International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 3 (2015) pp. 5561-5579

http://www.ripublication.com

An Effective Palette Reordering Scheme by Dynamic Self-Organizing Maps with Controlled Growth

P. Niraimathi[1], K. Bhoopathy Bagan[2]

[1,2] Department of Electronics Engineering, Madras Institute of Technology, Anna University, Chennai-600044, India
[1]mathi.pn@gmail.com,[2]bhoopathy02@yahoo.com

Abstract

Palette reordering is an effective approach for improving the compression of color indexed images. The best solution for palette re-ordering problem is obtained from Dynamic self-organizing maps with growing nodes suitably developed to overcome the problem of fixed structure with a motor map neural network. In this paper, we provide a novel algorithm by making use of Dynamic self-organizing maps that enables dynamic distribution of the spatial indexes in the image by varying the spread factors to obtain greater compression ratios than state-of-art techniques recently available in the literature. Experimental results assert the real effectiveness of the proposed method both in terms of compression ratio and zero-order entropy with improved computational speed.

Index Terms—Indexed images, Palette-Indexed Image, Self Organizing Maps, Growing Self Organizing Map, Spread Factor, Learning Rate.

Introduction

Color-indexed images are represented by a matrix of indexes called index image and by a color map or palette. The indexes points to positions in the color-map and, therefore, establishes the colors of the corresponding pixels. The mapping between index values and colors (typically, RGB triplets) is not unique it can be arbitrarily permuted, under the condition that the corresponding index image is changed accordingly. Employing digital displays with full colour involves high cost due to the necessity of high speed memory. Alternate solution found today most of the monitors is to provide 8 bits to specify the color of each pixel, which means that only 256 colors can be displayed while millions of colors may be displayable [3] - [5]. The palletized images are used in many applications such as game cartridges,

geographic information systems, shared whiteboards, application sharing, online training, simple animations and available over network connections such as online, CompuServe, prodigy, and the World Wide Web.

Palette reordering methods aims in finding suitable permutation that makes the palette image more amenable with lossless compressors. It is classified into two main methods. The first class is referred as color based methods which are characterized by relying only on the information provided by the color palette. The second class characterizes techniques relying only on the statistical information conveyed by the index image, independently of its meaning in terms of color representation. We refer to the techniques in this class as 'index-based methods'. The main idea behind index-based methods for palette reordering is that colors that occur frequently close to each other should have close indexes.

The bottle neck of these methods lies in obtaining the optimal configuration which results in high computational complexity. Indeed, the number of possible configurations for a table of M colors corresponds to the number of permutations of M objects, which equals to M! [6]. Most of the interesting cases exhaustive search is impractical, and then it is needed an approach to allow us to reach with low complexity for near-optimum solution.

The palette re-indexing method proposed by Zeng et al. [23] is based on one-step look ahead greedy approach. The algorithm starts by finding the index that is most frequently located adjacent to other (different) indexes, and the index that is most frequently found adjacent to it. Battiato et al. [6] proposed a greedy strategy based on sequentially selecting the best edge still not processed, i.e., the one with the largest weight. The methodology tends to smooth the relative transitions in the indexed image, solving in an approximate way a related optimization problem over a weighted graph. Chang et al. proved that it is possible to achieve high compression with acceptable image quality using the topology-preserving property of self organizing Kohonen feature map [10] which considers "1D string neural structure" wherein, the neuron closest to each fed training vector, called the "winning neuron", will update itself while the neighbouring neurons will update according to the neighbouring function and gain function. Rundo et al. suggested a Motor Map Neural Network based re-indexing that uses an unsupervised, application independent, highly adaptive learning algorithm called "Winner-take-all" learning driven by the reward function [12].

A dynamic feature map model called the growing self-organizing map (GSOM) is proposed in this paper to overcome the problem of self-organizing maps with adaptive learning rate and improved data distribution [11].. A detailed description of the GSOM algorithm is presented and its characteristics are elaborated. The need for a measure for controlling the growth of the GSOM (or any such algorithm) is highlighted, and an indicator called the spread factor is presented as a method of achieving such a control. The entropy of the re indexed images is varied by growing the map which is controlled by a spread factor. Certainly, the ordering of palettes changes which impacts effective re indexing.

This paper is structured as follows. Section II introduces the problem of re-indexing, Section III described the Self Organizing Map. Dynamic Self Organizing

Map is described in Section IV. Section V describes Effective re-indexing algorithm with GSOM. Experimental results are shown in Section VI. Conclusions are drawn in Section VII.

Problem Statement

The re-indexing problem can be formulated as follows. Let I be an image of $m \times n$ pixels with distinct colours in which $I(x, y)$ denotes the color at pixel location (x, y) of I. Consider V to be the colour palette consisting of N colours represented by $V = \{v_1, v_2, ..., v_N\}$. Let I' be an $m \times n$ matrix of indexes and $I'(x, y)$ point to colours in the colour palette V. The relationship between the image I and matrix I' is expressed by the following equation:

$$I(x, y) = V(I'(x, y)) \tag{1}$$

This represents an indexed image. The purpose of palette reordering is to find a new palette $P = \{p_1, p_2, ..., p_N\}$ such that the corresponding $m \times n$ matrix may be compressed efficiently by a lossless compression algorithm where in P can be obtained from V through permutation [9]. The residual entropy of local differences is considered to estimate the overall "energy" of the image. The information needed to reconstruct the original image is

1) the colour of pixel p_1
2) a table providing the correspondence between colours $S_1, S_2, ..., S_N$ with index $i_1, i_2, ..., i_N$
3) the sequence of differences is given by

$$D(I') = \{d_{x,y} \mid x = 1,2,..., m; y = 1,2,..., n\} \tag{2}$$

where $d_{x,y}$ is a local difference obtained by considering some specific patterns as better specified by information theory which tells us that any lossless scheme to encode the set of differences $D(I')$ requires bits per pixel (bpp) greater than or equal to the zero order entropy of the statistical distribution of $D(I')$. Hence, finding an optimal indexing scheme is a crucial step for differential lossless compression of indexed images.

Overview of Self Organizing Map

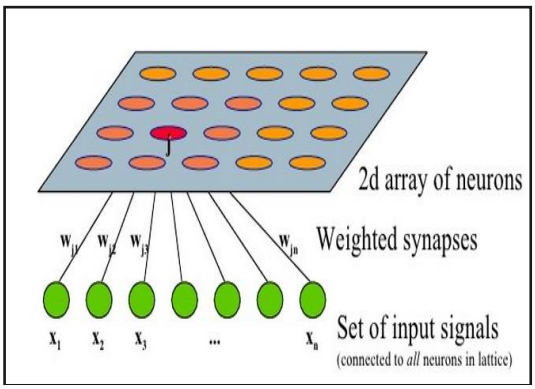
The attractiveness of neural networks come from the remarkable information processing characteristics of a biological system such as non-linearity, high parallelism, robustness, fault and failure tolerance, learning, ability to handle imprecise and fuzzy information, and their capability to generalize [25]. Self-organizing maps (SOMs) are well-known clustering methods proposed by Kohonen [24], and have proven to be extremely useful for input data with high dimensionality

and complexity [26]. The SOM architecture was originally motivated by the topological maps and the self-organization of sensory pathways in the brain [27]. The standard SOM algorithm is a computationally powerful abstraction of the actual biological mechanism. This algorithm realizes a self-organizing process following two simple rules [28]:

(a) start a competition among nodes finding the best matching unit (the winner)

(b) adapt the synaptic weights of the winner and topological neighborhood of it.

The main focus of the SOM is to summarize information while preserving topological relationships [29]. The objective of SOM is to represent high-dimensional input patterns with weight vectors that can be visualized in a usually two-dimensional (2D) lattice structure [31]. SOM input patterns are transformed into a one or usually 2D lattice structure, consisting of nodes associated with different clusters [30]. Each unit in the lattice is called a neuron, and adjacent neurons are connected to each other, which gives the clear topology of how the network fits itself to the input space. Input patterns are fully connected to all neurons via adaptable weights, and during the training process, neighboring input patterns are projected into the lattice, corresponding to adjacent neurons. A neuron is iteratively updated during training based on the learning vectors so a well-trained SOM represents a distribution of the input data over a 2D surface preserving topology. In this context, a cluster can be defined as a group of neurons with short distances between them and long distances to the other neurons [31].



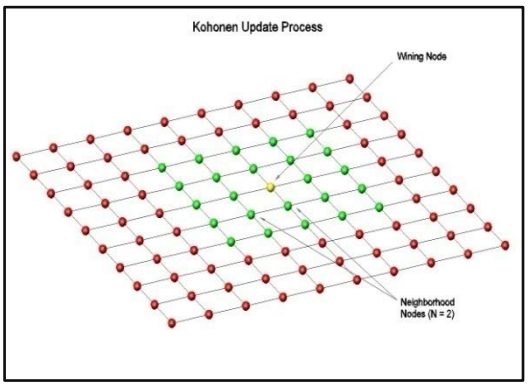


Fig. 1 Kohonen network architecture **Fig. 2** Winner Neuron on the rectangular grid

The representation of the input space of the training samples called "Map" preserves the topological properties of the input space. SOM operates in two modes: training and mapping. The training process is a competitive process which builds the map using input training set, called Winner Take All (WTA) algorithm [14]. Only one node is activated at any one time. This activated neuron is called a winning neuron. Fig 1 represents the neuron in the output layer are arranged on a map. At each learning step, the winner neuron is identified as the neuron that best matches the input pattern shown in Fig 2. A neighbourhood of the winner neuron is considered and an update involving both the input and output weights for neurons belonging to this neighbourhood is performed. The weight is updated if and only if the corresponding

palette re-indexing leads to an improvement in terms of zero-order entropy of local differences in the image processed otherwise, the neuron weights are not updated [14].

However, due to the static structure of SOM, the neighbourhood preservation cannot always lead to perfect topology preservation as the algorithm has to fix the size of the grid at the beginning of the training process. This includes the topology as well as the number of nodes. An alternative to overcome this drawback, several dynamic variants of SOM are developed. Therefore, dynamic self-organizing algorithm successfully solves the problem of knowing the suitable size of the map in advance.

Self-Generating Feature Maps

The advantages of self-generating neural networks have caused some interest in the recent past. Several researchers have described such new neural architectures, both in the supervised and unsupervised paradigms. Although these new models obviously become more complex compared with their fixed structure counterparts, significant advantages could be gained from such dynamic models, the main advantage being their ability to grow (or change) structure (number of nodes and connections) to represent the application better. The common principal among these variants is that starts with a minimum number of nodes, then the structure grow as the learning process. However, they are capable to find the best size of the output space.

4.1.1. Incremental SOM

The incremental grid growing (IGG) [32], growing cell structures (GCSs) [33], and growing neural gas (GNG) [33] are some of the best known incremental SOM. Additionally, several improved SOM and SOM-related algorithms have been proposed in recent years [34].

4.1.2. Hierarchical SOM

The hierarchical SOM (HSOM) is another variation of SOM [35]. The basic idea of HSOM is to use multiple SOMs from the low resolution level to the high resolution level, but the number of neurons in each layer is predefined [36].

4.1.3Growing Cell Structures (GCS)

The GCS algorithm [37], [38] is based on the SOM, but the basic two-dimensional grid of the SOM has been replaced by a network of nodes whose connectivity defines a system of triangles. This triangle is distributed as well as possible over the area of nonzero probability in the space. Heuristics are used to both add and remove network nodes and connections.

The GSOM algorithm presented in the next section has some characteristics similar to the IGG but uses a weight initialization method, which is simpler and reduces the possibility of twisted maps. The GSOM also carries out node generation interleaved with the self-organization, thus letting the new nodes smoothly join the existing network

4.2 Growing Self Organizing Map Approach

The growing self-organizing map (GSOM)[39], an extension of Kohonen's self-organizing map algorithm, adapts not only the position of the map weight vectors in the input space, but also the topology of the map output space grid. This additional feature allows for an unsupervised generation of dimension-reducing projections with optimal neighbourhood preservation, even if the dimensionality of the input data set is not known. It has the same topology structure and weight adaptation rules as the SOM and it can grow under the control of growing parameters, which the Self-Organising Map (SOM) cannot. The starts with a minimal number of nodes (usually 4) and grows new nodes on the boundary based on a heuristic. The major advantages of GSOM over SOM are summarized as follows:

i. The shape of GSOM represents the hidden data structure better than SOM that leads to better identifiable clusters.
ii. New nodes are added to the necessary regions while keeping the order of nodes.
iii. Fewer nodes at the beginning of the training leads to the speed improvement.

In GSOM, input vectors are organized into categories depending on their similarity to each other. For Euclidean distance each input vector presented, to all the output nodes are computed. The weights of the node with the bare minimum distance, all along with the neighboring nodes are adjusted. This ensures that is the output of these nodes is slightly enhanced. This process is repeated until some condition for extinction is reached. After a sufficient number of input vectors have been presented, every output node becomes sensitive to a group of similar input vectors, and may therefore be used to represent character of the input data. Thus the Growing SOMs have a dynamic topology of neurons to help solve the dynamic nature of data on the image. GSOM can be performed in three phases, initialization, growing and smoothing.

4.2.1. The Need for Controlling the Growth in GSOM

It is advantageous if there is a mechanism for initially observing the most significant nodes and then, spread the idea to overall data set and to further spread out the mapping and obtains finer clusters. This will also facilitate in making decisions on regions of the data that are not of interest and direct the finer clustering only to regions of interest. When a set of nodes has been identified, look at the selected attributes (dimensions) of the data helps to gain further insight into the structure of the data.

It is also necessary to have a measure for controlling the spread of the map such that finer clustering of the data set can be achieved hierarchically as required. To achieve this control on the spread of the map, a method is developed to specify the amount of spread required by specifying a spread factor. It is anticipated that a low spread factor will be given at the beginning of analysis and then gradually increasing the spread factor value for further observations of selected regions in the data. The spread factor takes values between zero and one and is independent of the number of dimensions in the data. It is possible to compare the results of different data sets with

a different number of attributes by mapping them with the same spread factor. As the spread factor takes values from zero to one, where zero is the least spread and one is the maximum spread, the user will be able to identify the amount of spread required.

4.3 Phases in the GSOM

This section provides detailed description of each of the phases in the GSOM algorithm. Although based on the SOM, the GSOM includes a number of significant variations which are required to implement its adaptable structure [40].

4.3.1 Initialization Phase

The starting four nodes are initialized with random values from the input vector space with no restrictions are enforced on the directions of the lattice growth. Thus the initial square shape lets the map grow in any direction solely depending on the input values. A numeric variable is initialized to "0" at initialization and will keep track of the highest accumulated error value in the network. A value called the spread factor (SF) has to be specified which allows the user to control the growth of the GSOM and is independent of the dimensionality of the data set used. SF is used by the system to calculate the growth threshold (GT). This will act as a threshold value for initiating node generation. A high GT value will result in less spread out map, and a low GT will produce a well-spread map. Figure 3 shows the illustration of initializing the nodes where the initialization phase starts with a minimal number of nodes, usually four and grows new nodes on the boundary or interior based on a heuristic.

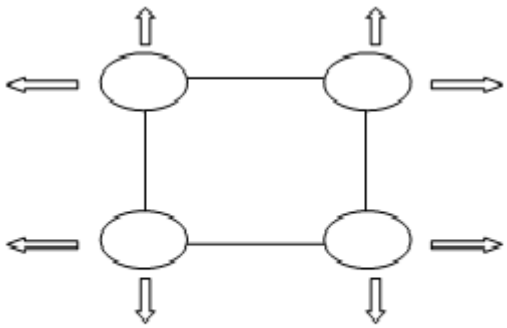


Fig 3. Initialization of nodes

All the starting nodes of the GSOM are boundary nodes, i.e. each node has the freedom to grow in its own direction at the beginning. Once a node is selected for growing, all its free neighboring positions will be grown new nodes from the boundary nodes.

4.3.2 Growing Phase

In the growing phase, when input data are presented to the network, a winner is found as per the algorithm. The difference between the input vector and the corresponding weight vector of the winner is accumulated as error value for that neuron. If neuron

contributes significantly to the total error (distortion) of the network, then it becomes under represented neuron. Therefore, a new neuron is generated as a neighbor of neuron to achieve a better representation to determine the criteria for new node generation.

Figure 4 shows the new node generation at the boundary of the network .A boundary node is one that has at least one of its immediate neighboring positions free of a node. In this model, each node will have four immediate neighbors. Thus a boundary node can have from one to three neighboring positions free. If a node is selected for growth, all its free neighboring positions will be grown new nodes. New nodes are generated on all free neighboring positions, as this is computationally easier to implement than calculating the exact position of the new node.

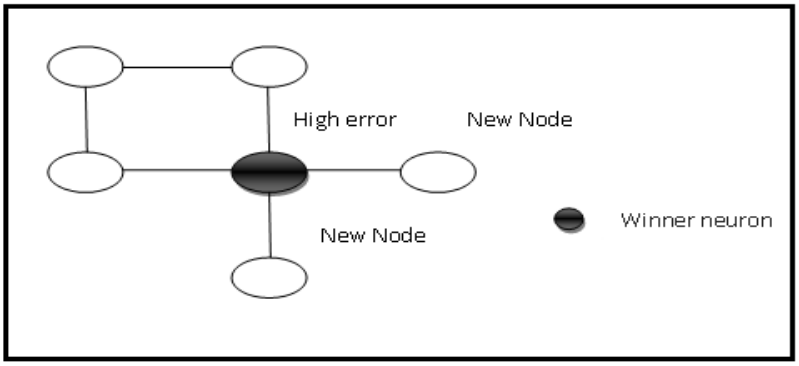


Fig. 4 New node generation at the boundary of the node

Alternatively, Figure 5 shows the new node generation at the interior of the node. If an internal node is selected for growth, four new nodes are inserted between the selected node and its four immediate neighbors. An internal node does not have free neighborhoods position. Therefore, the new added nodes and the node with the highest error are presented in the superior level of the map to guarantee a clear visualization and the development of oriented maps.

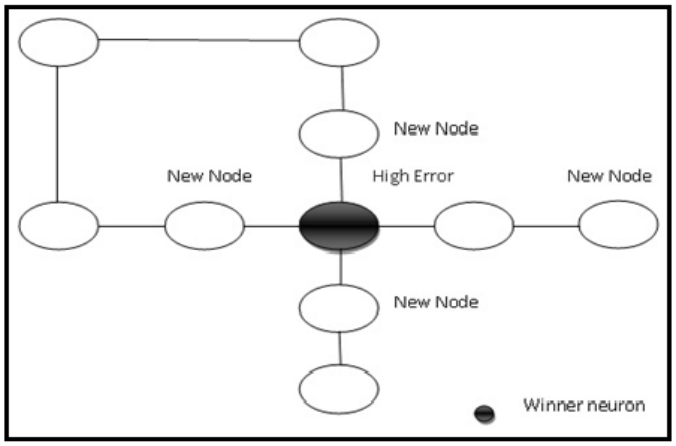


Fig. 5 New node generation at the interior of the node

4.3.3 Smoothing Phase

The smoothing phase occurs after the new node growing phase in the training mode. The growing phase stops when new node growth saturates, which can be identified by the low frequency of new node growth. Once the node growing phase is complete, the weight adaptation is continued with a lower rate of adaptation. No new nodes are added during this phase. The purpose is to smooth out any existing quantization error, especially in the nodes grown at the latter stages of the growing phase. During the smoothing phase, inputs to the network are similar to those of the growing phase. The starting learning rate (LR) in this phase is less than in the growing phase, since the weight values should not fluctuate too much without converging. The input data are repeatedly entered to the network until convergence is achieved. The smoothing phase is stopped when the error values of the nodes in the map become very small. Therefore, the smoothing phase has the following differences from the growing phase.

i) The learning rate (LR) is initialized at a lesser value.
ii) The neighborhood for weight adaptation is constrained only to the immediate neighborhood (even Smaller than in the growing phase).
iii) The rate of learning rate depreciation is smaller.
iv) No node growth

4.4 Related Work

Arthur L Hsu and saman [15], proposed enhanced topology preservation of dynamic self-organizing maps for data visualization to assist the growing of the dynamic self-organising map in achieving better topographic quality while maintaining or even improving level of quantisation error.Hiroki sasamura and Toshimishi saito [16], introduces a simple learning algorithm for growing self-organizing maps and its application to the skeletonization. To adapt the shape of the input data, map can have partial tree and loop topology. In our algorithm, the map can grow and the topology can change based on learning history of each cell. Rasika amarasiri et al [17], proposed a modified a growing self-organizing map for high dimensional data clustering. It has a dynamically growing structure that adapts to the natural structure of the data. This can be identified that the growing of the GSOM can get negatively affected when used with very large dimensional data. Guojian Cheng et al [18], proposed introduced image color reduced based on self-organizing map and growing self-organizing neural network. Color Reduction of Image is an important factor for segmentation, compression and transmission of images. An effective approach to solve color reduction problem is to consider it as a clustering problem and solve it by using some adaptive clustering methods.Alahakoon,D [19], proposed a self-growing cluster development approach to data mining. Once the clusters are identified, two additional layers on the feature map to analyses the clusters. In this paper, we present growing SOM to overcome the problem of self-organizing maps with adaptive learning rate and improved data distribution to achieve relatively fast and more reliable for effective reordering.

5. Effective Reindexing Algorithm with GSOM

The proposed algorithm is based on the ability of the GSOM to learn the "features" of the input pattern providing an appropriate output stimulus. During the learning process, the algorithm provides a palette shape clustering for searching (in the output stage of the network) the optimum indexing scheme.

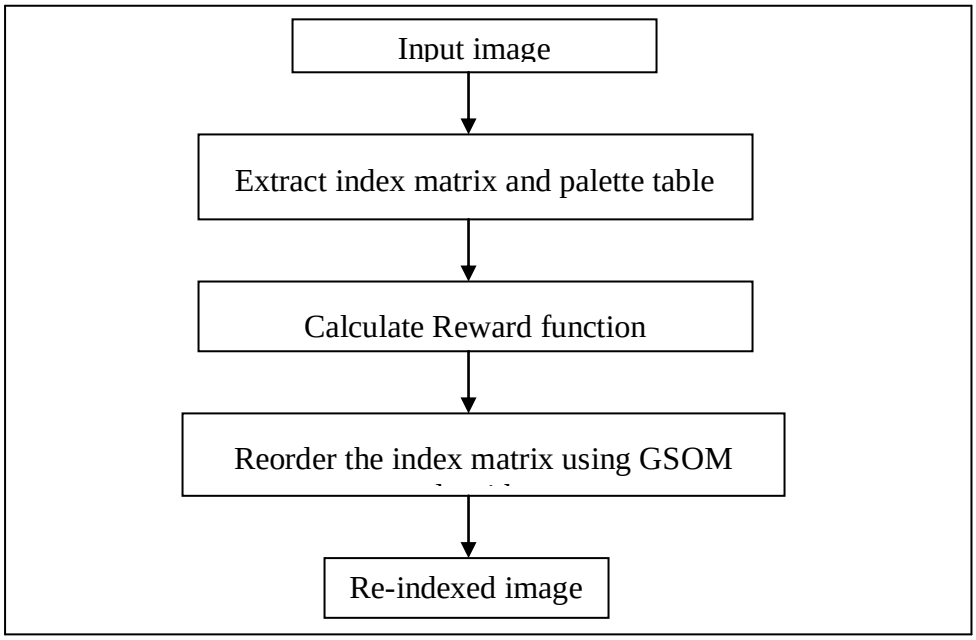


Fig. 6 Flow Diagram of Palette Re-indexing

The learning process has been modified respect to the classic algorithm in order to adapt itself to solve the palette re-indexing issue. The flow diagram of Palette re-indexing is vividly described in Fig 6.

The GSOM algorithm formation process is described in the following steps:

Step 1:

Let Q be the number of neurons of the Motor Map. Each neuron is composed by an input weight $W(t)_i^{in}$, $i = 1, \ldots, Q$ with $W(t)_i^{in} \varepsilon$ [0, 1] and output weight $W(t)_i^{out}$, $i = 1, \ldots, Q$. The range of the $W(t)_i^{in}$ values is [1,. . . ,M], where M is the overall number of colors (index) of the input image I. Reward function has been chosen:

$$Reward = -\sum_{i,j}[D(I')]^2 \tag{3}$$

This reward function is proportional to the zero order entropy of $D(I')$ [20]. The selection of reward function leads to find an optimum palette which is to minimize the entropy of the image. Luminance vector Y computed from the palette P of the image I. In case of RGB color space, the luminance can be approximated by the lightness factor. Luminance vector is given by,

$$y_i = 0.299\, r_i + 0.587\, g_i + 0.114\, b_i \tag{4}$$

Step 2:

Initialize the weight vector for each input vector which is starting at four nodes. Weight vector attributes at initial can take random values in this range. Determine the winning node that is closest to the input vector mapped to the current feature map, using the Euclidean distance. Winner neuron is the neuron with the minimum distance value in the input space.

$$d_i = \left| x_k - W(t)_i^{in} \right| \tag{5}$$

Where x_k is the input vector and $W(t)_i^{in}$ is the weight vector of each node. Input vector attributes are normalized to the range 0 to1. A value called the spread factor (SF) allows the user to control the growth of the GSOM [11].

$$\mathrm{GT} = -\mathrm{D} * \ln(\mathrm{SF}) \tag{6}$$

where D is the dimension of current node and SF is the Spread Factor. The SF allows the user to control the growth of the nodes. To achieve this control on the spread of the map, the amount of spread required is specified by the spread factor.

Step 3:

Normally low spread factor use the value between 0 - 0.3 which creates initial clusters. GT will act as a threshold value for initiating node generation. A high GT value will result in less spread out map, and a low GT will produce a well-spread map. As increasing the SF value with 0.5, increases the number of sub clusters. For well spread map, the value of spread factor will be high, normally it takes 0.8. This result in finer clustering, where the entropy is suitably reduced making it possible for effective palette reordering.

Step 4:

Calculate error value of the node. Error value is the difference between input vector and weight vector. GSOM uses a threshold value called the GT to decide when to initiate new node growth. GT will decide the amount of spread of the feature map to be generated If TE > GT, New nodes will always be grown from a boundary node. Each node will have four immediate neighbors. Where, N is number of nodes in the map. E_i is the error value of the node. The total error is used as a measure of determining when to generate a new neuron. Thus a boundary node can have from one to three neighboring positions free. The total error value is given by,

$$TE = \sum_{i=1}^{N} E_i \tag{7}$$

New nodes are generated on all free neighboring positions, as this is computationally easier to implement than calculating the exact position of the new node if a node is selected for growth, all its free neighboring positions will be grown new nodes. The newly grown nodes will be assigned initial weight values. After the iteration weight vector will be adapted by calculating the learning rate.

Step 5:

The weight vector adaptation is applied only to the neighborhood of the winner. Learning rate adaptation is also reduced exponentially over the iterations. The neighborhood, weights that are closer to the winner are adapted more [44]. The weight adaptation can be described by,

$$w_j(k+1) = w_j(k) * LR\,(x_k - w_j(k)) \tag{8}$$

where $w_j(k)$ indicates the weight vector of the nodes, LR is the Learning Rate which is normalized to 0 to 1. $w_j(k+1)$ is the weight vector adaptation of the node, for every iteration. The high LR to self organizes the weights within the regions of the map. Weight adaptation is then carried out with reducing neighborhood and learning rate. The high LR will not affect the other regions at this stage due to the map's being larger.

Step 6:

After the weight vector updating, it is possible to calculate the $D(I')$ again [25]. The ΔReward function will be computed as

$$\Delta\text{Reward} = \left(\text{Reward}^{\text{new}} - \text{Reward}^{\text{old}}\right)^2 \tag{9}$$

The average increasing of the reward function is weighted by,

$$b(t+1)_{win}^{new} = b(t)_{win}^{old} + \rho\,(\Delta\text{Reward} - b(t)_{win}^{old}) \tag{10}$$

where ρ is smoothing factor it takes the positive value between 0 and 1, $b(t)_{win}^{old}$ is winner neuron of the previous nodes.

Step 7:

If the $\Delta\text{Reward} > b(t+1)_{win}^{new}$, learning rate updated in the weight vector adaption equation. Once the node growing phase is complete, the weight adaptation is continued with a lower rate of adaptation. The starting Learning Rate (LR) in this phase is less than in the growing phase. Learning rate can be stated as,

$$LR\,(k+1) = \alpha + \psi(n) + LR \tag{11}$$

where α is the learning rate reduction and it is a constant value between 0 and 1. ψ(n) is the number of nodes in the current map. LR initially fixed as 0.9. The weight vector is updated every iteration using the learning rate. At the initial stage of growth, when the number of nodes are few, the high LR values will be reduced with the ψ(n). This reduces the fluctuations of the weights of a small map. As the network grows, the ψ(n) will take gradually higher values, reducing the value of LR. The set of differences has been computed by using different local pattern configurations. At all iteration the reward function is calculated with a check for the reward function to take zero value. The modified learning rate value is used from step 2) - 7) until the ending

criteria are verified. In Fig 7, a complete flow chart representation of the above growing SOM algorithm is clearly shown.

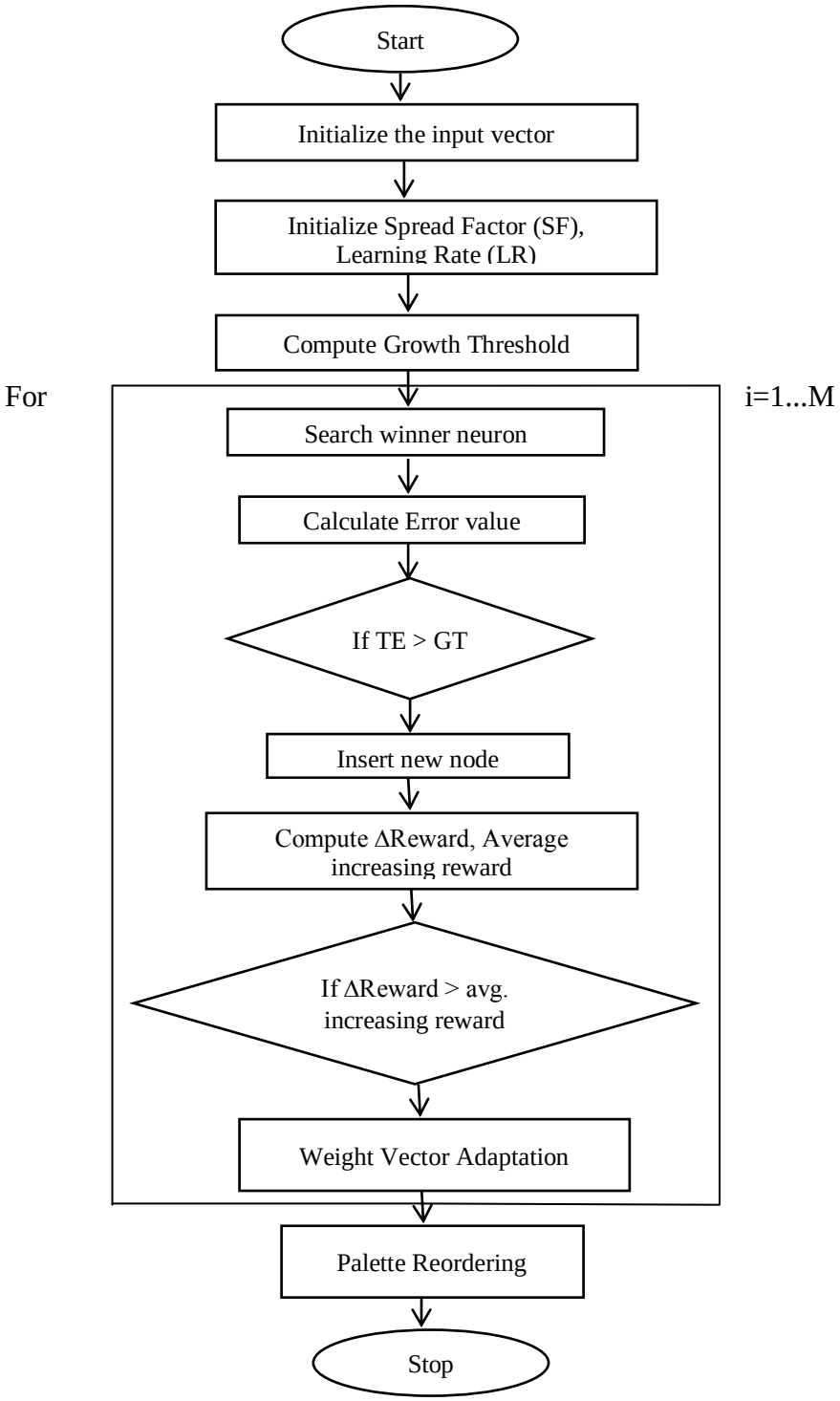


Fig.7. Flow Chart for GSOM Algorithm

6. Experimental Results

The dataset is organized in three groups: synthetic, a set of natural images also known as Kodak database, and a set of popular natural images. The last two sets contain quantized version of the same images with 256, 128 and 64 colors, respectively.

The images were initially obtained for the work from the following web address: http://www.dmi.unict.it/~iplab/MMap_reindexing/.All the dataset images are in PPM (Portable Pix Map) format. These images are converted to GIF (Graphics Interchange Format) using GIMP (GNU Image Manipulation Program) application.

6.1 Experimental Parameter Evaluation

Image with less than 64 colors, the maximum iteration was fixed to 300 and the image between 64 to 128 colors was fixed the iteration 500, the maximum iterations set to 700 for 128 to 256 colors. The initialization of following parameters values are: SF = 0.8, LR = 0.9, $\alpha = 0.5$

6.2 Simulation Outputs

Table 1 shows the different entropy comparison for all the previous method with GSOM. This entropy comparison done as an illustration for wall image in kodak database. Our proposed algorithm achieves the minimum entropy and thus reduces the bits per pixel with lossless codec.

Table.1 Different re-indexing schemes with entropy

Method	**Entropy**
Original Image	5.762
Random	6.860
Zeng's	5.010
Modified Zengs	4.928
Memon's	4.824
Battiato	5.397
SOM	4.806
APSO	4.797
GSOM	3.404

Table 2. Lossless Compression Results in bits Per Pixel Obtained With PNG

Images	**Colors**	**Original (bpp)**	**SOM (bpp)**	**APSO (bpp)**	**Proposed Algorithm (bpp)**		
					SF = 0.8	**SF = 0.5**	**SF = 0.1**
Pc	6	0.845	0.506	0.505	0.582	0. 604	0.605
Books	7	1.580	1.811	1.553	1.263	1.526	1.589
music	8	1.269	1.103	1.038	1.007	1.007	1.045
Winaw	10	0.506	0.580	0.568	0.516	0.539	0.539
Party8	12	0.429	0.345	0.297	0.398	0.399	0.399
Netscape	32	2.121	2.100	1.987	1.994	2.005	2.008
Sea_dusk	46	0.323	0.092	0.095	0.103	0.104	0.103
Benjerry	48	1.367	1.222	0.997	0.761	0.842	0.851
Gate	84	2.957	2.747	2.404	2.620	2.632	2.632
Descent	122	2.952	2.862	2.620	2.674	2.676	2.678
Sunset	204	2.608	2.049	2.139	2.151	2.151	2.152
Yahoo	229	2.053	1.861	1.695	1.866	1.868	1.868
Serrano	256	2.897	2.975	2.610	2.173	2.334	2.407
Ghouse	256	4.999	4.642	4.312	2.648	3.352	3.953
Frymire	256	2.680	2.863	2.431	2.340	2.340	2.340
Fractal	256	6.951	5.677	5.873	5.004	5.209	5.230
Cwheel	256	2.769	2.734	2.535	1.355	1.775	2.137
Clegg	256	5.699	4.597	4.556	3.900	4.348	4.434
Average	-	**2.5002**	**2.264**	**2.123**	**1.853**	**1.983**	**2.053**
Natural	64	7.360	3.009	2.645	2.644	2.643	2.646
	128	4.516	3.776	3.128	3.128	3.127	3.128
	256	5.528	4.605	4.667	3.378	4.164	3.727
Average	-	**5.801**	**3.796**	**3.48**	**3.05**	**3.311**	**9.501**

Table 2 depicts the reordered image with achieving zero order entropy compared to the original image. To confirm the increased compressibility of our method, the reordered image was compressed losslessly using PNG codec and the results are presented in terms of bits per pixel. It is evident from the results that, the comparative analysis of the algorithm facilitates decrease in the index differences of the palette image thereby reducing the zero-order entropy value.

From Table 2, initially values of the bpp are obtained through GSOM with SF = 0.1. As there is an increase in SF value optimal reordering takes place making the bits per pixel reduced accordingly. The results of our method outperform the images compressed by SOM [12] and APSO [22].

Table 3. shows the values of re-indexing time taken by the algorithm for images with different colors, from which it is vivid that the re-indexing time drops down

fairly well. This highlights the performance of the proposed method in comparison with the method proposed in [21] and Adaptive Particle Swarm Technique in [22].

Table 3 Comparison of re-indexing time for images of different colors in previous method with the proposed algorithm

Sl. No.	Images	No. of colors	Joshua et al [21]	APSO [22]	GSOM
1	Clegg	256	40	16.56	2.2
2	Descent	128	7	4.727	0.48
3	Seadusk	64	2	0.77	0.21

6.3 Graphical User Interface

GUI is a type of user interface that allows users to interact with electronic devices through graphical icons**.** Figure 9. shows the GUI front end for user interface developed to show the bits per pixel value for both SOM and DSOM method by clicking bpp push button as an illustration. In the similar fashion other simulation outputs can also be generated by GUIs (also known as graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application for the all the algorithms considered for comparison .

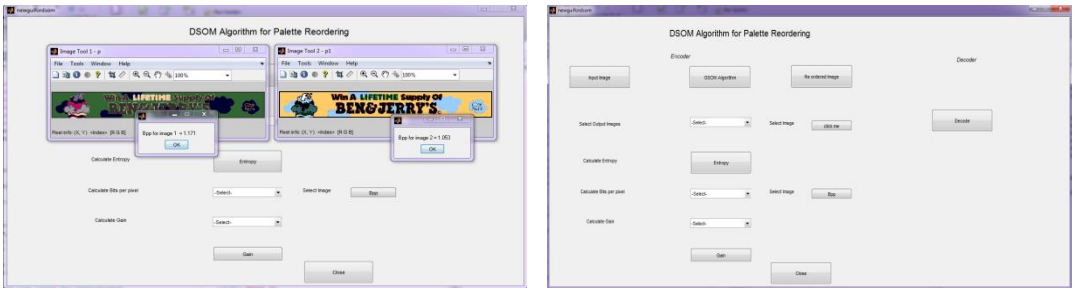


Fig. 9. a) GUI front end b) bpp values for SOM and GSOM

7. Conclusion

In this paper, we described a technique that shows a good performance on the optimum palette scheme generation without any initial assumption on the palette index scheme or on the pixel distribution. Indeed, it is interesting to note that a palette re-indexing algorithm presented in literature is based on the assumption that the differences of neighboring pixels of well reordered images. The GSOM algorithm

robustly solves this problem by including adaptive learning rate and increasing value of spread factor for dynamic pixel distribution. This leads to an effective re-ordered image in terms of bpp and reduced zero order entropy. Also experimental results assure that the proposed method excels with improved computational speed. This work can be extended by applying the algorithm to classification applications, object based image analysis, clustering sensor networks & data mining.

8. References

[1] A. Zaccarin and B. Liu, "A novel approach for coding color quantized images," IEEE Trans. ImageProcess., vol. 2, no. 4, pp. 442–453, Oct. 1993.

[2] A. C. Hadenfeldt and K. Sayood, "Compression of color-mapped images,"IEEE Trans. Geosci. Remote Sens., vol. 32, no. 5, pp. 534–541,May 1994.

[3] M. J. Gormish, "Compression of palletized images by color," presented at the IEEE Int. Conf. Image Processing, 1995.

[4] Tay Vaughan, Multimedia: Making It Work, McGraw Hill, 2011.

[5] K. Sayood, Lossless Compression Handbook, Academic, New York, 2002.

[6] S. Battiato, G. Gallo, G. Impoco, and F. Stanco, "An efficient re-indexing algorithm for color-mapped images," IEEE Trans. Image Process., vol. 13, no. 11, pp. 1419–1423, Nov. 2004.

[7] A. Pinho and A. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images," IEEE Trans. Image Process., vol. 13, no. 11, pp. 1411–1418, Nov. 2004.

[8] W. Zeng, J. Li, and S. Lei, "An efficient color re-indexing scheme for palette-based compression," in Proc. 7th IEEE Int. Conf. Image Processing, pp. 476–479, Sep. 2000

[9] S. Battiato, F. Rundo, F. Stanco, "Color Palette Images Re-indexing by Self Organizing Motor Maps", Eurographics Italian Chapter Conference pp 241-246,2006

[10] S.-C. Pei, Y.-T. Chuang, and W.-H. Chuang, "Effective palette indexing for image compression using self-organization of kohonen feature map," IEEE Trans. Image Process., vol. 15, no. 9, pp. 2493– 2498, Sep. 2006.

[11] Damminda Alahakoon, Saman K. Halgamuge, *Member, IEEE*, and Bala Srinivasan, "Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery", IEEE Transactions on Neural Networks, vol. 11, no. 3, May 2000.

[12] S. Battiato, F. Rundo, F. Stanco, "Self organizing motor maps for color-mapped image re-indexing", IEEE Trans. Image Process. 16, 2905–2915,2007.

[13] T. Kohonen, "Self organized formation of topologically correct feature maps," Bio.Cybern., vol. 43, 1972.

[14] T. Martinetz, H. Ritter, and K. Schulten, "Neural Computing and Self-Organizing Maps". Reading, MA: Addison-Wesley, 1992.

[15] Arthur L Hsu and saman K. Halgamuge, "Enhanced topology preservation of dynamic self-organizing maps for data visualization", IFSA World Congress and NAFIPS International Conference, vol. 3, 2003.

[16] Hiroki Sasamura, Toshimichi Saito, "A Simple Learning Algorithm for Growing Self-organizing Maps and Its Application to the Skeletonization", IEEE Int. joint Conf. on neural networks, vol. 1, 2003.

[17] Rasika amarasiri, Damminda Alahakoon, Kate A. Smith "HDGSOM: A modified a growing self-organizing map for high dimensional data clustering", IEEE International Conference on Hybrid Intelligent Systems, 2004.

[18] Guojian Cheng, Jinquan Yang, Kuisheng Wang, Xiaoxiao Wang,"Image color reduction based on self-organizing maps and growing self-organizing neural networks",IEEE International Conference on Hybrid Intelligent Systems, 2006.

[19] Alahakoon, D. Halgamuge, S.K. "A self-growing cluster development approach to data mining",

[20] S. Battiato, F. Rundo, F. Stanco, "A Novel Image Re-Indexing by Self OrganizingMotor Maps", In Proceedings of IEEE International Conference on Image Processing 2007 (ICIP07), pp. VI-1 – VI-4, USA 2007.

[21] J. Van Hook, F. Sahin, Z. Arnavut, "Improved lossless compression of color palette images by re-indexing with particle swarm optimization", Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009.

[22] P. Niraimathi, M. S. Sudhakar, K. Bhoopathy Bagan, "Effficient reordering algorithm for color palette image using adaptive particle swarm technique", Applied Soft Computing 12, 2199–2207, 2012.

[23] A. J. Pinho and A. J. R. Neves, "A note on Zeng's technique for color reindexing of palette- based images," in the IEEE Sig. Process. Letters, vol. 11, no. 2, pp. 232–234, Feb. 2004.

[24] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.

[25] S.H. Liao, C.H. Wen, Artificial neural networks classification and clustering of methodologies and applications—literature analysis from 1995 to 2005, ExpertSystems with Applications 32 (2007) 1–11.

[26] T. Mu, A.K. Nandi, Breast cancer detection from FNA using SVM with different parameter tuning systems and SOM-RBF classifier, Journal of the Franklin Institute 344 (2007) 285–311.

[27] R. Miikkulainen, J.A. Bednar, Y. Choe, J. Sirosh, Modeling self-organization in the visual cortex, in:E. Oja, S. Kaski (Eds.), Kohonen Maps, Elsevier, Amsterdam,1999.

[28] A. Forti, G.L. Foresti, Growing hierarchical tree SOM: an unsupervised neural network with dynamic topology, Neural Networks 19 (2006) 1568–1580.

[29] P.C. Chang, T.W. Liao, Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory, Applied Soft Computing 6 (2006)198–206.

[30] R. Xu, D. Wunsch, Survey of Clustering Algorithms, IEEE Transactions on NeuralNetworks 16 (3) (2005) 645–678.

[31] T. Kohonen, Self-Organizing Maps, 3rd edition, Springer-Verlag, Berlin, 2001.

[32] J. Blackmore, K. Miikkulainen, Incremental grid growing: encoding highdimensional structure into a two-dimensional feature map. in: Proceedings of the IEEE International Conference on Neural Networks (ICNN 1993), 450–455.

[33] B. Fritzke, Growing cell structures—a self-organizing network for unsupervised and supervised learning, Neural Networks 7 (9) (1994) 1441–1460.

[34] B. Fritzke, A growing neural gas network learns topologies, in: Advances in Neural Information Processing Systems (NIPS'94), Vol. 7, MIT Press, Cambridge,1995, pp. 625–632.

[35] J. Zhang, D. Dai, An adaptive spatial clustering method for automatic brain MR image segmentation, Progress in Natural Science 19 (2009) 1371–1382.

[36] S. Marsland, J. Shapiro, U. Nehmzow, A self-organizing network that grows when required, Neural Network 15 (8-9) (2002) 1041–1058.

[37] B. Fritzke, Let it Grow—Self Organizing Feature Maps with ProblemDependent Cell Structure Amsterdam, The Netherlands, 1991, pp. 403–408.

[38] "Growing cell structure: A self organizing network for supervised and un-supervised learning," Neural Networks, vol. 7, pp. 1441–1460, 1994.

[39] Hiroki Sasamura, Toshimichi Saito, "A simple learning algorithm for growing self-organizing maps and its application to the skeletonization", IEEE International joint Conference on Neural Networks, Vol. 1, Jun. 2003.

[40] Norashikin Ahmad, Damminda Alahakoon, "Generating concept trees from dynamic self-organizing map", World Academy of Science, Engineering and Technology, 41, Jun. 2010.