Self-organizing topological tree for online vector quantization and data clustering

Papilinski, Andrew P.; Xu, Pengfei; Chang, Chip Hong

2005

Xu, P., Chang, C. H., & Papilinski, A. (2005). Self-organizing topological tree for online vector quantization and data clustering. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 35(3), 515-526.

<https://hdl.handle.net/10356/91435>

<https://doi.org/10.1109/TSMCB.2005.846651>

IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics © 2005 IEEE.
Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder. <http://www.ieee.org/portal/site>.

Self-Organizing Topological Tree for Online Vector Quantization and Data Clustering

Pengfei Xu, Chip-Hong Chang, *Senior Member, IEEE*, and Andrew Paplinski, *Member, IEEE*

Abstract—The self-organizing Maps (SOM) introduced by Kohonen implement two important operations: vector quantization (VQ) and a topology-preserving mapping. In this paper, an online self-organizing topological tree (SOTT) with faster learning is proposed. A new learning rule delivers the efficiency and topology preservation, which is superior of other structures of SOMs. The computational complexity of the proposed SOTT is $O(\log N)$ rather than $O(N)$ as for the basic SOM. The experimental results demonstrate that the reconstruction performance of SOTT is comparable to the full-search SOM and its computation time is much shorter than the full-search SOM and other vector quantizers. In addition, SOTT delivers the hierarchical mapping of codevectors and the progressive transmission and decoding property, which are rarely supported by other vector quantizers at the same time. To circumvent the shortcomings of clustering performance of classical partition clustering algorithms, a hybrid clustering algorithm that fully exploit the online learning and multiresolution characteristics of SOTT is devised. A new linkage metric is proposed which can be updated online to accelerate the time consuming agglomerative hierarchical clustering stage. Besides the enhanced clustering performance, due to the online learning capability, the memory requirement of the proposed SOTT hybrid clustering algorithm is independent of the size of the data set, making it attractive for large database.

Index Terms—Data clustering, online learning, self-organizing map (SOM), tree structure, vector quantization (VQ).

I. INTRODUCTION

SELF-organizing maps (SOMs) [18] fall into the class of unsupervised, competitive learning neural networks. One distinctive feature of SOM is its ability to form localized clusters (activity bubbles) around the maximally responsive neuron (winner) through competitive learning algorithm with only a simple layer of feed-forward synaptic connections and a competitive winner-take-all (WTA) neuronal layer. The location of each activity bubble is determined by the characteristics of the input data. A competitive neural network, in general has the lateral interactions in the WTA layer to produce the excitatory and inhibitory actions based on the distance from the most active output neuron. SOM has in addition a time-variable topological neighborhood of active neurons around the activity bubble. The winner and its neighborhood neurons that fall under the in-

fluences of the activity bubble have their synaptic weights adjusted toward the input stimulus. This gives SOM the topological ordering capability whereby the spatial location of a neuron in the lattice corresponds to a particular feature or a group of features of the source. It is this simple network topology and the unsupervised self-annealing property of SOM that underline its strength and interest over classical statistic approaches in emerging new applications that call for clustering of unlabeled multivariate data [7], [8], [11], [17].

A dichotomy in the applications of SOM is that of quantization [10], [12] in which the aim of data clustering is to obtain an optimal set of codebook vectors that maximizes the rate-distortion performance, versus clustering which aims to segregate a chaotic mixture of patterns for the purpose of knowledge discovery and analysis. Unsupervised learning allows the SOM to group datasets on the basis of their perceived closeness in a multivariate hyperspace. The synaptic weights of the trained network provide an approximation to the Voronoi vectors in a self-organizing manner. As the primary function of SOM is to visualize the topological relationship of input vectors, for pattern classification, the trained network needs to be calibrated by a supervised optimization technique called learned vector quantization (LVQ) [18] to minimize the average probability of misclassification.

However, the classical SOM algorithm suffers from several disadvantages in its application to clustering problems. First, the clustering result is sensitive to the number of partitions which has to be predefined. Second, the clustering result of one partition provides knowledge at only one similarity level. Third, SOM based algorithm also suffers from the problem of favoring “equally-sized compact spheroidal clusters,” i.e., isotropic clusters. This is caused by the mean square error (MSE) function and vector quantization (VQ) nature of SOM clustering. Actually, these limitations are also common to the classic k -mean and many other partition clustering algorithms. Many variants of SOM have been proposed to mitigate its weaknesses in classification ability. Growing SOM [1], growing Grid [9] and incremental grid growing [3] are a class of algorithms that overcomes the first problem by growing the SOM to a suitable number of partitions through the insertion of new neurons. Tree-structured SOM [20], [21] provides a hierarchical structure to reduce the computation complexity and alleviates the first two problems simultaneously. Recently, a grow hierarchical SOM (GHSOM) [26] is also proposed to grow a hierarchical SOM. To solve the third problem, some researchers generalize the model of SOM, e.g., SOMN [28], to a network of more generalized distribution models (e.g., Gaussian, delta) at the expense of increased computational complexity. However, in real applications, the data

Manuscript received July 22, 2003; revised March 1, 2004. This paper was recommended by Guest Editor J. Peng.

P. Xu and C. H. Chang are with the Centre for High Performance Embedded Systems, Nanyang Technological University, 639798 Singapore (e-mail: xupengfei@pmail.ntu.edu.sg; echchang@ntu.edu.sg).

A. Paplinski is with the School of Computer Science and Software Engineering, Monash University, Melbourne, Vic 3800, Australia (e-mail: app@ccse.monash.edu.au).

Digital Object Identifier 10.1109/TSMCB.2005.846651

distribution may not satisfy the distribution assumed a priori. Although many variants of SOM have been proposed, the three problems of SOM mentioned are far from being well solved, the last two problems, in particular.

In VQ, yet another dichotomy is based on whether the search for the best matching codevector is constrained or unconstrained. An unconstrained search algorithm, frequently used in the generalized Lloyd algorithm (GLA) [23], and the basic SOM, requires a full search over the entire set of codevectors to compute the distortion between an input data point and every codevector to locate a best matching codevector. As the computational complexity of the full search VQ is proportional to the number of codevectors, it becomes computationally impractical as the feature space or the number of clusters grows beyond a certain limit. It has been reported in [20], [21] that the basic N -neuron SOM algorithm using full search algorithm to find the best matching neurons takes about 10^4 to 10^5 iterations with $N \times M \times 3$ floating point operations to organize a M -dimensional input dataset. To reduce the encoder complexity, a constrained search procedure structures codevectors in such a way that only a small subset of the entire codebook is searched for the best representative codevector with minimal distortion. Among the many constrained search algorithms [12], tree search vector quantizer (TSVQ) [4] is one of the best known approaches to speedup vector quantizers. In general, fixed rate TSVQ performs worse than the full-search VQ. It is attributable to the accumulation of the bias introduced by the constrained search from the parent nodes to the child nodes. Improved performances have been observed in variable rate TSVQ [12], but direct application of the tree search algorithm on VQ lacks the important topology preserving property of SOM. The first tree-structured SOM (TS-SOM) was introduced by Koikkalainen and Oja in 1990 [20], and was later improved by Koikkalainen [21]. Their proposed TS-SOMs are organized layers by layers. That is, after one layer has been organized, its nodes are frozen and the organization proceeds on the next layer. A limitation of this type of TS-SOM is that it is not readily implementable as an online learning scheme. All training data are fed into the system repeatedly at every layer which taxes the system resources heavily for large database. In order to improve the performance of learning, TS-SOM uses a lateral search to lower the inherent bias present in the tree-structured search. However, this lateral search increases the computation load and annihilates the hierarchical and progressive mapping of codevectors along the binary- or quad-tree. Another Binary Tree-structured SOM (BT-SOM) was proposed by T. D. Chiueh *et al.*, for VQ [6]. They compare their VQ performance against GLA based TSVQ. However, the one-dimensional (1-D) binary tree model limits its scalability in processing time, and its degradation in performance due to the local decision making procedure against the basic SOM has not been evaluated.

In this paper, we propose a new multiresolution self-organizing topological tree (SOTT) to accelerate the search procedure. Unlike the BT-SOM and existing TS-SOM algorithm, in our proposed SOTT algorithm, neurons on different layers are organized concurrently. Two problems have to be overcome before the SOTT can be made amenable to online learning. First, the tree-structured search procedure is usually globally subop-

timal and fails to find the real best matching neurons during training. In other words, the “winner” [18] of the network is not necessary the closest neuron to the input vector. Second, by imposing the structural constraints with a hierarchically connected lattice, two kinds of neighborhood relationship co-exist in the network: the interlayer parent–child relationship and the intralayer sibling relationship. As the synaptic weights of the neurons are updated during training, it is a challenging problem to maintain both topological relationships.

A multipath search [5] can be used to overcome the first problem. In most tree-structured algorithms, the winner of one layer is selected from among the children of the winner of its upper layer. This has the drawback of creating localization of responses. The multipath search is used to overcome the bias introduced by a single path tree search algorithm. The performance enhances as the bias reduces effectively with increasing number of paths, at the expense of the increased computation burden. The multipath search has been proven to be a good method to control the tradeoff between the performance and computation complexity for large trees.

The second problem is more intricate. We tackle this problem by introducing a novel concept of the “winning path” as opposed to the winning neuron in the basic SOM. The neurons on the winning path are updated in accordance to the learning rates at each layer. The winners and their inter- and intralayer neighborhood neurons are updated by specifically designed learning rules to keep both the parent–child relationship and sibling relationship intact. There is still a dilemma: the closest neurons to the input vector on each layer usually do not lie on the same path. There is no easy way of deciding an optimal winner path in this situation. What we have proposed is to use a multipath search for a winner on the last layer, and the path to this leaf is deemed the winner path. In some applications, however, it is also important to consider the intermediate layers. We have introduced a special parameter to modulate the adaptations at different layers. The proposed winner path based updating procedure has an added advantage of mitigating the bias due to the limited-path search in a large tree. This is because with an appropriate modulation, the neurons on a winner path are more likely to learn by the same degree from the input data, so the distances between them will be reduced. Consequently, less bias will be introduced in the subsequent updates.

To effectively alleviate the three clustering problems associated with the basic SOM, we suggest the use of the proposed SOTT in hybrid clustering scheme. In hybrid clustering scheme, a low complexity partition clustering algorithm is first applied to reduce the large amount of data before the computational intensive agglomerative hierarchical clustering (AHC) algorithm is used to merge the resultant clusters progressively to obtain the dendrogram [2]. An AHC algorithm will progressively merge sets of data points, and yields a *dendrogram*, which is a nested series of partitions representing the merging progress of data points [15]. One important factor for hybrid clustering scheme is the linkage metric, which is a proximity measure used to merge subsets rather than individual points in AHC. The linkage metric directly influences the effectiveness and computation complexity of AHC. However, a majority of the existing effective linkage metrics require a time complexity of

$O(N^2)$ and can only be calculated after all patterns have been labeled. This makes AHC computational intensive comparing with partitional clustering techniques like k -mean and SOM. Hybrid clustering, nevertheless, inherits the efficiency of the partition clustering and the prowess of discrimination of AHC, and it had been successfully used in several clustering schemes before [2], [22], [26]. The proposed SOTT hybrid clustering is based on the SOTT learning and an AHC stage with a novel linkage metric. The computation of this novel linkage metric blends in well with the winning path selection phase of the SOTT, thus, it can be evaluated online and much faster.

The rest of this paper is organized as follows. The structure of the SOTT and the basic terminologies used are presented in Section II. In Section III, the proposed learning algorithm of the SOTT is described. A unified partition and AHC algorithm based on SOTT is presented and a new linkage metric for the proposed hybrid clustering on SOTT is discussed in Section IV. VQ of images is taken as an example to test the learning ability of the proposed SOTT. The results are compared with the basic SOM algorithm in Section V. The clustering performance of the SOTT hybrid clustering is also evaluated. Section VI concludes the paper.

II. STRUCTURE OF SOTT AND RELATED TERMINOLOGY

The simplest SOM network consists of an input layer connected to a one-dimensional linear array of neurons in the WTA layer. SOMs encountered in practice are more commonly structured with neurons typically arranged on a two-dimensional square or hexagonal grid laid out in a lattice. As the number of neurons increases, the growth of the search space calls for a multilayer SOM to ease the network from the time consuming training and mapping processes. Irrespective of the geometrical forms of the lattice, as the dimensionality of the data distribution increases, direct extension of the basic training algorithm to a stack of SOM layers is inadequate to cluster the input with a fixed rate TSVQ because the final codebook structure will become highly irregular and unpredictable. In this section, we describe the structure of a SOTT, and define some terminologies specific to the SOTT.

A static SOTT can be viewed as a multilayer SOM, with fixed depth and breadth. A SOTT realizes the mapping of an n -dimensional Euclidean space R^n onto a finite codebook space W

$$\Lambda : R^n \rightarrow W \quad (1)$$

where the input vector $x = (x_1, x_2, \dots, x_n) \in R^n$ and $W = \{w_{ij} \in R^n \mid i = 1, 2, \dots, L \text{ and } j = 1, 2, \dots, N_i\}$ define a finite set of codevectors, w_{ij} which are points in R^n . The structure of the SOTT network is characterized by L , the number of layers and N_i is the number of neurons at the i th layer for each $i = 1, 2, \dots, L$.

A two-dimensional (2-D) quad-tree structured SOTT with $L = 3, N_i = 4^{i-1}$ is shown in Fig. 1(a). Fig. 1(b) shows a 1-D string structured SOTT with the same number of neurons.

The number of layers L of an SOTT is called the depth of the tree. For a static SOTT, the number of children per neuron,

Fig. 1. Structure of SOTT: (a) 2-D quad tree. (b) 1-D string.

B , is fixed and is called the branching factor. The first layer with N_0 neurons is called the root. In general, the i th layer has $B^{i-1} \times N_0$ neurons. The last layer is called the leaf layer and it has $B^{L-1} \times N_0$ neurons. The neurons on the leaf layer are known as the leaf-neurons.

Two kinds of relationships exist in a SOTT: 1) the intralayer neighborhood relationship and 2) the interlayer parent-child relationship. Let G_i be a fully connected graph where its vertices and edges are defined respectively by the neurons and their interconnections at the i th layer. From the formulation of a near-neighbor mesh of a finite lattice, we define the k -neighborhood N_v of the neuron $v \in G_i$ as the spatial support of the position of v in G_i . A neuron, u is said to be in the k -distance neighborhood of the neuron v if there is a connected path from u to v and their minimal Euclidean distance, $\|u - v\| \leq k$. Furthermore, for each nonleaf neuron $v \in G_i$, a subtree H_v of neurons rooted at v with a branching factor of B can be found. A neuron u is said to be a child of v iff $v \in G_i$ and $u \in G_{i+1} \cap H_v$. The neurons of $G_{i+1} \cap H_v$ that have the same parent neuron v , are called the siblings.

It should be noted that, unlike the TS-SOM of [21] which uses a fixed neighborhood for the adaptation, the neighborhood $N_v(t)$ of a winning neuron, v for each layer is monotonically decreasing with the training epoch time, t . Using a fixed neighborhood or starting the training with a small neighborhood, results in a map which is not globally ordered and consists of small patches of local ordering between which the ordering direction changes discontinuously. The temporal function of neighborhood improves the spatial resolution of the map without losing the acquired global ordering through the proposed layer updating process. In addition, the winning neuron and its neighbors will not be updated in a uniform manner. The nonuniform function that alters the update of each weight vector within the winning neighborhood by an amount relative to its topographical distance from the winning neuron is called the neighborhood taper. Because of the parent-child relationship, the winning neurons of each adaptive layer in the winning path must inherit the weight vector of their parents. New parameters are introduced in the search for the winning path during training and the neighborhood tapers for updating the winning path are tailored to maintain this close coupling of the parent-child relationship.

III. TRAINING ALGORITHM OF SOTT

Training is a process whereby a set of input vectors are presented to the network one at a time to cause the synaptic weight vectors of a group of neurons in the lattice to be tuned. The

Fig. 2. Major steps of the SOTT training algorithm.

SOTT as defined earlier is a fully connected, feed-forward network for which the spatial arrangement of the trained neurons conveys significant meaning. Upon training, the parent neuron has similar but coarser resolution than its children and the siblings have similar synaptic weight vectors of finer resolution. The resolutions of the synaptic weights increase as the neurons descend from the root to the leaf layers. Within the same layer, neighboring neurons have similar weight vectors and as the neurons are further apart, their weight vectors become discriminable. Our proposed algorithm adapts the synaptic weight vectors of the SOTT via an online incremental learning scheme. The main steps involved in the training algorithm are illustrated in Fig. 2.

A. Initialization of SOTT

Because of the topological neighborhood of SOM, initialization of SOM is not as crucial as for some other clustering algorithms, e.g., GLA. A better initialization scheme will enable the network to converge faster by using priori knowledge of the input data distribution. For example, in VQ of images, the neurons of the 1-D string SOTT can be initialized with uniformly distributed weight vectors on the major diagonal of the input space

$$w_{ij} = [(j - 1/2) \times 255/N_i, \dots, (j - 1/2) \times 255/N_i]^T \quad \forall i = 1, 2, \dots, L \text{ and } j = 1, 2, \dots, N_i \quad (2)$$

where w_{ij} represents the weight vector of the j th neuron on the i th layer. This initialization method is very simple, but it provides a good neighborhood and parent-child relationship among all neurons on the lattice.

B. Butterfly Permutation for Input Randomization

Our proposed SOTT algorithm can be used for an online learning scheme, which is very useful for large training sets. However, online learning causes the learning performance to be order dependent, particularly when the training set contains a high degree of redundant information. For example, in image processing, the input data consists of pixel samples from a two dimensional coordinate system. Training the SOTT with a raster scanning order of the input pixels has a strong tendency to over-train some groups of neurons. A block based butterfly jumping sequence was used to subsample the pixels from each

$x_1 \backslash x_2$	0	1	2	3	4	5	6	7
0	0	32	8	40	2	34	10	42
1	48	16	56	24	50	18	58	26
2	12	44	4	36	14	46	6	38
3	60	28	52	20	62	30	54	22
4	3	35	11	43	1	33	9	41
5	51	19	59	27	49	17	57	25
6	15	47	7	39	13	45	5	37
7	63	31	55	23	61	29	53	21

Fig. 3. Butterfly permutation order for input of a 2-D coordinate system.

block to form different training sweeps by Pei and Lo [25]. This fractal scanning process does not produce a sequence of inputs with maximal randomization of the input data. The similarity between adjacent blocks' texture and spatial details are still preserved. To alleviate the order dependent learning problem and accurately reflect the learning ability for a given source, a global butterfly permutation sequence is used to present the spatially correlated input data from a multidimensional coordinate system. The aim is to let the neurons learn the characteristics of the training source as early as possible to prevent the performance from being degraded by order dependent learning. The butterfly permutation is defined by a mapping: $\pi : Z_I \rightarrow Z_J^n$ of an input order number $r \in Z_I$ to a n -dimensional coordinate system, $(x_1, x_2, \dots, x_n) \in Z_J^n$ where Z_J^n is a finite integer space which is bounded by $[0, 2^J - 1]^n$. Then, the butterfly permutation π can be expressed mathematically as

$$\begin{aligned} \pi(r) &= \left\{ (x_1, x_2, \dots, x_n) \in Z^n \mid x_j \right. \\ &= \sum_{i=0}^{L-1} 2^{L-1-i} (r_{ni} \oplus r_{ni+j}) \quad \forall j \neq n, x_n \\ &= \left. \sum_{i=0}^{L-1} 2^{L-1-i} r_{ni} \right\} \end{aligned} \quad (3)$$

where r_i is the i th bit of the binary representation of the decimal number r (r_0 is the least significant bit).

The presentation order, r of the pixels for a 2-D ($n = 2$) coordinate system, with $x_1, x_2 \in [0, 7]$ is shown in Fig. 3. This special case of the proposed butterfly jumping sequence corresponds to the localized butterfly jumping sequence for an image subblock of size 8×8 defined by [25].

Fig. 4 illustrates the difference in permutation effect of the blockwise localized jumping sequence proposed by [25] and our global butterfly jumping sequence. The permuted data are read back in the raster scan order to form a 2-D image to make the distinction in spatial characteristics visually perceivable. A gray scale ramp image and a natural image (Lena) are used for comparison. The results show that under the global jumping sequence, the training data are more evenly diffused to ensure that the maximally distinctive data are fed in succession. Consequently, the quantization results of images could be improved by as much as several decibels from those without the global jumping sequence.

Fig. 4. (a) Original image. (b) Image permuted by the localized butterfly jumping sequence. (c) Image permuted by the global butterfly jumping sequence.

C. Search for the Winning Path

The updating of the winning neuron and its neighborhood at each layer will not be performed until a winning path has been identified for each input. To trace the winning path, we need to search for a single winning leaf. Our proposed search algorithm uses two key parameters, λ and κ , to bias the competitiveness of some layers and emulate the positive effect of a multipath search. This is unlike the basic SOM, TS-SOM and other TSVQs, where the search for best matching units is performed layer-by-layer, in isolation. The proposed algorithm for the search for a best matching leaf neuron and the winning path is shown in the pseudo-code in Fig. 5. The overall idea of the algorithm in Fig. 5 is to find the winning_child neurons progressively on each layer, until a winning child at the leaf layer (win_leaf) is found. Then the path to the win_leaf is set as the winning path (win_path).

In Fig. 5, the subroutine **Compute_error**($x(t), i, v, e_v$) computes the distortion between the input $x(t)$ and the B children of the parent neuron, v at level i . The weight vector of a child neuron u of v is denoted by w_u and its error (distortion) is computed by the Euclidean distance $\|x(t) - w_u\|$ biased by a fraction of its parent error, e_v . The fractional constant, called the leakage ratio λ_i is introduced so that adjacent layers, instead of only one layer, are considered in finding the best matching neuron at the leaf layer. The value of λ is fixed at each layer but varying across layers to facilitate the control of performance tradeoff at different layers. By tuning the leakage ratio λ_i , some intermediate layers can be emphasized according to the specific requirement of the application. This can be useful in some applications like progressive compression and transmission. It has been demonstrated by simulation that a small sacrifice in performance of the leaf layer can enhance the performance of the intermediate layers.

Another key parameter, κ is introduced in the subroutine **Find_win_child** to produce an equivalent effect of a κ -path search algorithm. **Find_win_child** is a recursive routine that calls **Compute_error** to calculate the errors of the children of a set of input neuron-error pairs. The κ best matching neurons are used as input to search for the κ best matching successor neurons at the next layer until the leaf layer is reached. Searching for the best matching leaf neuron from among the children of a

```

Find_win_path( $x(t), \text{root}$ )
-- $x(t)$ : input at time  $t$ ;  $\text{root}$ : root neuron of SOTT;
begin
  initialize  $\kappa$ ; -- initialize path search factor
  for ( $i = 1$  to  $L$ ) initialize  $\lambda_i$ ;
  --initialize the level of emphasis at each layer
   $v = \text{root}$ ;  $e_v = \|x(t) - w_v\|$ ;
  -- compute error at root;
   $V_1 = (v, e_v)$ ;
  ( $v, e_v$ ) = Find_win_child( $x(t), 1, V_1$ );
  -- find winning children recursively until a
  winning leaf is found
  win_leaf =  $v$ ;
  --assign the win_leaf as the neuron-error pair
   $v$  returned by Find_win_child
  win_path = Trace_parent( $v$ );
  -- trace the winning path from the winning
  leaf to the root
end

```

```

Find_win_child( $x(t), i, V_i$ )
--  $x(t)$ : input at time  $t$ ;  $i$ : layer number;  $V_i$ : Set of
neuron-error pairs, ( $v, e_v$ ) at  $i$ -th layer;
begin
  if ( $i = L$ ) return first element of  $V_i$ ;
  -- return winning leaf neuron
  Initialize  $V_{i+1} = \emptyset$ ;
  -- begin with an empty set of winning neurons
  for (each ( $v, e_v$ )  $\in V_i$ )
     $V_{i+1} = V_{i+1} \cup \text{Compute\_error}(x(t), i, v, e_v)$ ;
    -- compute errors for all children of node  $v$ 
  Delete all except  $\kappa$  neuron-error pairs with
  smallest error values in  $V_{i+1}$ .
  return Find_win_child( $x(t), i+1, V_{i+1}$ );
end

Compute_error( $x(t), i, v, e_v$ )
--  $x(t)$ : input at time  $t$ ;  $i$ : layer number;  $v$ : neuron;
 $e_v$ : error of  $v$ ;
begin
  Compute  $e_{u_j} = \|x(t) - w_{u_j}\| + \lambda_i e_v \quad \forall u_j \in \text{child}(v)$ 
  return  $\bigcup_{j=1}^B (u_j, e_{u_j})$ ;
end

```

Fig. 5. Algorithm for the best matching leaf neuron and winning path search.

single best matching neuron at the upper-layer is suboptimal. This is because the weight vector of the final winning neuron at the leaf layer for a single path search is most probably not the closest to the input vector. The parameter κ provides the control necessary to minimize the bias caused by branching at the higher level of the hierarchy. The value of κ can be tuned to trade computational complexity for overall performance. From simulation results, SOTT performs much better with $\kappa = 2$ than with $\kappa = 1$. The performance of the SOTT with $\kappa = 2$ is almost the same as that of the full search method but its computation time is much shorter and it is only a little longer than that of $\kappa = 1$. If κ_i is made different for some layers, we have a more complicated but interesting dynamic multipath search.

The recursive procedure **Find_win_child** is called at the main routine **Find_win_path** with the root neuron and its error equal to the Euclidean distance from the input vector. Once the winning leaf neuron is found, the function **Trace_parent** traverses the winning path bottom-up via the parent neuron of the winning leaf neuron until the root neuron is reached.

D. Updating of Winning Path Neurons and Their Neighborhoods

Once the winning path Γ is found, all neurons along the winning path and their neighbors are updated by the following equations:

$$w_v(t+1) = w_v(t) + \alpha(m)[x(t) - w_v(t)] \quad \forall v \in \Gamma \quad (4)$$

$$w_u(t+1) = w_u(t) + \alpha(m)h(u, v, m)[x(t) - w_u(t)] \quad \forall v \in \Gamma \text{ and } u \in N_v(m) \quad (5)$$

where v is the winning neuron, and $N_v(m)$ denotes the neighborhood of v at the sweep number m . A sweep is a period during which the updating parameters remain constant. In our simulations, we set the number of input data in one sweep to be six times the number of neurons on the leaf layer. On average, each neuron on the leaf layer will be updated six times as the winning neuron in a sweep. The learning rate of the winning neurons at the m th sweep, $\alpha(m) = \alpha(0)k_1^m$, is a monotonic decreasing gain function of the sweep time. The initial learning rate, $\alpha(0)$ is usually set to a value less than 1. The learning rate for the neighborhood of the winning neuron is scaled by the taper function $h(u, v, m)$. This neighborhood taper is usually defined as a time-decreasing Gaussian function of the spatial distance between the winning neuron and its neighbors, i.e.,

$$h(u, v, m) = e^{-\frac{\|u-v\|^2}{\sigma_i(m)^2}} \quad \forall v \in \Gamma \text{ and } u \in N_v(m) \quad (6)$$

where $\sigma_i(m) = \sigma_i(0)k_2^m$ is the neighborhood width at the i th layer at the sweep number m . It defines the boundary of the lateral interaction between the winning neuron v and other neurons on the i th layer. The initial neighborhood width of the i th layer, $\sigma_i(0)$ is usually set to a value between 1 and $N_i/2$ where N_i is the number of neurons at the i th layer. Both the learning rate, $\alpha(m)$ and neighborhood width, $\sigma(m)$ decrease with m . The constant k_1 and k_2 are set to a value typically between 0.8–0.99.

The updating function at each layer is in essence, similar to the method proposed by Pei and Lo [25]. However, being a multilayer system, the neurons are likely to be dragged far away from their parents in the learning process as mentioned by P. Koikkalainen and E. Oja [20]. If there is no provision to preserve the correlation between the parent and children neurons, the tree structure will be destroyed, making the search for best matching neurons in the winning path more difficult and the overall distortion enlarged.

Maintaining both the intralayer relationship (the neighborhood relationship) and the interlayer relationship (the parent–children relationship) correct is important for achieving good performance of the proposed SOTT. To keep the neurons of the same family close during the entire training process, the following updating rules are imposed.

- 1) The initialization of neighborhood widths, $\sigma_i(0)$ at adjacent layers is made to be proportional to the ratio of the numbers of neurons on those two layers

$$\frac{\sigma_i(0)}{\sigma_{i-1}(0)} = \frac{N_i}{N_{i-1}}. \quad (7)$$

- 2) The children neurons will only be updated if their parent neuron is also updated.

- 3) The neighborhood neurons will only be updated if it is sufficiently close to the winning neuron of their layer. This is to minimize unnecessary computation with little or no effect as the neighborhood learning rate reduces exponentially with increasing $\sigma_i(m)$. A neighborhood neuron, u is considered to be close enough to the winning neuron, v_i at the i th layer if

$$\frac{\|u - v_i\|}{\sigma_i(m)} \leq \delta. \quad (8)$$

The value of δ is usually set to be between 1 and 10. In our simulation, we use $\delta = 5$. The neighborhood neurons satisfying both rules 2) and 3) will be updated with (5).

E. Convergence Criteria

At the end of each sweep, the weight vectors of the leaf neurons in the current and the previous sweeps are compared. If their difference is less than a small threshold value, then the training is completed. The convergence criterion can be stated as

$$\frac{\sum_{j=1}^{B^L-1} \|w_{Lj}(m) - w_{Lj}(m-1)\|}{B^L-1} \leq \varepsilon. \quad (9)$$

In our simulation, if the average square difference of the neuron weights, w_{Lj} at the leaf layer is less than 0.01, the training is terminated.

IV. HYBRID CLUSTERING ALGORITHM ON SOTT

As mentioned in the introduction, the ability of SOM in clustering suffers from three major disadvantages that limit its effectiveness. To unbridle the limitations, hybrid clustering scheme [2], [22], [27] based on SOTT and a novel linkage metric is applied for enhanced clustering performance. The main idea behind the hybrid clustering is to combine the efficiency of the partition clustering and the prowess of discrimination of AHC. The general concept of hybrid clustering is illustrated in Fig. 6. To differentiate the clusters formed in different stages, the clusters formed by the partition clustering are called the atomic clusters since they are indiscernible in the later AHC stage, whereas the clusters merged during the AHC stage are called the interim clusters. At the outset of the AHC, the interim clusters C_i is the same as the atomic clusters, A_i . An iterative process is carried out to optimally merge the interim clusters greedily. At each step, two interim clusters C_i and C_j with the maximum *linkage metric* are sought for merging into a larger cluster. The process of maximum link search repeats until a predefined criterion is met.

As mentioned earlier, the linkage metric is the key factor for AHC stage. To merge clusters rather than individual points, the distance between individual points has to be generalized to the distance between clusters (sets of points). Such derived proximity measure is called a linkage metric. The type of linkage metric used influences the AHC algorithms and their clustering results and efficiency significantly, since it reflects the *closeness* and *connectivity* between clusters, and usually the computation of linkage metric is the major bottleneck of AHC. Traditionally, the dissimilarity measure (usually, the Euclidean

Fig. 6. Overview of hybrid clustering algorithm (a) N samples, (b) k atomic clusters, and (c) final clusters.

distance) is calculated for every pair of points with one point from each cluster. Then, a specific operation such as minimum (single link), average (average link), or maximum (complete link) is applied to the pair-wise dissimilarity measures. A majority of the existing effective linkage metrics suffers from the $O(N^2)$ time complexity [2], [15]. Besides, they must be calculated only after all patterns are labeled. In view of this, we introduce a new linkage metric for our proposed hybrid clustering algorithm on SOTT.

Before introducing the novel linkage metric, a metric $\text{Bond}(A_i, A_j)$ to assess the connectivity of two atomic clusters, A_i and A_j is defined as follows:

$$\text{Bond}(A_i, A_j) = \sum_{\forall p | \{w_1(p), w_2(p)\} = \{i, j\}} e^{1 - \frac{d_2(p)}{d_1(p)}} \quad (11)$$

where p represents a data point in the data set, $w_i(p)$ represents the neuron that has the i th minimum distance to p , and $d_i(p)$ represents the distance between $w_i(p)$ and p . The data points that fulfill the criterion for the summation, $\forall p | \{w_1(p), w_2(p)\} = \{i, j\}$ are those that cause neurons i and j to become the winner and runner-up neurons, in permutable order.

The proposed linkage metric, $\text{Link}(C_i, C_j)$ is defined as follows:

$$\text{Link}(C_i, C_j) = \max_{S_k \subseteq C_i, S_l \subseteq C_j} \frac{\sum_{A_m \in S_l} \sum_{A_n \in S_k} \text{Bond}(A_m, A_n)}{|S_k \cup S_l|} \quad (12)$$

$$|S_i| = \sum_{A_j \in S_i} |A_j| \quad (13)$$

where C_i represents the i th interim cluster, and $S_k \subseteq C_i$ is a subcluster of C_i .

In traditional effective linkage metric, such as single-link, complete-link, and average-link, the computation complexity for $\text{Link}(C_i, C_j)$ is $O(N_i \times N_j)$, while N_i and N_j represents the number of data points in the interim clusters C_i and C_j , respectively, since the computation involves all the distances between two data points in the interim clusters C_i and C_j . On the other hand, the proposed linkage metric in (13) only involves the $\text{Bond}(A_i, A_j)$ of the atomic clusters instead of all data points in the pair of interim clusters (C_i, C_j) . However, the calculation for $\text{Link}(C_i, C_j)$ of (13) is still quite involved since the bonds of all possible combinations of the subclusters, S of each interim cluster must be calculated to seek for the maximum link. In our simulation, we use a greedy growing of the subclusters to find the local optimal S_m and S_n for calculating the value of $\text{Link}(C_i, C_j)$. So the computational complexity is $O((k_i + k_j)k_i k_j)$, while k_i and k_j represent the number of atomic clusters in the clusters C_i and C_j , respectively.

Fig. 7. The 512×512 gray-level test images. (a) Baboon. (b) Bridge. (c) Bird. (d) Lena. (e) Goldhill.

The most noticeable feature of SOTT hybrid clustering, which differentiates it from other hybrid clustering schemes and makes it attractive in a practical viewpoint, is the online clustering feature. With the online SOTT of the first stage, one data point is processed and discarded at a time. Only one data pass is needed to obtain the result, and the training process is resumable. In the second AHC stage, the linkage metric only need $|A_j|$ and $\text{Bond}(A_i, A_j)$ for each atomic cluster rather than all the data points in the atomic clusters, and the $|A_j|$ and $\text{Bond}(A_i, A_j)$ can be updated online during SOTT learning. Thus, the AHC is very fast and can be carried out at any preset interval of the SOTT. Furthermore, due to the online learning nature, the memory requirement of the proposed algorithm depends on the structure of SOTT rather than the size of the data set. These properties are very appealing for large database clustering [2].

V. SIMULATION RESULTS

In this section, we measure the performance of the proposed SOTT in VQ of gray-level images and compare it to the performance of the basic SOM, GLA [23], and GLA-based balanced TSVQ [4] algorithms. The ability of SOTT quantizer to preserve the topological property of the image is also studied. Additionally, its performance in multiresolution mapping is investigated. The two SOTT specific parameters, namely the leakage ratio, λ and the path search factor, κ are tuned to evaluate their effect on the network's learning ability and its computational efficiency.

In our simulations, the training data is fed from the test images to the SOM and SOTT networks using the butterfly permutation order described by (3) of Section III-B until the networks converge. The test images of size 512×512 are shown in Fig. 7.

To accurately reflect how well the neural networks have learnt from the training data, the testing and training data are obtained from the same source, and the search method is kept the same in both the training and testing phases. For fair comparison, the convergence criteria of GLA and GLA-based balanced TSVQ are set to be the same as in the SOM and SOTT algorithms. The basic SOM used is essentially a one-layer SOTT with the same codebook size (i.e., the number of neurons is the same as the number of leaf neurons of SOTT), and the common parameters of the two networks under test are set to the same values. The parameters for these two networks are listed in Table I. Among

TABLE I
TEST PARAMETERS OF SOM AND SOTT

Parameters	κ	λ	δ	$\alpha(0)$	K_1	K_2
Value	2	0.2	5	0.99	0.95	0.85

these parameters, κ , λ , and δ are used to tune the SOTT and they are not used on the original SOM.

The quality of the reconstructed image is measured using the *peak signal-to-noise ratio* (PSNR) in decibels. The PSNR is defined as follows:

$$\text{PSNR} = 10 \cdot \log \frac{255^2}{E[\hat{x} - x]^2} \quad (14)$$

where x and $\hat{x}$ represent the original input vector and the quantized vector, respectively. $E[x]$ denotes the expected value of x .

The performance results of the two networks in terms of the PSNR of the reconstructed image of Lena, and the training time for different codebook sizes are tabulated in Table II. The training time for both networks is measured on a Personal Computer equipped with a Pentium Pro, 2.1 GHz Processor and 512 MBytes of system memory. 1-D string neural topology with branching factor $B = 4$ is used for the SOTT. The first layer of SOTT is either 1 or 2 depending on the codebook size.

The best reconstruction quality and the fastest training time are highlighted in bold and italic print in Table II. As expected, when the number of neurons is small (<32), there is a small penalty in the training time of SOTT due to its structural overhead. As the network size increases, the SOTT speeds up dramatically. The acceleration over the basic SOM is more than ten times for large codebook of size greater than 512. The quality of reconstructed images, based on the codebooks generated by both networks, are compatible.

To better assess the differences in reconstruction performances, different images with varying degrees of textures and diverse spatial distribution of gray-levels are tested on both networks with the same codebook size of 256. The resulting PSNR of the reconstructed images and training time are shown in Table III.

Out of the five test images, the qualities of the reconstructed images of Lena, Bird, Bridge, and Goldhill for the SOTT degrade by about 0.5 dB compared with those of the full-search basic SOM. The deterioration of PSNR of Baboon for SOTT is the worst (more than 1.5 dB). This is attributable to the high frequency of intensity and texture variations of Baboon.

To visualize the topological property of SOTT, the red, green, and blue component values of every pixel in a color image (Lena), is fed into the SOTT for training. After the network has converged, the weights of the neurons at each layer are shown in the r-g-b feature space in Fig. 8. The topological relationship between neighboring neurons is visualized by connecting lines of different styles to distinguish the different layers. Fig. 9 shows the visual effect of the topological structure of the basic SOM.

It can be seen from Fig. 8 that every layer in the SOTT can be considered as a single SOM, which has its own topology comparable to a single SOM. The tightly coupled parent-child topographical relationship is evident from Fig. 8.

In the 1-D string neighborhood structure, the topology order can also be measured by an objective function J , which is defined in [18] as follows:

$$J = \sum_{i=2}^N \|w_i - w_{i-1}\| - \|w_N - w_1\| \quad (15)$$

where N is the number of neurons and w_i is the weight vector of the i th neuron. The lower the value of J , the better the codebook is ordered.

In our simulations, we use this function to compare the topology order of the basic SOM and the leaf layer of SOTT. The result is plotted in Fig. 10. It can be concluded that their topology order is comparable.

The multiresolution progressive decoding performance of the SOTT is evaluated with a network of branching factor $B = 4$ and depth $L = 5$. The leaf layer has 256 neurons. The images reconstructed from different layers of the SOTT are shown in Fig. 11.

Different leakage ratios, λ are experimented on Lena and the quantization results of different layers of the SOTT are presented in Table IV. The multiresolution performance of TSVQ is also listed for comparison. For ease of reference, PSNRs of Lena trained by the basic SOM for the corresponding codebook sizes from Table I are reproduced in the last column of Table IV.

In Table IV, the best PSNRs for each codebook size are printed in bold and italic. The overall quality of the image reconstructed from the codebooks obtained at different layers of the SOTT for a fixed λ is comparable to that obtained from several independently trained basic SOMs of equivalent codebook sizes. It should be noted that codebooks of different resolutions are generated simultaneously with SOTT through only one pass of training with nonexhaustive search of the best matching neurons as opposed to training $L - 1$ basic SOMs independently with the full search algorithm. The leakage ratio improves the performance of lower resolution codebook at the expense of a trivial degradation to the performance of the higher resolution codebook.

Different values of the multipath search factor, κ of the SOTT are also tested to evaluate how the winning path search factor influences the performance of the SOTT. The resulting PSNRs and training time are shown in Table V.

$\kappa = 1$ is the same as the single path tree-structured search algorithm. Comparing with $\kappa = 1$, $\kappa = 2$ produces the much better performance with a slight increase in training time. $\kappa = 3$ gives the optimal performance. There is little or no advantage in increasing the path search factor beyond $\kappa = 3$.

There are occasions when the performance of the SOTT becomes a little worse as κ increases (while maintaining the condition that the encoding κ is still higher than the training κ). This is not surprising as the performances of SOM and SOTT reach the margins of diminishing return near convergence, they become highly sensitive to the variation of parameters and input data distribution. Although the intuitive interpretation of the network behavior relative to its parameter variations is clear, there is no rigorous mathematical analysis of the dynamics of the self-organizing algorithms and their proof remains yet to be discovered.

TABLE II
PSNR (DECIBEL) AND TRAINING TIME (s) WITH DIFFERENT CODEBOOK SIZES FOR LENA IMAGE

Codebook Size	SOM		SOTT		GLA		TSVQ	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
4	24.95	0.08	24.95	0.08	25.00	2.5	25.00	2.5
8	28.48	0.13	28.49	0.18	28.69	10.5	28.16	4.6
16	30.60	0.27	30.25	0.31	30.58	44.3	30.48	5.3
32	32.38	0.86	32.23	0.76	32.40	59.1	32.22	11.8
64	33.91	2.53	33.70	1.62	33.85	124.5	33.58	13.3
128	35.40	9.44	35.22	3.89	35.35	184.9	34.92	12.9
256	36.94	35.80	36.55	9.00	36.80	555.2	36.22	16.3
512	38.39	147.20	37.96	22.01	38.14	1062.6	37.72	19.9
1024	39.95	604.13	39.45	64.73	39.50	1745.5	39.06	19.0

TABLE III
PSNRs (DECIBEL) AND TRAINING TIME (s) OF SOM AND SOTT FOR DIFFERENT IMAGES

Image	SOM		SOTT		GLA		TSVQ	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Lena	36.90	35.80	36.55	9.00	36.71	305.6	36.09	10.2
Bird	42.33	39.22	41.60	8.73	40.02	343.8	41.22	9.8
Bridge	31.54	43.95	31.19	9.18	31.33	401.8	30.75	18.5
Goldhill	35.46	42.88	34.87	9.28	35.18	355.8	34.22	12.9
Baboon	28.83	40.83	27.02	10.70	28.33	492.2	26.78	16.1

Fig. 8. One-dimensional string color map of a three-layer SOTT.

Fig. 9. One-dimensional string color map of SOM.

Tuning of these parameters is highly empirical and is best optimized with a specific application through a long series of simulations. Hence, once the network has been tuned for a minimal acceptable level of performance (quality) for a domain-specific application, computational complexity, resource requirements,

Fig. 10. Topology order J of SOM and SOTT.

Fig. 11. Lena images reconstructed from different layers of SOTT.

implementation simplicity and other features will be the prevailing factors. In this perspective, SOTT has a clear competitive advantage over the basic SOM and GLA algorithms owing to its fast learning, fast encoding and multiresolution mappings of progressively enhanced qualities. When comparing with GLA and TSVQ, SOTT has the advantages of the graceful topological ordering feature and online learning capability.

Several datasets are used to simulate the clustering performance and computation efficiency of the proposed SOTT based hybrid clustering algorithm to compare them with those achieved by the contender clustering algorithms. Fig. 12(a)–(d) shows four different 2-D data sets used in the simulation. These data sets are chosen, partly because similar data sets have been used extensively by other researchers [13], [16], [22], and partly because it is easier to evaluate the quality of the clustering results on 2-D data sets. Due to the different geometrical shapes and densities of the clusters, and the scattering noise in these data sets, traditional algorithms like the k -mean and AHC will fail to find the proper clusters as proven in [13], [16]. CURE has been tested to prove that it failed in clustering the data sets (b) and (c) [16], as seen in Fig. 12(e) and (f). Spectral clustering,

TABLE IV
PSNR (DECIBEL) OF LENA IMAGE RECONSTRUCTED WITH DIFFERENT LAYERS AND LEAKAGE RATIOS

Layer	Size	SOTT						TSVQ	SOM
		$\lambda=0$	$\lambda=0.05$	$\lambda=0.10$	$\lambda=0.15$	$\lambda=0.20$	$\lambda=0.25$		
2	4	23.59	24.46	24.56	24.70	24.75	24.79	25.00	24.95
3	16	30.32	30.26	30.40	30.36	30.49	30.50	30.65	30.60
4	64	33.37	33.49	33.52	33.61	33.66	33.62	33.51	33.91
5	256	36.41	36.46	36.48	36.58	36.55	36.45	36.09	36.94

TABLE V
PSNR (DECIBEL) AND TRAINING TIME OF LENA USING SOTT WITH DIFFERENT VALUES OF κ

κ	PSNR (dB)	Time (s)
1	35.11	7.07
2	36.55	8.68
3	36.64	10.40
4	36.49	12.50
5	36.56	13.16
6	36.67	14.31
7	36.45	16.97
8	36.54	17.55
9	36.44	18.15
10	36.63	19.42

Fig. 12. (a)–(d) Data sets used in the clustering performance evaluation. (e)–(f) Failures of clustering (b)–(c) by CURE. (g)–(i) Failures of clustering (b)–(d) by spectral clustering.

which has proven to be efficient in some challenging clustering problem [24], unfortunately failed in three of the four test data sets, as seen in Fig. 12(g)–(i).

Fig. 13 shows the clusters found by our proposed SOTT based hybrid clustering algorithm for each of the above data sets. For data sets (a), (c), and (d), our SOTT hybrid clustering algorithm produces the natural clustering results as perceived by the human eyes. For data set (b), only the vertically elongated textual patterns next to the horizontally aligned objects are imperfectly clustered into two different clusters instead of one.

The overall computation time of our proposed SOTT hybrid clustering algorithm can be divided into two parts: one on the learning and partition clustering time of the SOTT and the other is the time it used for AHC. In the first stage of the SOTT partition clustering, it can be shown that for n patterns and k atomic

Fig. 13. Clustering results of different data sets given by our proposed algorithm.

Fig. 14. Computation time versus size of the database.

clusters, the computational complexity of SOTT learning stage is mainly dependant on the SOTT structure, learning parameters rather than on the size of the data set, while the computational complexity of mapping all the patterns in the data set to the atomic clusters is $O(n \log k)$ due to the tree-structured search. In the second stage of AHC, the computational complexity of $\text{Link}(C_i, C_j)$ for all clusters dictates the computational complexity of the stage and it is $O(k^3)$. Usually n is much larger than k , thus the computational complexity in the second stage is negligible comparing with that in the first stage. This is in congruence with the observation of our simulation. Therefore, it can be concluded that the proposed SOTT hybrid clustering algorithm is scalable to the data set size. The scalability of our proposed algorithm is manifested in Fig. 14. In Fig. 14, it can be seen that the proposed SOTT hybrid clustering algorithm is computationally much more efficient than the recently proposed CURE algorithm [13], especially for large data sets.

Fig. 15. Clustering results for different number of initial clusters. (a) $k = 64$. (b) $k = 16$.

Though the use of hybrid clustering algorithm can completely unbridle the second and third limitations mentioned in the introduction, as observed from the clustering results of Fig. 15, the number of atomic clusters will still influence the performance of the hybrid clustering algorithm despite its effect has been reduced. However, the number of atomic clusters and the clustering results of the proposed SOTT hybrid clustering algorithm are not as vulnerable to the setting of the learning parameters of the SOTT as to the growing variants of SOM. Usually it is difficult to predict the number of atomic neurons in advance. However, the different resolutions created by the SOTT structure provide a flexibility to alleviate the problem associated with a fixed or an inappropriate choice of the number of atomic clusters. With the ability to maintain a cohesive parent-child and sibling relationships in the SOTT during the partition clustering stage, the AHC can be applied on different layers which have different number of neurons (i.e., partitions). A clustering validation [14] process can then be added to decide which level of clustering and merging is preferred for the given data.

VI. CONCLUSION

A novel online hierarchical SOTT is proposed in this paper in which a multipath search is used to overcome the bias of the tree-structured search. A parameter named leakage ratio is introduced to control the performance tradeoff between different neuronal layers. During the learning phase, specific updating rule is designed to keep both the interlayer and intralayer relationship well during the learning phase.

The hierarchical structure of the proposed SOTT is also exploited in a hybrid clustering algorithm to overcome the clustering performance deficiencies of classic k -means and SOM algorithms. To fully exploit the advantages of online learning, and reduced computational complexity of the SOTT architecture, a new linkage metric is introduced for the AHC. This linkage metric can be updated online during the partition clustering stage and does not need to wait for the patterns to be labeled. The proposed SOTT hybrid clustering algorithm has demonstrated to be computational efficient and possesses good scalability. Besides, its clustering performance is more robust against learning parameter variation compared with the growing variants of SOM.

The experimental results show that, the performance of the proposed SOTT is comparable to the basic full-search SOM. At the same time, the computation efficiency of SOTT is much better than that of basic SOM and other vector quantizers. In addition, the progressive encoding property brought by SOTT is attractive for developing new efficient and elegant solutions to data clustering and pattern classification applications.

REFERENCES

- [1] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, "Dynamic self-organizing maps with controlled growth for knowledge discovery," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 601–614, May 2000.
- [2] P. Berhkin, "Survey of clustering data mining techniques," Accrue Software, Tech. Rep., 2002.
- [3] J. Blackmore and R. Miiikkulainen, "Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, 1993, pp. 450–455.
- [4] A. Buzo, A. H. Gray Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-92, no. 10, pp. 562–574, Oct. 1980.
- [5] R. F. Chang, W. T. Chen, and J. S. Wang, "Image sequence coding using adaptive TSVQ with multipath searching," *Proc. IEEE*, vol. 139, no. 1, pp. 9–14, Jan. 1992.
- [6] T. D. Chiueh, T. T. Tang, and L. G. Chen, "Vector quantization using tree-structured self-organizing feature maps," *IEEE J. Sel. Areas Commun.*, vol. 12, no. 9, pp. 1594–1599, Dec. 1994.
- [7] M. Edwards and D. R. Morse, "The potential for computer-aided identification in biodiversity research," *Trends Ecol. Evol.*, vol. 10, pp. 153–158, 1995.
- [8] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: Ordering, convergence properties and energy functions," *Biolog. Cybern.*, vol. 67, no. 1, pp. 47–55, 1992.
- [9] B. Fritzke, "Growing Grid—A self-organizing network with constant neighborhood range and adaption strength," *Neural Process. Lett.*, vol. 2, no. 5, pp. 1–5, 1995.
- [10] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.
- [11] J. Giske, G. Huse, and O. Fiksen, "Modeling spatial dynamics of fish," *Rev. Fish. Biol. Fish.*, vol. 8, pp. 57–91, 1998.
- [12] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.
- [13] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1998, pp. 73–84.
- [14] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," in *Intelligent Information Systems Journal*. Norwell, MA: Kluwer, 2001.
- [15] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," in *ACM Comput. Survey*, vol. 31, 1999, pp. 264–323.
- [16] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: A hierarchical clustering algorithm using dynamic modeling," *IEEE Computer: Special Issue on Data Analysis and Mining*, vol. 32, no. 8, pp. 68–75, 1999.
- [17] S. Kaski and T. Kohonen, "Exploratory data analysis by the self-organizing map: Structure of welfare and poverty in the world," in *Proc. 3rd Int. Conf. Neural Networks in the Capital Markets—Neuronal Networks in Financial Engineering*, A. P. N. Refenes, Y. Abu-Mostafa, J. Moody, and A. Weigend, Eds. Singapore, 1996, pp. 498–507.
- [18] T. Kohonen, *Self-Organizing Maps*. New York: Springer-Verlag, 2001.
- [19] —, "Self-organized formation of topologically correct feature maps," *Biolog. Cybern.*, vol. 43, pp. 59–69, 1982.
- [20] P. Koikkalainen and E. Oja, "Self-organizing hierarchical feature maps," in *Proc. Int. Joint Conf. on Neural Networks (IJCNN'90)*, vol. II, San Diego, CA, 1990, pp. 279–284.
- [21] P. Koikkalainen, "Progress with the tree-structured self-organizing map," in *Proc. 11th Eur. Conf. Artificial Intelligence (ECAI'94)*, 1994, pp. 211–215.
- [22] C.-R. Lin and M.-S. Chen, "A robust and efficient clustering algorithm based on cohesion self-merging," in *Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2002, pp. 582–587.
- [23] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, 1980.
- [24] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2001.
- [25] S. O. Pei and Y. S. Lo, "Color image compression and limited display using self-organization Kohonen Map," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 2, pp. 191–205, Apr. 1998.
- [26] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.
- [27] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 11, pp. 586–600, 2000.
- [28] H. Yin and N. M. Allinson, "Self-organizing mixture networks for probability density estimation," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 405–411, Mar. 2001.

Pengfei Xu received the B.Eng. degree in electrical engineering from Xi'an Jiaotong University, Shaanxi, China, in 2002. He is currently pursuing the Ph.D. degree in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

His research interests include data clustering and image segmentation.

Chip-Hong Chang (S'92–M'98–SM'03) received the B.Eng. degree (hons.) in electrical engineering from National University of Singapore, Singapore, in 1989, and the M.Eng. and Ph.D. degrees from the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

His industrial experience includes being a Component Engineer of General Motors, Singapore, and Technical Consultant of Flextech Electronics Pte. Ltd. He joined the Electronics Design Centre,

Nanyang Polytechnic, as a Lecturer in 1993. Since 1999, he has been with the School of Electrical and Electronic Engineering, NTU, where he is currently an Assistant Professor. He has served a number of administrative roles during his academic career. He holds concurrent appointments at NTU as the Deputy Director of the InterSchool Research Centre, Centre for High Performance Embedded Systems, and the Program Director of VLSI Design and Embedded Systems Research Group of the Centre for Integrated Circuits and Systems. His current research interests include low-power arithmetic circuits, design automation and synthesis of digital filters, self-organizing maps and algorithms and architectures for digital image processing. He has published more than 100 refereed international journal and conference papers and book chapters.

Andrew P. Paplinski (M'90) received the M.Eng. and Ph.D. degrees from the Faculty of Electronics, Warsaw University of Technology, Warsaw, Poland, in 1967 and 1980, respectively.

From 1967 to 1981, he occupied a variety of teaching, research, and managerial positions in the Department of Computer Science, Faculty of Electronics, Warsaw University of Technology. During this time, he supervised the design of minicomputer systems KRTM-20 and GEO-20. He was a recipient of numerous awards from the Ministry of Science

and Technology. In 1982, he visited the Department of Computer Science, Australian National University, Canberra, Australia, and from 1983 to 1986, was a Research Fellow and Lecturer in the Department of Electrical and Electronic Engineering, University of Adelaide, Adelaide, Australia, where he worked on multivariable control systems. Since 1987, he has been Lecturer, Senior Lecturer, and Associate Professor in the School of Computer Science and Software Engineering, Monash University, Melbourne, Australia. His recent research activities concentrate around computational intelligence and span areas of computer vision, ultrasonic imaging, neural networks, and mathematical modeling. He also investigates aspects of hardware implementation of the related algorithms. He published over eighty papers, book chapters, technical reports, and five textbooks. Since 1994, he has collaborated with the Image Processing Group from King's College London on the interpretation of the Posterior Capsule Opacification images. Currently, he collaborates with the Department of Computer Systems and Electrical Engineering, Lulea University of Technology, Sweden, on modeling autism using artificial neural networks. Currently, he supervises five Ph.D. students.