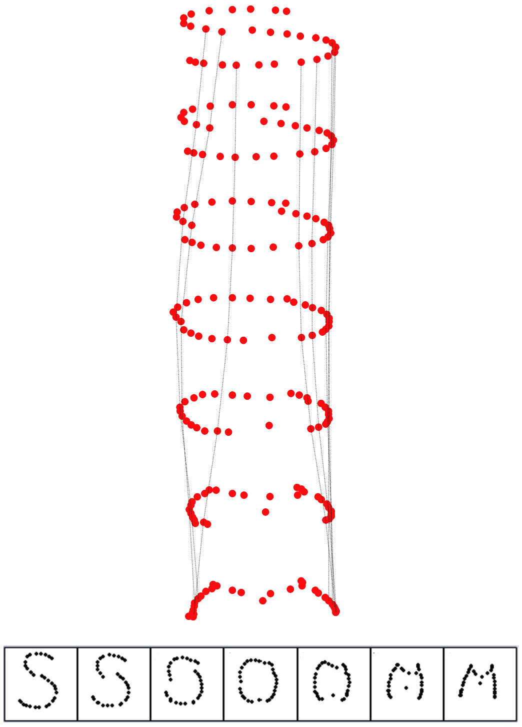
(a) (b) (c)

Fig. 6: Maps of the second artificial manifold set generated by a SOM^2. (a) The data vectors of 400 classes. The data vectors of 9 of the 400 classes are indicated with larger markers and the manifold shapes. (b) The reference maps generated by the SOM^2. (c) The organized meta-map. The arrowheads indicate reference vectors with the same index in each reference map.



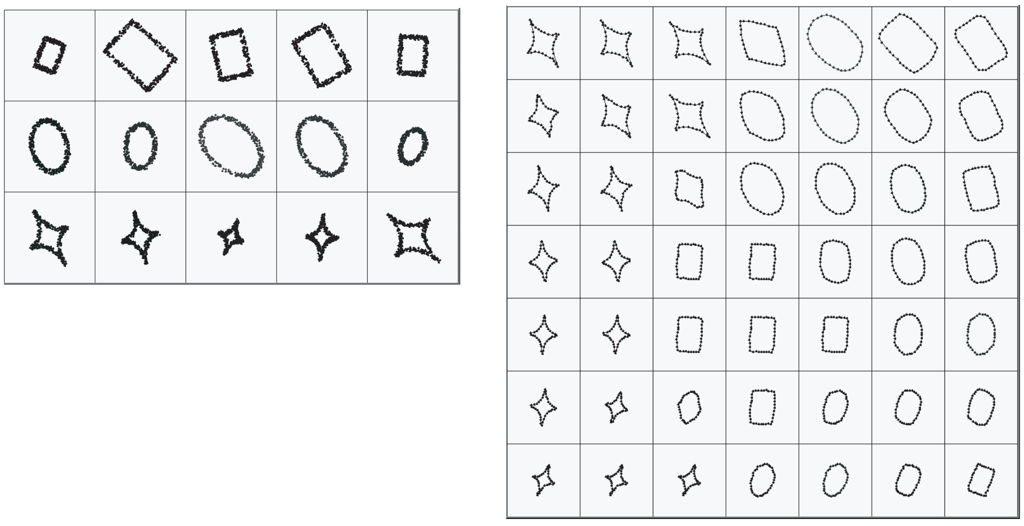
(a)





(b) (c)

Fig. 7: The maps of three datasets that are not homotopic. (a) Three datasets used in this simulation. (b) The meta-map organized by a SOM^2. (c) The meta-map organized by a NG×SOM.


(a) (b)

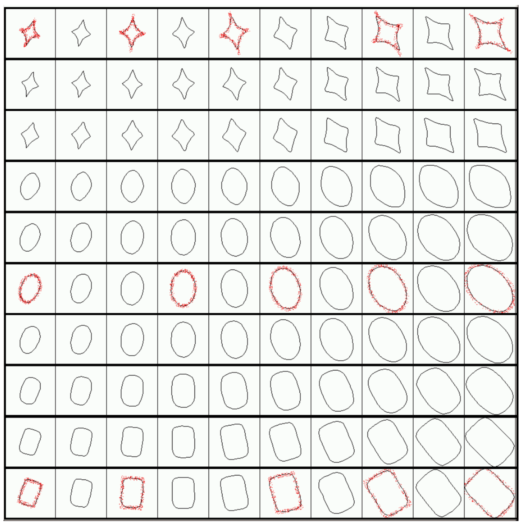

(c)

Fig. 8: The results of the shape classification task. (a) 15 contours used in the simulation. (b) A map of the contours generated by a SOM^2. (c) A map of the contours generated by a SOM^3. The red dots show the original data overlaid on the best-matching reference maps.

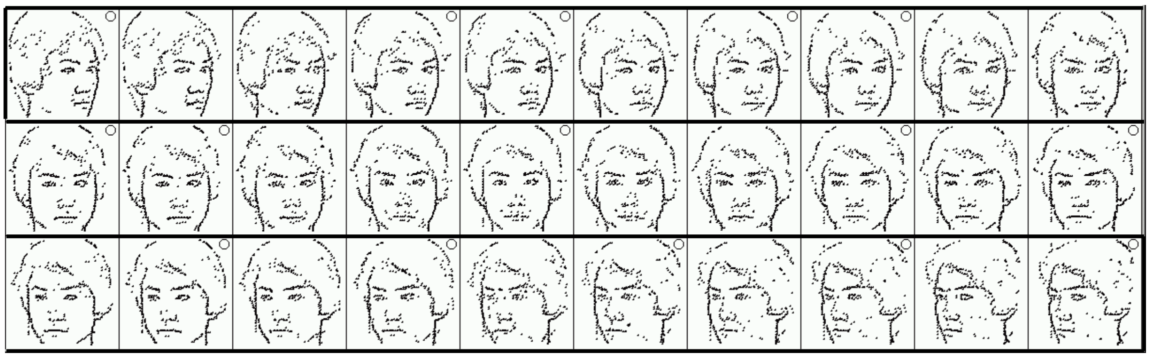
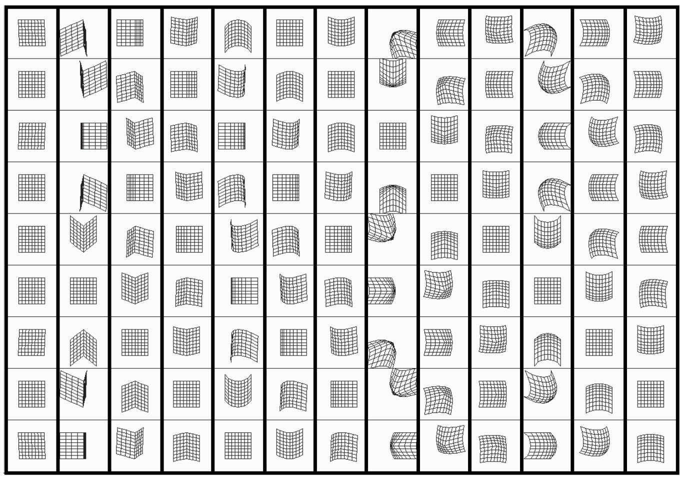
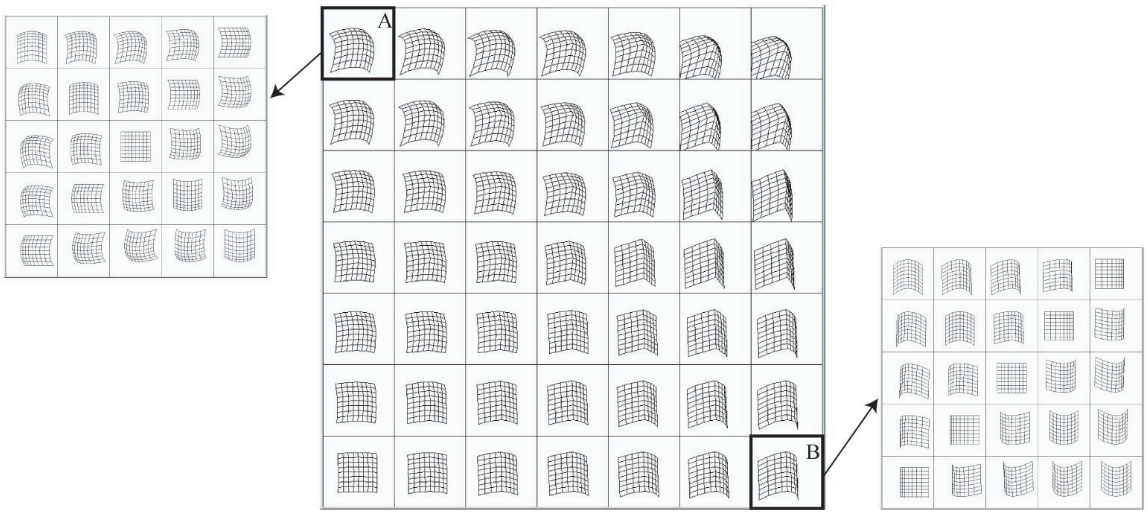


Fig. 9: A map of face images organized by a NG×SOM using the DDR method. The one-dimensional meta-map space is indicated by the division into 3 lines. The BMMs of the input face images are indicated by circles.

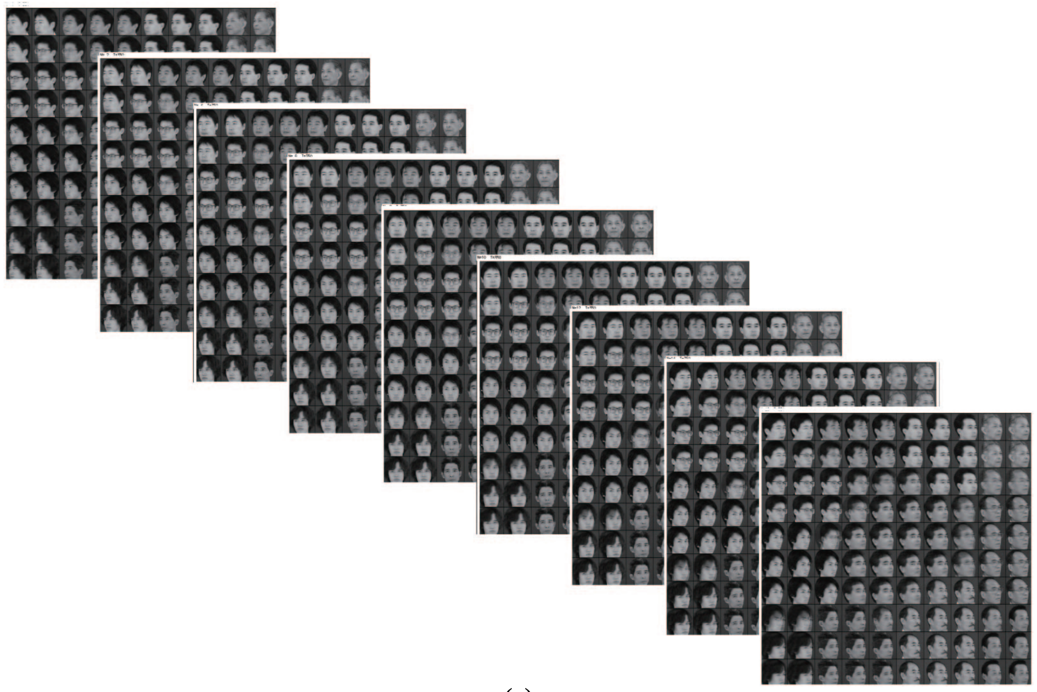


(a)

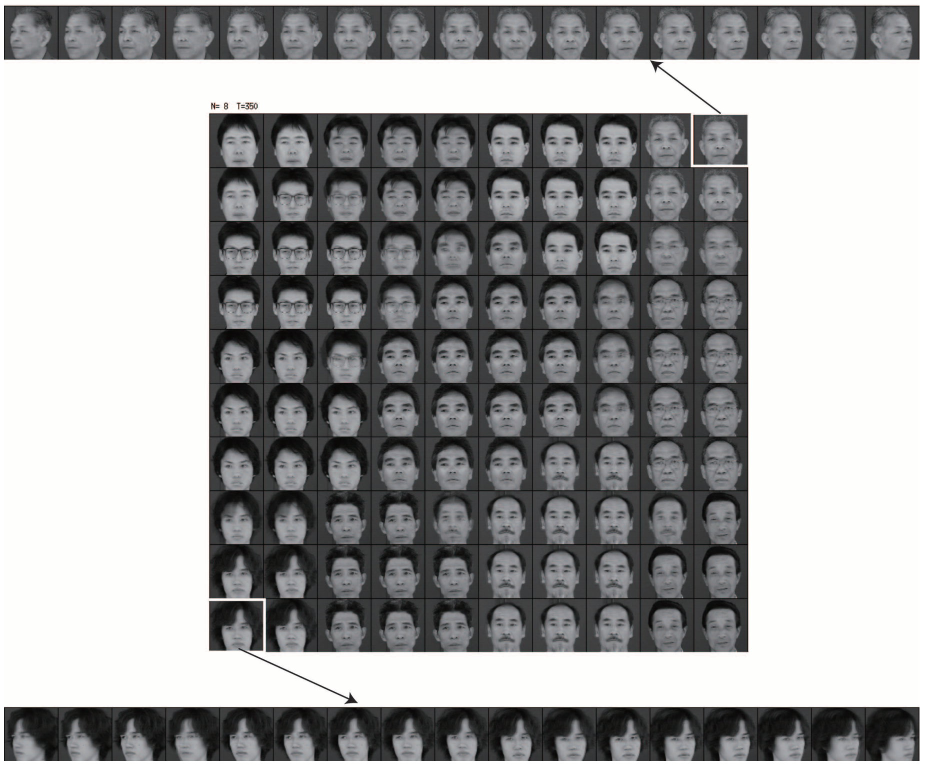


(b)

Fig. 10: Maps of 3D objects generated from 2D images by a SOM^2. (a) 2D images of 13 objects used in the simulation. (b) Meta-map of the objects generated by the SOM^2. For clarity, each box in the meta-map shows only one reference vector of the corresponding reference map. Two representative reference maps are also presented in full.

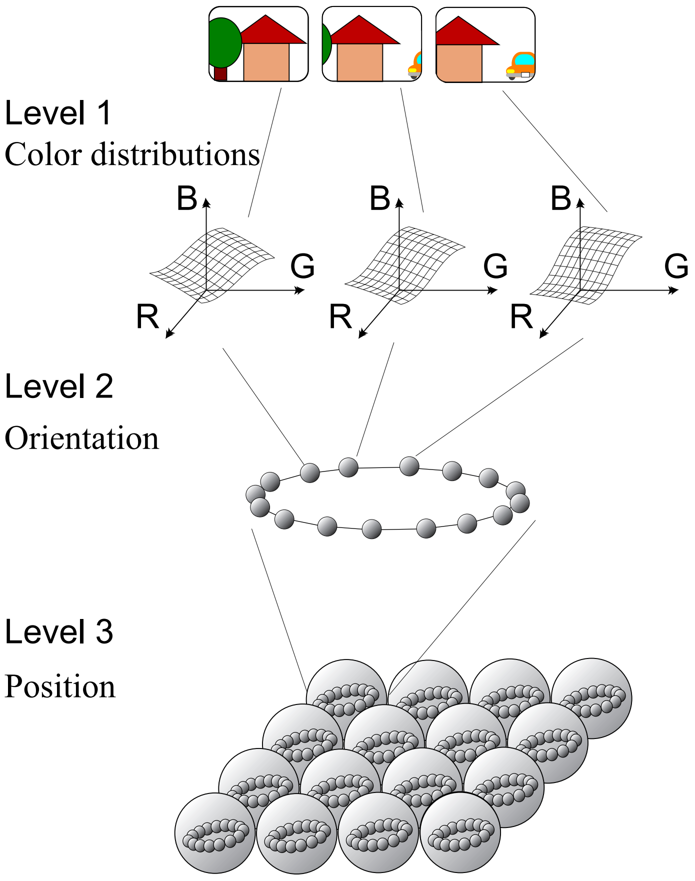


(a)

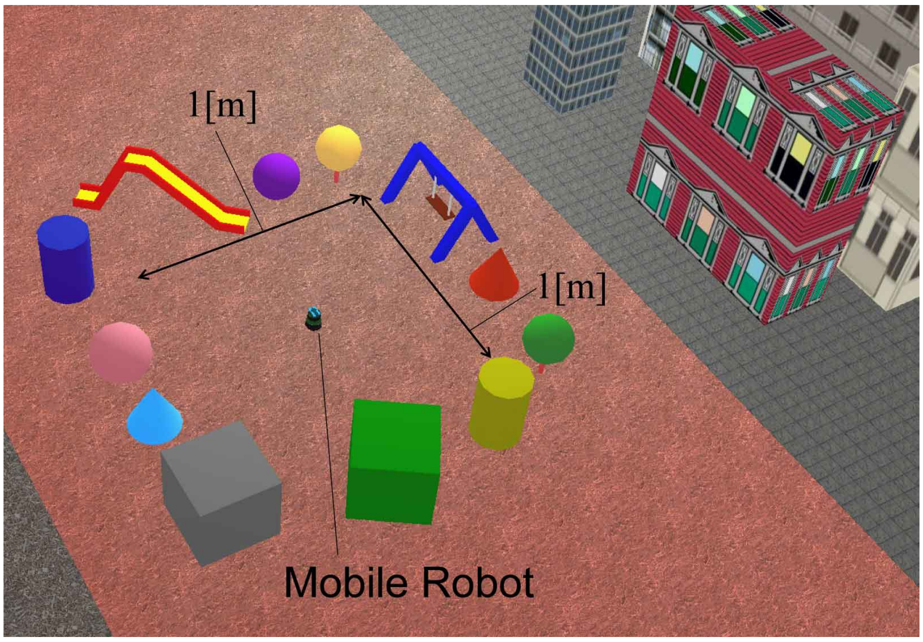


(b)

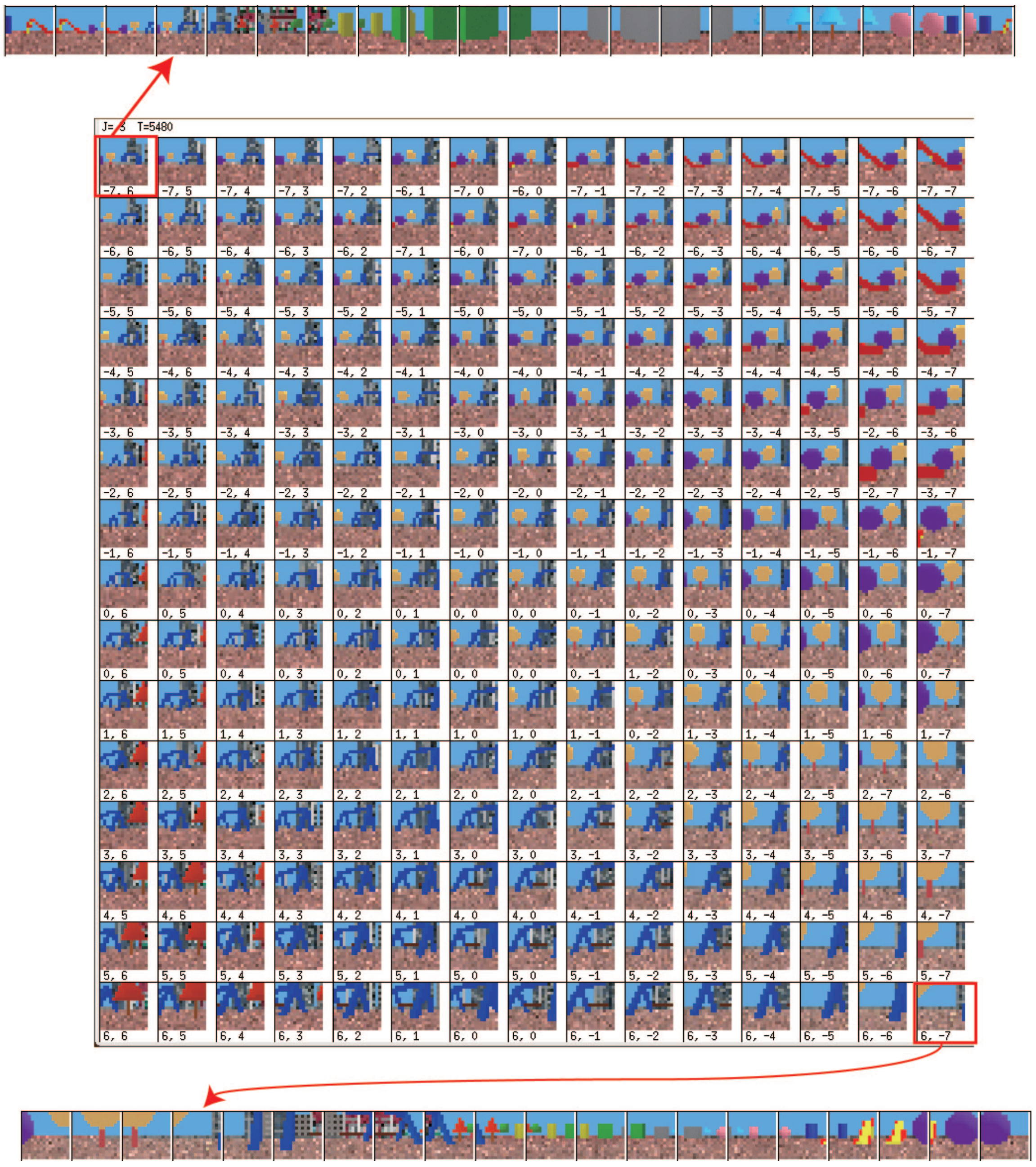
Fig. 11: Maps of faces organized by a SOM^2. (a) Entire meta-map organized by a SOM^2. (b) Each fiber represents a map of faces, while each section, i.e., each reference map, represents the continuous change in viewpoints.


Level 1
Color distributions
B
G
R
Level 2
Orientation
Level 3
Position

(a)

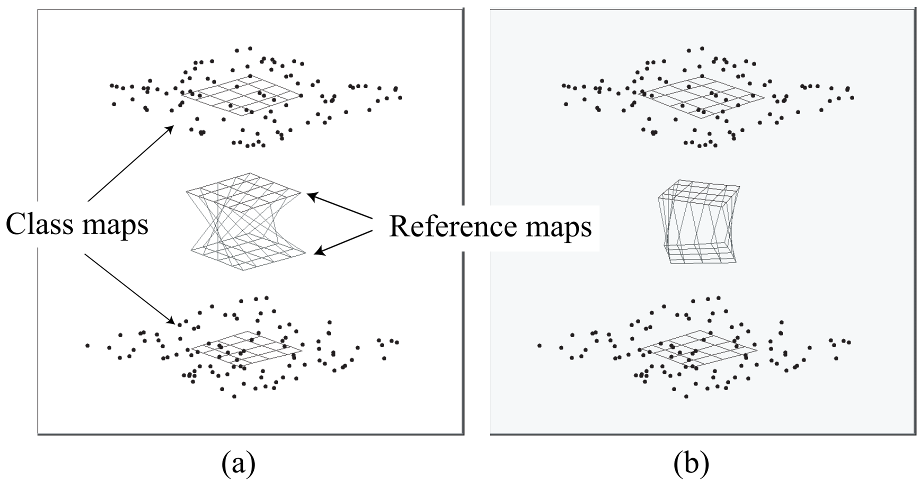

1[m]
1[m]
Mobile Robot

(b)



(c)

Fig. 12: Maps of camera images of a mobile robot organized by a SOM^3. (a) Architecture of the SOM^3. (b) Work field built in the simulator WEBOTS. (c) Organized meta-meta-map created by the SOM^3 along with two representative reference meta-maps. The input images (16×16 images) are indicated at the best matching reference map in the meta-meta-map. For clarity, each image indicated in the meta meta map corresponds to one of the reference maps of each meta-map with the same index; the two representative reference meta-maps are shown in full.



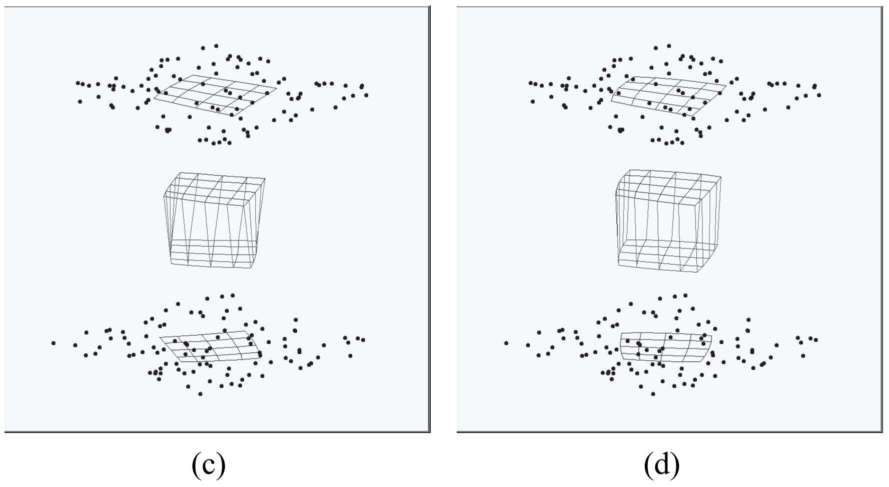


Fig. 13: Homologization process of given manifolds by SOM^2.

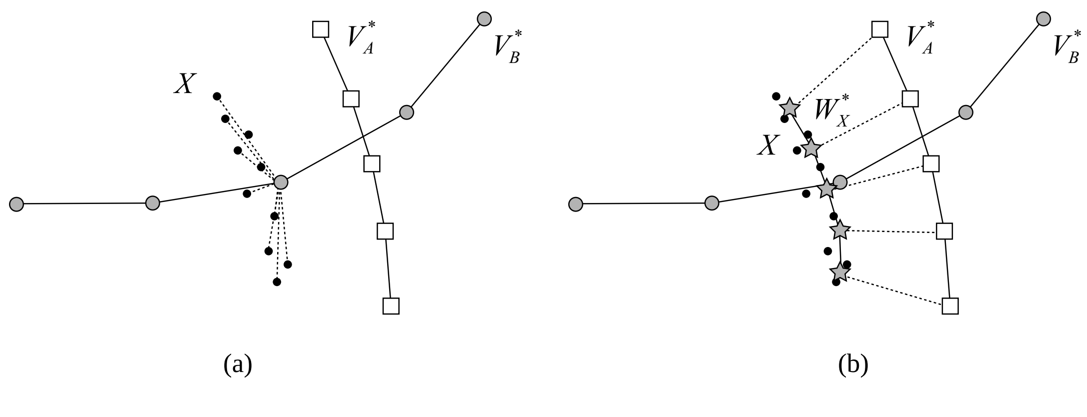


Fig. 14: Methods for determining the BMM. (a) $\mathbf{V}_B^*$ becomes the BMM for dataset X, using the least error criteria. (b) $\mathbf{V}_A^*$ becomes the BMM when the distance is measured by the class map.