Paper

# Self-Organizing Tree Using Cluster Validity

Yasue Sasaki* Non-member
Yukinori Suzuki* Member
Takayuki Miyamoto* Non-member
Junji Maeda* Non-member

Self-organizing tree (S-TREE) models solve clustering problems by imposing tree-structured constraints on the solution. It has a self-organizing capacity and has better performance than previous tree-structured algorithms. S-TREE carries out pruning to reduce the effect of bad leaf nodes when the tree reaches a predetermined maximum size ($U$), However, it is difficult to determine $U$ beforehand because it is problem-dependent. $U$ gives the limit of tree growth and can also prevent self-organization of the tree. It may produce an unnatural clustering. In this paper, we propose an algorithm for pruning algorithm that does not require $U$. This algorithm prunes extra nodes based on a significant level of cluster validity and allows the S-TREE to grow by a self-organization. The performance of the new algorithm was examined by experiments on vector quantization. The results of experiments show that natural leaf nodes are formed by this algorithm without setting the limit for the growth of the S-TREE.

**Keywords:** self-organizing tree, pruning algorithm, cluster validity, significant level, vector quantization

## 1. Introduction

Clustering techniques have been widely used in engineering problems such as pattern recognition, image processing, computer vision and recently in data mining. Clustering techniques enable detection of statistical regularities in a random sequence of input vectors and/or patterns and division of a collection of objects into a number of subgroups. The objects in each subgroup show a certain degree of closeness or similarity [1]~[3]. To determine optimal clustering, all possible partitions must be computed. However, exhaustive enumeration of all possible partitions is required to obtain an optimal clustering. This task is clearly computationally impossible.

To avoid this exhaustive enumeration, iterative methods have been employed. The criterion function by which objects are moved from one cluster and to another is optimized by an iterative method so as to improve the value of the criterion function. There are principally two types of iterative methods. One method is used in situations in which the data distribution is known. A Bayesian or maximum likelihood approach can be used to solve the clustering problem by estimating the values of parameters of a distribution. The other method is used in a situation in which the data distribution is not known. In this situation, clustering can be treated as an optimization problem by specifying a suitable cost function to be minimized [4] [5].

The computational burden required to solve an optimization problem for clustering is large. In order to reduce the computational burden, structural constraints such as lattices and trees have been proposed. Tree-structured clustering methods have become popular for vector quantization (VQ) literatures. Our attention has been focused on binary trees in tree-structured clustering for VQ, which is a special case of clustering. Tree-structured clustering is fast, scale well (in processing time) with the number of feature dimensions and clusters, and can capture hierarchical structures in the data [6] [7].

Campos and Carpenter proposed self-organizing tree (S-TREE), a family of methods that enables hierarchical representation of data [8]. S-TREE models solve clustering problems by imposing tree-structured constraints on the solution. Tree-structured clustering algorithms adapt their weights vectors by online incremental learning. S-TREE has a self-organizing capability and has better performance than previous tree-structured algorithms. One essential algorithm of an S-TREE is pruning, because an S-TREE grows in a greedy fashion. Pruning is a mechanism for reducing the effect of bad leaf nodes. Unnecessary nodes are pruned when the tree reaches a predetermined maximum size ($U$). However, it is difficult to determine $U$ beforehand because it is problem-dependent. $U$ gives the limit of tree growth and can also prevent self-organization of the tree. Therefore, to realize natural growth of the tree according to its self-organizing mechanism, a new pruning algorithm that does not require maximum size $U$ is needed. In this paper, we propose an algorithm for pruning of an S-TREE using cluster validity. This algorithm prunes extra nodes based on a significant level of cluster valid-

* Muroran Institute of Technology
27-1, Mizumoto-cho, Muroran 050-8585, Japan

ity and allows the S-TREE to grow by a self-organizing mechanism. The performance of the new algorithm was examined by experiments on VQ.

The remainder of the paper is as follows. An outline of the S-TREE and a description of the new algorithm for pruning are given in section 2. In section 3, examples of VQ using the proposed algorithm are given to show the performance of the algorithm. Finally, conclusions are given in section 4.

## 2. Improved S-TREE Algorithm

An S-TREE is a binary tree structure as shown in Fig. 1. There are two kinds of nodes in a binary tree: leaf nodes and inner nodes. A leaf node has no child nodes, and the remaining nodes are inner nodes. Each node $j$ has three attributes: node weight vector $\mathbf{w}_j$, accumulated cost $e_j$ for splitting a node, and counter $N_j$ (the number of times the node has been updated). The main steps in the S-TREE algorithm are shown in Fig. 2. In the first step of the S-TREE algorithm, the tree is initialized with a single root node. When input vector $\mathbf{A}$ is inputted to the tree, the algorithm finds a leaf node $\mathbf{j}$ and determines whether it should modify the tree structure by splitting that node or not. If a cost function $e_{\mathbf{j}}$ is greater than threshold $E$, the algorithm splits node $\mathbf{j}$. $e_{\mathbf{j}}$ is computed from the square distance from $\mathbf{A}$ to the winning vector $\mathbf{w}_{\mathbf{j}}$.

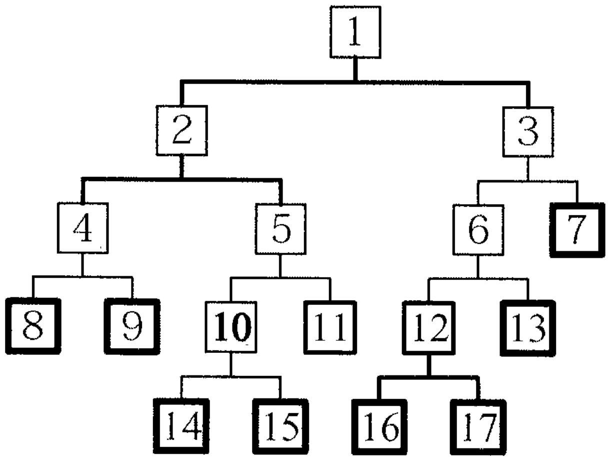


Fig. 1. An example of a binary tree. The nodes drawn by thick lines indicate leaf nodes and the other nodes are inner nodes.

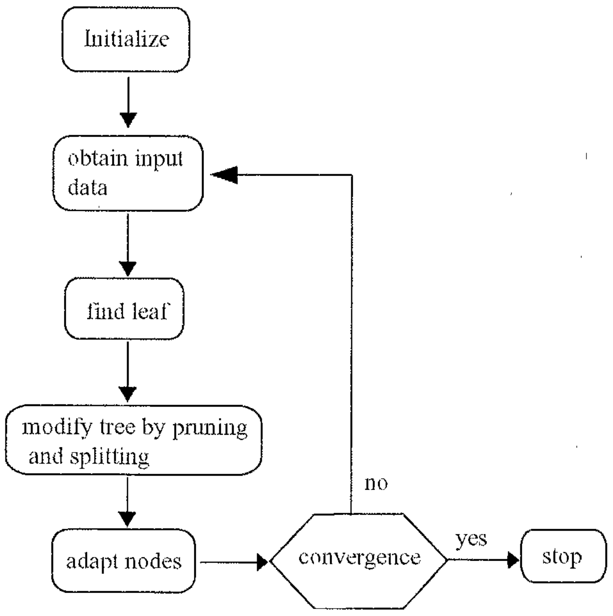


Fig. 2. Block diagram of the S-TREE algorithm.

After modification of tree structure by splitting, the accumulated cost $e_j$, the counter $N_j$, and weight vector $\mathbf{w}_j$ are updated for each node $j$ in the path connecting the root node (j = 1) to the winning leaf ($j = \mathbf{j}$) in response to the input vector $\mathbf{A}$. Update is carried out as follows.

$$\Delta \mathbf{e}_j = \epsilon, \cdots\cdots (1)$$

$$\Delta N_j = 1, \cdots\cdots (2)$$

$$\Delta \mathbf{w}_j = (\mathbf{A} - \mathbf{w}_j)/N_j, \cdots\cdots (3)$$

$$\Delta E = \beta_1 (e_{\mathbf{j}} - E), \cdots\cdots (4)$$

where $\beta_1$ is a constant given in advance. $\epsilon$ is the value of the cost measure for the current input vector and is computed as

$$\epsilon = \epsilon_0 / \bar{\epsilon}_0, \cdots\cdots (5)$$

where $\epsilon_0$ is $\| \mathbf{A} - \mathbf{w}_{\mathbf{j}} \|^2$ and $\bar{\epsilon}_0$ is a fast-moving average of $\epsilon_0$ computed using

$$\Delta \bar{\epsilon} = \beta_2 (\epsilon_0 - \bar{\epsilon}_0). \cdots\cdots (6)$$

$E$ is also increased to $\gamma E$, where $\gamma$ is given in advance. After splitting, two children nodes are initialized as follows: The left child weight vector is set to $\mathbf{w}_{\mathbf{j}}$ and the right child weight vector is set to $(1+\delta)\mathbf{w}_{\mathbf{j}}$, where $\delta$ is a small positive constant. The counter $N_j$ for each child is set to 1. The cost variable $e_j$ for each child is set to $e_{\mathbf{j}}/2$. Since the S-TREE grows in a greedy fashion, it has a pruning mechanism to reduce the effect of bad splits. Pruning is performed when the number of nodes exceeds a predetermined $U$.

To set convergence criteria, the S-TREE uses a window of a fixed size and computes the performance of the algorithm over consecutive windows of the training data. To compensate for fluctuations that can occur for small windows, a moving average of the performance on consecutive windows is used to check for convergence. Taking $\bar{C}_\tau$ to be the smoothed moving average performance on window $\tau$, S-TREE's convergence criterion is defined by

$$\frac{| \bar{C}_{\tau-1} \bar{C}_{\tau} |}{\bar{C}_{\tau-1}} \le \eta, \cdots\cdots (7)$$

where $\bar{C}_\tau = \bar{C}_{\tau-1} + \beta_3 (C - \bar{C}_{\tau-1})$ and $C$ is the performance on window $\tau$.

As mentioned above, the S-TREE algorithm performs pruning when the number of nodes exceeds a predetermined $U$. However, since $U$ is problem-dependent, it is difficult to determine an appropriate value of $U$ beforehand. Furthermore, $U$ gives the limit of tree growth

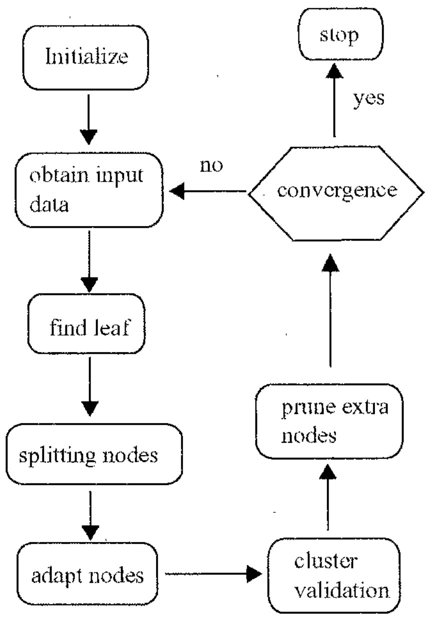


Fig. 3. Block diagram of the improved S-TREE algorithm.

and can prevent self-organization of the tree. We therefore propose a modified S-TREE algorithm that does not require $U$. The main procedures of the algorithm are shown in Fig. 3. In the modified algorithm, after the tree has been modified by splitting and adapting procedures for a set of vectors in one window, pruning is carried out using a significant level of cluster validation. Pruning is carried out for each inner node with two leaf children. This method was originally presented by Duda and Hart as a method to specify the number of clusters in the data [10]. We implement the pruning algorithm of the S-TREE according to the theory proposed by Duda and Hart. The idea is as follows. When clustering is performed using cluster validity $J(c)$, it monotonically decreases as the number of clusters $c$ increases. $J(c)$ decreases rapidly up to $c = \hat{c}$, which is the number of optimal clusters, as $c$ increases. Exceeding $\hat{c}$, $J(c)$ decreases gradually and finally becomes zero at $c = n$. Based on this idea, a null hypothesis that is frequently used in statistical reasoning is employed to determine $\hat{c}$.

Suppose that there is a set $\mathbf{A}$ of $n$ samples and that there is more than one cluster for samples $n$. The null hypothesis states that all $n$ samples are from a normal population with mean $\mu$ and covariance matrix $\sigma^2 I$, where $I$ is an identity matrix. If this hypothesis is true, any clusters found must have been formed by chance and any observed decrease in $J(c)$ has no significance. $J(1)$ is computed as

$$J(1) = \sum_{\mathbf{A}} \| \mathbf{A} - \mathbf{m} \|^2, \cdots\cdots\cdots\cdots (8)$$

where $\mathbf{m}$ is mean of the $n$ samples. $J(1)$ is the sum of the difference between the cluster center and samples. According to the null hypothesis, the distribution for $J(1)$ is approximately normal with mean $nd\sigma^2$ and variance $2nd\sigma^4$. $d$ is a dimension of $A_i \in \mathbf{A}$. Suppose that the set of samples is partitioned into two subsets, $\mathbf{A}_1$ and $\mathbf{A}_2$, so as to minimize $J(2)$:

$$J(2) = \sum_{i=1}^{2} \sum_{A \in \mathbf{A}_i} \| \mathbf{A} - \mathbf{m}_i \|^2, \cdots\cdots\cdots\cdots (9)$$

where $\mathbf{m}_i$ is the mean of the samples in $\mathbf{A}_i$. According to the null hypothesis, this partitioning is spurious.

If we know a sampling distribution of $J(2)$, we can determine the limit of $J(2)$ to abandon the null hypothesis for one cluster. For large $n$, it is approximately normal with mean $n(d - 2/\pi)\sigma^2$ and variance $2n(d - 8/\pi^2)\sigma^4$. The limit value for $J(2)$ can be obtained by assuming that the suboptimal partition is nearly optimal, by using the normal approximation for the sampling distribution, and by estimating $\sigma^2$ by

$$\hat{\sigma}^2 = \frac{1}{nd} \sum_{A \in \mathbf{A}} \| \mathbf{A} - \mathbf{m} \|^2 = \frac{1}{nd} J(1). \cdots\cdots\cdots\cdots (10)$$

The null hypothesis is rejected at the $p$-percent significance level if

$$\frac{J(2)}{J(1)} < 1 - \frac{2}{\pi d} - \alpha \sqrt{\frac{2(1 - 8/\pi^2 d)}{nd}}. \cdots\cdots\cdots\cdots (11)$$

$\alpha$ is determined by

$$p = 100 \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-1/2u^2} du = 100(1 - erf(\alpha)). \cdots\cdots\cdots\cdots (12)$$

In the present algorithm, after modification of the tree by splitting and adapting nodes has been completed for a set of vectors in one window, each inner node with two leaf nodes is tested to determine whether it satisfies (11) or not. If a node does not satisfy (11), then its children represent spurious clusters and can be pruned.

## 3. Experiments on Vector Quantization

VQ is a special case of clustering and is mainly used for data compression. Data compression is essential technology for communication via the Internet. It consists of two components: an encoder and a decoder. The encoder converts the original data into a compressed representation that has a smaller size in bits than the original data, the decoder uses the compressed data to reconstruct the original data.

VQ uses a code book for encoding and decoding data. It is carried out to create a small code book capable of encoding and decoding with the smallest possible between original and decoded data. The compression ratio is determined by

$$r = \frac{\text{size of original data in bits}}{\text{size of compressed data in bits}}. \cdots\cdots (13)$$

The higher the value of $r$ is, the better the compression algorithm is. The quality of the decoded data can be measured using the peak-signal-to-noise ratio (PSNR) in dB:

$$\mathrm{PSNR} = 10log_{10}\frac{\sigma^2}{\mathrm{MSE}}, \cdots\cdots\cdots\cdots\cdots\cdots\cdots (14)$$

where $\sigma^2$ is the variance of the original data and MSE is the reduction mean square error.

For image compression experiments, a training set was prepared by taking 4 x 4 blocks from four 256 x 256 gray-scale images (Bird, Bridge, Camera, Goldhill) showing in Fig. 4 (a)-(d). Two test sets were prepared in a similar fashion using the 256 x 256 gray-scale lena image (Fig. 4(e)) and peppers (Fig. 4(f)). Parameters in the S-TREE algorithm are shown in Table 1, where $T$ shows a window size. These parameters were set according to Campos and Carpenter [8]. Training was performed by changing of the $p$ significance level.

Fig. 5 shows typical decoded images of the test set of lena. Fig. 5(a), (b) and (c) are images with $p = 20$, $p = 40$, and $p = 60$, respectively. PSNR of the image shown in Fig. 5(a) and (b) is 26.0 dB and 26.0 dB, respectively. The PSNR of the image with $p = 60$ is 25.9 dB. The PSNRs of the other images are almost the same as that with $p = 20$. Fig. 6 also shows decoded images of the test set of peppers. Fig. 6(a), (b), and (c) are the cases of $p = 20$, $p = 40$, and $p = 60$, respectively. Furthermore, PSNR of the images shown in Fig. 6(a), (b), and (c) are 26.2 dB, 26.2 dB, and 25.9 dB, respectively.

The PSNR of the test set of lena by changing of the percent significant levels are shown in Fig. 7. For these computation, squared distance $D_w$ by input vectors in a window is computed as

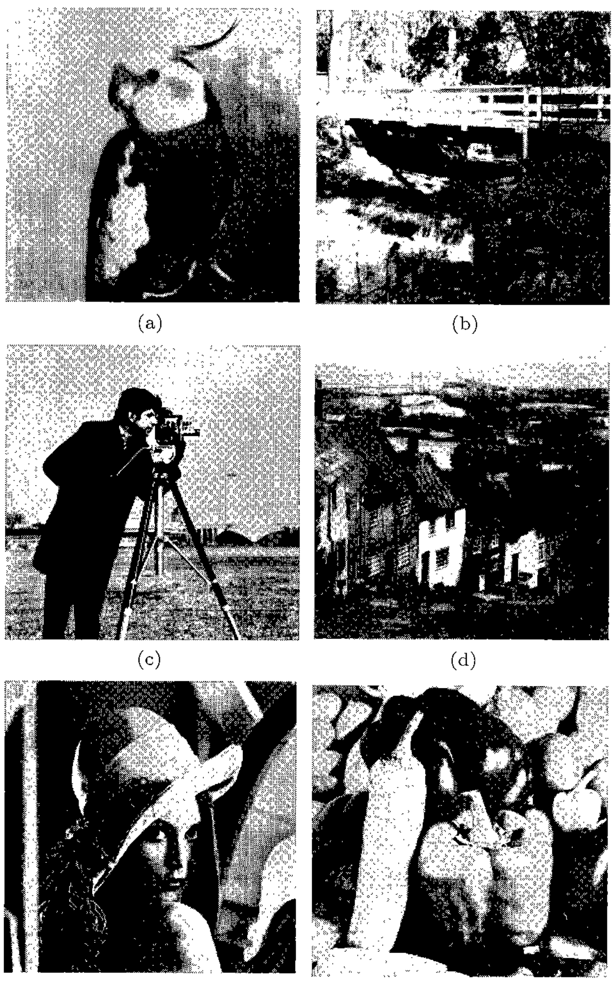
$$D_i = \sum_j \sum_{\mathbf{A}\in\Lambda_j} \| \mathbf{A} - \mathbf{w}_j \|^2, \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (15)$$

where $\Lambda_j$ is the set of input mapped into $\mathbf{w}_j$.

$$D_w = \sum_{i=1}^{T} D_i. \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (16)$$

When $D_w < 2.35 \times 10^6$ is satisfied, the algorithm was considered to be converged. In Fig. 7, PSNRs by the S-TREE algorithm are also shown. In this computation, parameters were employed as the same parameters shown in Table 1. Tree growth of the S-TREE was stopped when the number of leaf nodes was the same as that of corresponding significant level of the proposed algorithm. As shown in Fig. 7, the PSNRs of the S-TREE algorithm are approximately the same as the PSNRs of the proposed algorithm. Namely, if we give the number of leaf nodes that is determined by the proposed algorithm to the S-TREE algorithm beforehand, the S-TREE shows the same performance as that of the proposed algorithm. This means that the propose algorithm is able to form natural clusters without setting the limit for the growth of the S-TREE.

The curve showing PSNR change is flat except for the


(a) (b) (c) (d) (e) (f)

(a) Bird, (b) Bridge, (c) Camera, (d) Goldhill. (e) and (d) show test sets, lena and peppers.

Fig. 4. Four training images.

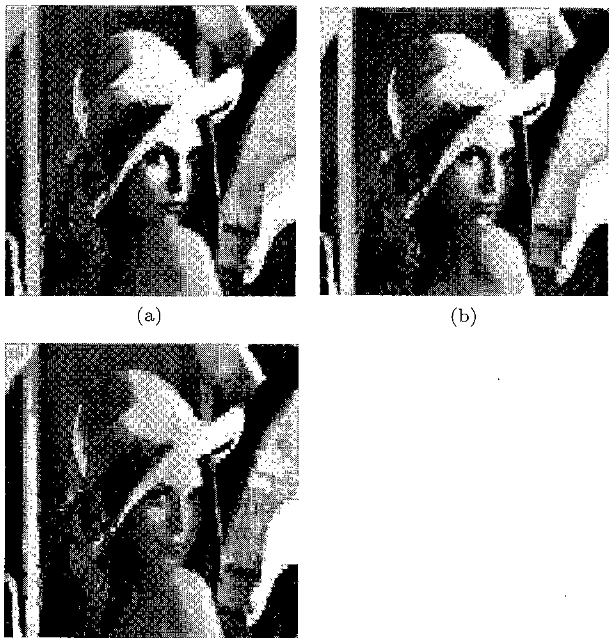

(a) (b) (c)

Fig. 5. Decoded images of lena by changing of the significant level, $p$.

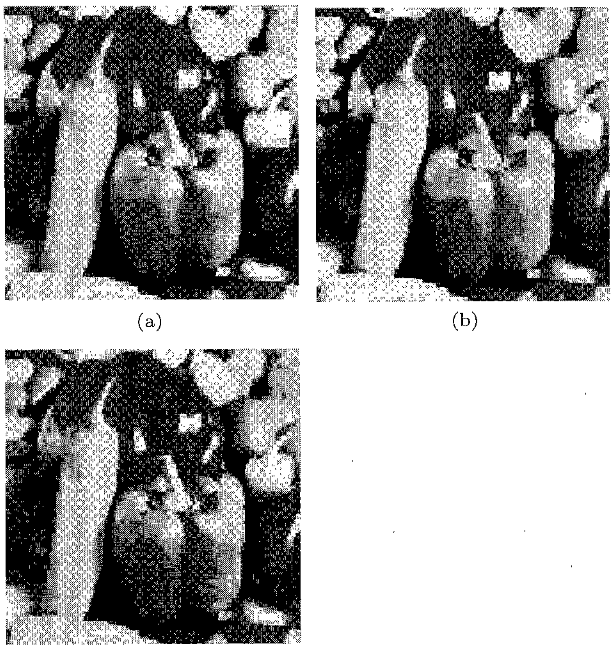


Fig. 6. Decoded images of peppers by changing of the significant level, $p$.

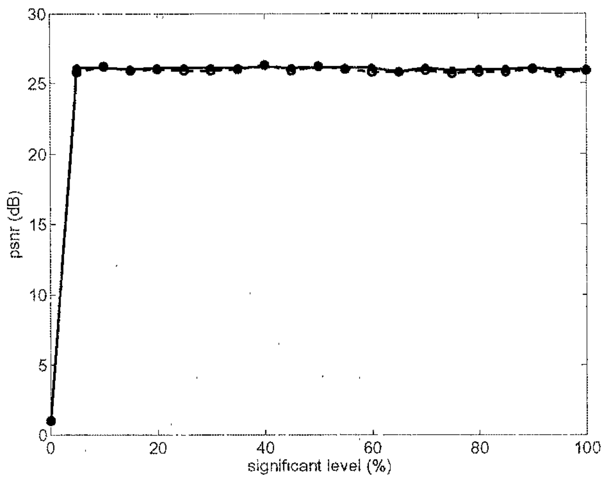


Fig. 7. The dotted line shows PSNR by changing of the significant level, $p$. The solid line also shows PSNRs by the previous S-TREE algorithm.

Table 1. Parameters used in experiments.

| parameter | value |
|---|---|
| $E_0$ | 50 |
| $\beta_1$ | 0.02 |
| $\beta_2$ | 0.075 |
| $\beta_3$ | 0.2 |
| $\gamma$ | 1.5 |
| $\Gamma$ | 0.4 |
| $\delta$ | 0.0001 |
| $\eta$ | 0.0115 |
| $T$ | 1024 |

part of at $p = 0.0$. This indicates that the PSNR is not sensitive to change in $p$. Fig. 7 shows that the change in the significant level does not affect the PSNR. This means that the proposed algorithm can obtain decoded image whose PSNR is independent of $p$. This is a desirable property of the proposed algorithm, because parameter $p$ given in advance does not affect the quality of the decoded image.

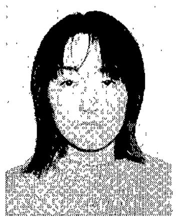
## 4. Conclusions

In this paper, we have proposed a pruning algorithm for an S-TREE using a significant level of cluster validity. Since the algorithm does not set the limit for the growth of the S-TREE in advance, it allows natural growth of the tree according to its self-organizing mechanism and it also prunes extra nodes. The results of experiments show that natural leaf nodes are formed by this algorithm without setting the limit for the growth of the S-TREE. In the proposed algorithm, since we do not need to set $U$ beforehand, it is useful for practical applications.

(Manuscript received March 31, 2003,
revised July 28, 2003)

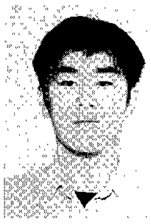
## References

(1) I. Gata and A.B. Geva: "Unsupervised optimal fuzzy clustering," *IEEE Trans. PAMI*, Vol. 11, No. 12, pp.773-781 (1989)
(2) J.C. Bezdek: Pattern recognition with fuzzy objective function algorithms, Plenum Press: New York (1987)
(3) Y. Suzuki, A. Kawamura, S. Saga, and J. Maeda: "Perceptual clustering with fuzzy encoding," *Trans. IEEJ(C)*, Vol. 123, No. 1, pp.144-149 (2003)
(4) P. Fränti and T. Kaukoranta: "Binary vector quantizer design using soft centroids, Signal processing: Image Communication," Vol. 14, pp.677-681 (1999)
(5) C. Amerijckx, M. Verleysen, P. Thissen, and J. Legat: "Image compression by self-organizing Kohonen map," *IEEE Trans. Neural Networks*, Vol. 9, No. 3, pp.503-507 (1998)
(6) B. Hammer and T. Villmann: "Generalized relevance learning vector quantization," *Neural Networks*, Vol. 15, pp.1059-1068 (2002)
(7) N. Vlajic and H.C. Card: "Vector quantization of images using modified adaptive resonance algorithms for hierarchical clustering," *IEEE Trans. Neural Networks*, Vol. 12, No. 5, pp.1147-1162 (2001)
(8) M.M. Campos and G. Carpenter: "S-TREE: self-organizing trees for data clustering and online vector quantization," *Neural Networks*, Vol. 14, pp.505-525 (2001)
(9) Y. Sasaki: A study on clustering using a self-organizing tree, Master thesis of Muroran Institute of Technology (2003)
(10) R.O. Duda and P.E. Hart: Pattern recognition and scene analysis, John Wiley & Sons: New York, pp.239-243 (1973)

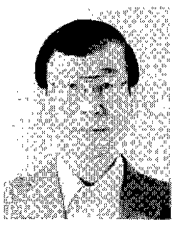


**Yasue Sasaki** (Non-member) received the B.S. degree in information engineering from Muroran Institute of Technology in 2001. She also received the M.S. degree in information engineering from Muroran Institute of Technology in 2003. Her research interests includes clustering theory and its application to image processing.

**Yukinori Suzuki** (Member) received the B.S. degree in electrical engineering from Muroran Institute of Technology in 1980. In 1985 he received Ph. D. degree in biomedical engineering from Hokkaido University. From 1985 to 1989 he was a research associate in the Department of Electronic Engineering in Muroran Institute of Technology. He was a visiting scholar at the Center for Adaptive Systems at Boston University. He is currently an associate professor at the Department of Computer Science and Systems Engineering in Muroran Institute of Technology. He was the program Chair of International Workshop on Soft Computing in Industry, which was held in 1999 in Muroran, Japan. He was a guest editor of Proceedings of the IEEE, 2001 special issue of Industrial Innovations Using Soft Computing.



**Takayuki Miyamoto** (Non-member) received the B.S. degree in information engineering from Muroran Institute of Technology in 2003. He currently working for his M.S. degree in Muroran Institute of Technology.



**Junji Maeda** (Non-member) received the B.S. degree in physics from Hokkaido University in 1972, and the M.E. and Ph.D. degrees in applied physics from Hokkaido University in 1974 and 1985, respectively. He was an Assistant Professor of the Department of Applied Physics at Hokkaido University. In 1988, he joined Muroran Institute of Technology, where he is currently a Professor of the Department of Computer Science and Systems Engineering from 1994. His research interests include image processing and computer vision.