International Journal of Control and Automation
Vol. 11, No. 5 (2018), pp.143-162
http//dx.doi.org/10.14257/ijca.2018.11.5.13

SOM Aided Visualization and Retrieval of Flower Database Using Tree-Structured Representation

M. K. M. Rahman

Dept. of EEE, United International University,
United City, Madani Avenue, Badda, Dhaka-1212, Bangladesh
masuk@eee.uiu.ac.bd

Abstract

This paper presents a new tree-structured image representation together with an improved self-organizing map (SOM)-based system for visualization and retrieval of flower database. The tree-based representation organizes the objects of a flower image in a 3-level hierarchy, where the root node represents whole image and the bottom-level nodes represent local objects. Such representation integrates both global and local image features that help to retain the benefits of both traditional flat features and region-based approach. We have developed a Multi-layer SOM (MLSOM) based system that can perform image retrieval in a much efficient way compared with any direct method of image retrieval. The MLSOM organizes the flower images in a much better way compared with other SOM-based methods. Experimental results corroborate that the proposed tree-representation together with MLSOM-based method can perform better than other feature representations and methods in terms of visualization, speed and accuracy of retrieval.

Keywords*: Flower image retrieval, multi-layer self-organizing map (MLSOM), self-organizing map (SOM), tree-structured image representation*

1. Introduction

1.1. Flower Image Retrieval

Flower image processing is an interesting problem that can be used in a wide range of applications including live plant resource and data management, and education on flower taxonomy. There are over 250,000 named species of flowering plants and many plant species have yet to be classified and named. It is useful that for a given query plant image, one can determine relevant images of the same flower species from a flower data set. This is the process of image retrieval. Flower image retrieval, however, is and will still be one of the most difficult tasks because of a number of factors. These include the lack of accepted models, the great biological variations of species, the quality of flower images, and the difficulty in locating flower regions. There have been several methods proposed for processing flower images [1,2,3]. An iterative segmentation algorithm was proposed to obtain flower regions [1] that contain the interior of the flower's silhouette. A general contour-oriented shape dissimilarity measure for comparing potato flowers was proposed in [2]. A feature extraction and supervised-neural-network-based method was developed in [3] for performing recognition of sixteen wild flowers, but the method can only process images with one flower in each image.

1.2. Traditional Feature Representation

It is generally agreed that an effective feature representation of flower image content

Received (October 28, 2017), Review Result (March 22, 2018), Accepted (April 5, 2018)

is the most important issue for image retrieval. There are many methods [4,5,6] for extracting features from flower images. Traditionally, flat vector-type image features like color, texture and their combinations [4-7] are used for image processing, where the length of feature-vector does not change from one image to another. Color histogram [5-6] is computationally efficient and generally insensitive to small change of camera positions. However, color histogram represents only one aspect of many others, and features like textures, shapes, sizes and positions of the objects cannot be captured. As a result, images with similar histograms may have dramatically different semantics [8]. Texture feature is used to describe local property of an image that contains the information about pixels' mutual positions and distances. Combination of color and texture features [7] is shown to be more effective to describe image content compared with each individual type. These image features are useful for comparing the overall image similarity, especially the natural scene images like trees, sky, water etc. But they are unable to describe high level and object-based properties of image contents. This has certainly made flat vector-type features not suitable for processing flower images, which consists of at least two objects, namely flowers, and background.

1.3. Region-based Feature Representation

The region-based approaches are introduced to overcome the above limitations. Instead of assigning features to the whole image, features are assigned by decomposing image into regions and similarity between two images is expressed as the similarity among their regions [7,9-11]. IRM [9] is one of popular region-based method that allows the matching of a region from one image against several regions from other image and it reduces the adverse effect of inaccurate segmentation. The similarity between two images is then expressed as the weighted similarity among their regions [9].

1.4. Tree-Structured Image Representation

A tree-structured image representation is proposed in this paper. Unlike traditional flat vector-type features, the tree-structured representation has a superior capability to represents rich information of complex object [12-14]. A simple two-level tree-structure representation can be formed from region-based approach, where the root or parent node represents the global features like color histogram, and child-nodes represent local characteristics from image regions. Clearly, tree-structured image representation has been shown to be effective for image processing and image analysis [15]. Other successful works can be found in [12-14,16-18,25]. A two-level tree-structured image representation was proposed in [12,14] for improved image retrieval and classification. The approach is useful for generic image retrieval application where the nature of image content is unknown. However, such tree representation can be further improved for domain-specific application such as flower image processing. In [16], a Binary Space Partition (BSP) tree representation was used for coding images in terms of segmented regions. Recently Salembier and Garrido [17] proposed a BSP-tree-based region-oriented image representation, which can be used for image processing, segmentation and information retrieval. Basically, the BSP tree concentrates on a compact and structural representation of regions. Though BSP is able to process images efficiently, partitioning a region into two sub-regions cannot always deliver a good representation of objects under a real image scenario. In the case of flower images, flowers, leaves, and other backgrounds are clearly belongs to three categories of objects. Thus, a better tree structure, other than BSP, is required to represents the contents of flower image.

1.5. SOM Approach for Image Visualization and Retrieval

While feature representation plays an important role for retrieval accuracy, speed of the retrieval system is another major concern. Speed can be a critical issue when tree-structured features representation is used in a direct method where a query image is compared with all the images in the database without adopting any technique like clustering. Self-organizing map (SOM) is a powerful tool to visualize any database and to perform efficient retrieval of data [12]. The basic idea of SOM is to organize data through clustering, and compare a query data to limited cluster centers called neurons. Thus, SOM can provide a topologically-ordered visualization of the data, and can perform retrieval task which is much faster than any direct method. In handling the tree-structured data, SOM [23] is adopted in the field of image processing. In the SOM-SD model [23], an improved SOM model, all the nodes of a tree from bottom nodes to root node are processed within a single SOM layer. There are, however, several drawbacks in this SOM-SD model. The crucial point of this model is that the 2-D output positions of winner neurons for child nodes constitute one part of the input vector of a parent node in the tree. This part of input may be changed after the updating of weight vectors. Thus, this leads to a slow training convergence. Furthermore, a single SOM layer is used to process all the nodes of a tree structure. This has made visualization and image retrieval results unsatisfactory when SOM-SD is used.

1.6. Proposed Method

In this work, a new 3-level tree-based image representation is proposed and an effective SOM-based method is developed for visualization and retrieval of flower images. The proposed representation uses a two-step segmentation process that organizes the contents of a flower image in a 3-level tree. Unlike BSP tree [17], this method can divide an image or region into two or more regions or sub-regions. Also the proposed 3-level structure is more meaningful for flower image compared with 2-level structure that was applied for general image [12]. The representation assigns traditional features like color-histogram to the root node of the tree, whereas region-based local features are assigned to the 2nd level and 3rd level nodes. Thus the tree structure integrates the global and local image features that help to retain the benefits of both traditional flat features and region-based approach. Processing such high-dimensional and complex tree-structured data is computationally very demanding, especially when data retrieval is performed using a direct method, *i.e.*, comparing a query to all the data in the database. In this work, we have used a multi-layer SOM (MLSOM) model [13] that not only speeds-up the retrieval task significantly, but also provides an effective visualization map. Our MLSOM uses three separate layers of SOM to process the nodes from three different levels of tree, where root nodes (representing whole image) are mapped into the neurons of the top layer of MLSOM. This makes MLSOM to achieve much better results in terms of visualization, clustering, training convergence and retrieval performance when compared to previous SOM-SD model that relies on using a single SOM to process all 3-level nodes. The proposed tree-structured representation is compared with other traditional feature representations in terms of visualization and retrieval accuracy. We have also compared our MLSOM-based system with other methods such as direct method IRM [9] and SOM based methods. Our obtained results indicate that the proposed method can deliver better visualization effect and better image retrieval performance in terms of both accuracy and speed.

The rest of this paper is organized as follows. In Section 2, the tree-structured representation of flower images is described. Section 3 elaborates the structure of MLSOM for visualization and retrieval of flower images. Section 4 presents the experimental results and discussions. Finally, conclusion is drawn in Section 5.

2. Tree-Structured Feature Representation

2.1. Flower Image Representation

A tree-structured image representation is used to structurally organize the contents of a flower image. A two-step segmentation technique is used to generate a 3-level tree-structured representation of flower image. First, image segmentation is performed based on color. A flower image is partitioned into homogeneous color areas. Second, each segmented region is partitioned into sub-regions where all pixels are coherent in each sub-region. Hence, a root node, second-level nodes, and third-level nodes, which form the 3-level structure, are used to represent the whole image, the homogeneous color regions, and the sub-regions respectively. Figure 1 demonstrates this 3-level tree-structured representation. This partition scheme is designed to represent perceptual objects in a flower image in the best possible way. For example, all the flowers in a flower image are organized into node N_3 as shown in Figure 1.

In the proposed tree-structured representation, the global and local features are assigned to different nodes of a tree according to their hierarchy. Global features are assigned to a node at the top of the tree, whereas local features are assigned to nodes at the rest of the tree. Color histograms, the most global features, are assigned to the root node representing the whole image. The second-level nodes, representing the homogeneous color area, contain local features such as color, shape, and size. The extracted features for the third level nodes are textures, shapes, and sizes of the sub-regions. Thus, this tree-structured representation enables separation and organization of different types of image features in a meaningful way. For example, the texture feature is more appropriate to describe a sub-region represented by the third level node than the whole image represented by the root node. Although the size and the shape features are repeated in the second and the third level, they do not necessarily represent the same region or sub-region. They are weighted differently in these two levels for measuring the similarity between two regions or sub-regions. For example, the shape and the size features of the nodes in the third level are assigned with more weights compared with the nodes in the second level because they are more meaningful.

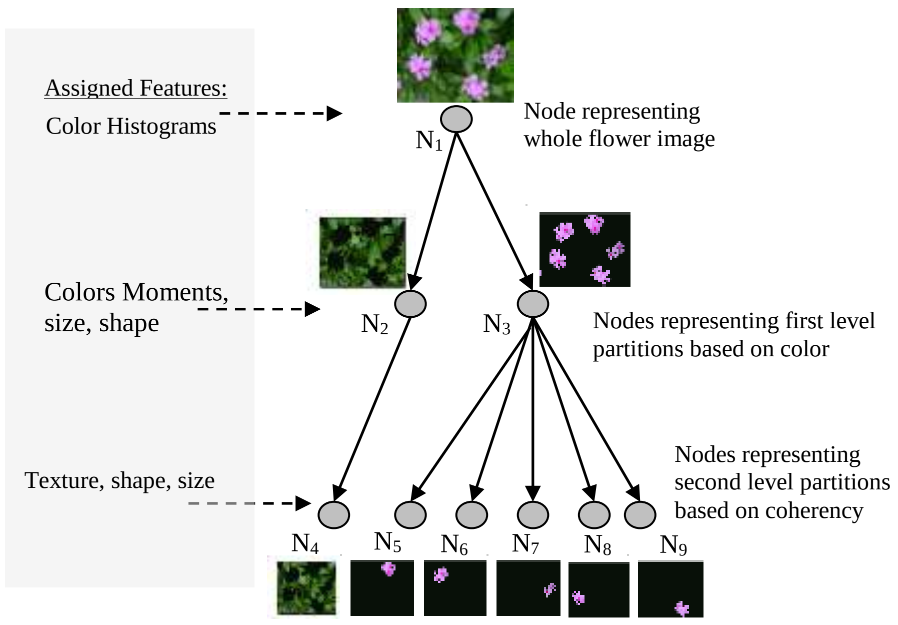

Assigned Features:
Color Histograms
Node representing whole flower image
N1
Colors Moments, size, shape
N2
N3
Nodes representing first level partitions based on color
Texture, shape, size
Nodes representing second level partitions based on coherency
N4
N5
N6
N7
N8
N9

Figure 1. Illustration of Tree-Structured Flower Image Representation. Different Types of Features are Assigned to Different Levels of Nodes

2.2. Image Segmentation

Color clustering [19] is a popular choice among image segmentation techniques [20,21] proposed in the last two decades. However, depending on data representation and the nature of image sets, different segmentation methods usually perform differently under various applications. In this study, we have implemented a 2-step segmentation method, which is computationally inexpensive and suitable for this flower-processing task. The number of clusters needs not to be predefined. This is essential for this application because the number of flower objects is not fixed in all flower images. In our proposed method, HSV color model is used because it is able to closely follow the human visual system. Also, HSV has a better color coherence feature compared with other color models. The three HSV channels are used consecutively to cluster image pixels in the color domain.

To begin the first-step segmentation process, each of the hue, saturation and value (HSV) channels of an image is quantized into 16, 10 and 10 levels respectively. Since the hue channel contains the most significant color information, higher quantization levels are used for the hue channel to form the initial clusters. These values of quantization level are found to be effective for the flower images. In order to determine the first cluster in the hue channel, the highest peak is firstly found from its histogram. Then the first cluster is formed by the pixels lying within a range of the hue values around the peak of the histogram. The range is determined by searching a left and a right margin of the hue values around the peak, whereas pixels within this range will not be considered for the next clustering. The left and right margins are found by a number of conditions based on local minima, decay along the histogram, the distance between the peak and the margin, and the number of pixels between them. Similarly, the second cluster is found by searching the peak of the remained histogram and the range around the peak. Thus the above process is repeated until all the pixels are clustered. To obtain meaningful clusters, the minimum size to form a cluster is set to 5% of the total pixels in an image. Once clusters are formed using hue channel, each cluster can form sub-cluster using value and saturation channel consecutively. The complete clustering process is summarized as follows:

```
Step 1: Find the initial clusters of image pixels using the histogram of hue channel.
Step 2: For each cluster found in step 1, find sub-clusters using the histogram of the
        saturation channel.
Step 3: For each of the newly formed clusters after the above two steps, find sub-
clusters using the histogram of the value channel.
```

After clusters are formed at the first partitioning, an image is segmented into homogeneous color regions shown in Figure 1. Each color region is then partitioned into sub-regions of coherent pixels at the second-step partitioning. Two pixels are considered coherent and are kept in the same sub-region, if one pixel is the direct neighbor of the other. The minimum region size for the second-step partitioning must be more than 1% of the whole image size. Figure 1 demonstrates the second-step partitioning. It should be noted that the color partitioning segments an image into several regions or groups of region with similar colors. The segmented objects after the color partitioning may result with different combinations of regions and different number of nodes as well.

2.3. Feature Extraction

As a global feature, color histograms are used as the attributes of root nodes. Each of the HSV color channels is quantized into 16 levels to calculate color histogram features.

$$\boldsymbol{F}^1 = [h_{H1}, \ldots, h_{H16},\ h_{S1}, \ldots, h_{S16},\ h_{V1}, \ldots, h_{V16}\ ], \tag{1}$$

where $h_{H_1} = \frac{n_{H_1}}{n_T}$, n_{H_1} is the total number of pixels that fall into the first quantized hue level and n_T is the total number of pixels in the image. Similarly, other components are calculated. The features used for regions at the second-level nodes of a tree are color moments, shape and size. The complete feature vector is

$$\boldsymbol{F}^2 = [\mu_H \quad \mu_S \quad \mu_V \quad \sigma_H \quad \sigma_S \quad \sigma_V \quad s_1 \quad s_2 \quad s_3 \quad A], \tag{2}$$

where μ_H, μ_S and μ_V are the means of the 3 HSV channels of a region, σ_H, σ_S and σ_V are their corresponding standard deviations, and A is normalized region size. To describe the shape property of a region, three shape features s_1, s_2 and s_3 are used. They are normalized inertia [9] with orders from 1 to 3 that are shown to be useful for region-based image retrieval. For a region *M* in the Euclidean space $\Re^2$, the inertia with order α (=1,2,3) is:

$$I_\alpha(M) = \sum_{(x,y)\in M} \left[(x-\bar{x})^2 + (y-\bar{y})^2\right]^{\alpha/2} \Big/ \left[V(M)\right]^{1+\alpha/2}, \tag{3}$$

where (*x*, *y*) is a point in region *M*, $(\bar{x}, \bar{y})$ is the centroid of the region, α is the order and *V(M)* is the number of pixels in the region. The value of the above inertia becomes minimum when the pixels of the image region are arranged in a circular shape. Let I_1^o is the minimum value achieved by Eq. (3) for the pixels from a circular image region and using α=1. Then the shape feature s_1 is calculated as $I_1(M)/I_1^o$ that indicates the closeness of the region to circular shape. Other two features s_2 and s_3 are calculated similarly by using α=2 and α=3.

The features used for sub-regions at the third-level nodes are texture, shape and size. The complete feature vector is

$$\boldsymbol{F}^3 = [\mu_{t_1} \quad \mu_{t_2} \quad \mu_{t_3} \quad \sigma_{t_1} \quad \sigma_{t_2} \quad \sigma_{t_3} \quad s_1 \quad s_2 \quad s_3 \quad A], \tag{4}$$

where μ_{t_1} μ_{t_2} μ_{t_3} σ_{t_1} σ_{t_2} and σ_{t_3} are means and standard deviations of a region's wavelet features, size and shape features are similar to that of second level nodes. The moments of wavelet coefficients in high frequency bands, which represent energy in the high frequency bands by the Daubechies-4 wavelet transform [22], are effective in representing regional texture [10]. After one-level wavelet transform of the intensity flower images, an image block is decomposed into the four frequency bands: the *LL*, *LH*, *HL*, and *HH* bands. The mean and standard deviation values in the *LH*, *HL* and *HH* bands are then used as texture features [10].

Table 1. The Relative Weights of the Features for Comparing Similarity between a Node in the 2nd/3rd Level Nodes of the Tree Structure and a Neuron at the 2nd/3rd SOM Layer

	Second level/layer									
Features	A	s_1	s_2	s_3	μ_{c_1}	μ_{c_2}	μ_{c_3}	σ_{c_1}	σ_{c_2}	σ_{c_3}
Weights	7%	5%	5%	5%	21%	21%	21%	5%	5%	5%
	Third level/layer									
Features	A	s_1	s_2	s_3	μ_{t_1}	μ_{t_2}	μ_{t_3}	σ_{t_1}	σ_{t_2}	σ_{t_3}
Weights	10%	7%	7%	7%	18%	18%	18%	5%	5%	5%

To summarize, we have used 48, 10 and 10 features for nodes in 1st, 2nd and 3rd level of tree. For 2nd and 3rd level nodes' features, weighted similarity scheme is used to evaluate the similarity in the feature space. The relative weights defined for the above features are empirically given in Table 1. As color and texture are the most important features, more weights are assigned to them. The experimental studies described in Section 4 show that the relative weights defined in Table 1 are suitable for image retrieval.

3. Visualization and Retrieval of Flower Images by MLSOM

3.1. Self-Organizing Map (SOM)

Self-Organizing Map (SOM) consists of *M* neurons located on a regular low dimensional grid, which is usually in 1-D or 2-D. The lattice of the 2-D grid is either hexagonal or rectangle. The basic SOM learning algorithm is iterative. Each neuron *i* has a *d*-dimensional feature vector $w_i = [w_{i1}, \ldots, w_{id}]$. At each training step *t*, a sample data vector *x(t)* is randomly chosen from a training data set. Distances between *x(t)* and all feature vectors are computed. The winning neuron, denoted by *c*, is the neuron with the feature vector closest to *x(t)*:

$$c = \arg\min_i \|x(t) - w_i\| \quad , i \in \{1, \ldots, M\} \quad . \tag{5}$$

At each training step, the winner neuron and its neighbor neurons are updated through moving its feature vector towards the data vector. The weight-updating rule in the sequential SOM algorithm is written as

$$w_i(\mathrm{t}+1) = \begin{cases} w_i(t) + \varepsilon(t) h_{ic}(t)(x(t) - w_i(t)), & \forall i \in N_c \\ w_i(t), & \text{otherwise} \end{cases} , \tag{6}$$

where $\varepsilon(t)$ is learning rate parameter that decrease monotonically with time, N_c is a set of neighboring neurons of the winning neuron, and $h_{ic}(t)$ is the neighborhood kernel function that define closeness of a neuron to the winning neuron *c*. Through the iterative training procedure, the neurons' feature vectors become topologically ordered, while different neurons become representative of different data samples.

3.2. Processing Tree-structured Image Data by MLSOM

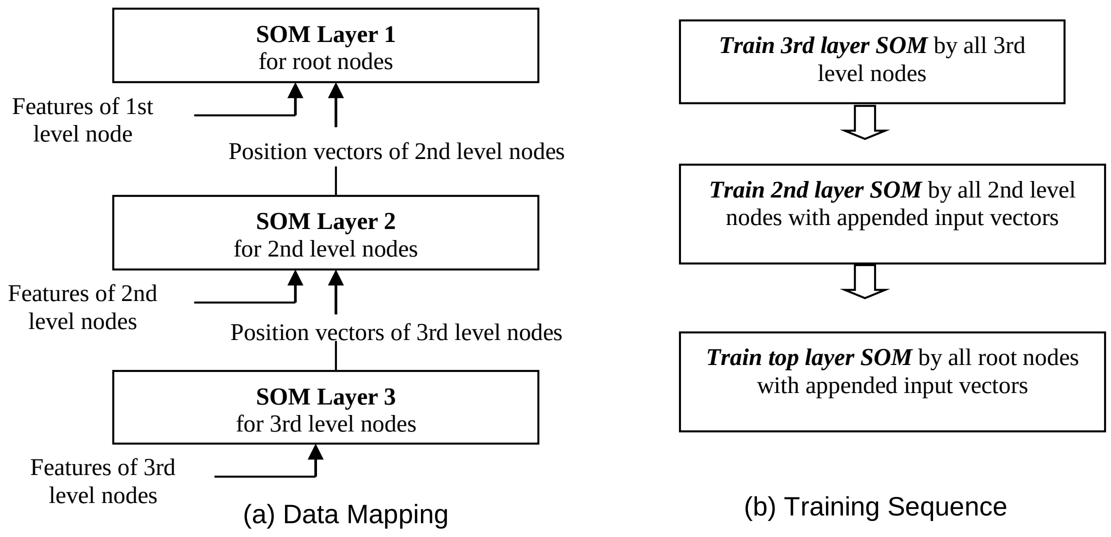

SOM Layer 1
for root nodes
Features of 1st level node
Position vectors of 2nd level nodes
SOM Layer 2
for 2nd level nodes
Features of 2nd level nodes
Position vectors of 3rd level nodes
SOM Layer 3
for 3rd level nodes
Features of 3rd level nodes
(a) Data Mapping
Train 3rd layer SOM by all 3rd level nodes
Train 2nd layer SOM by all 2nd level nodes with appended input vectors
Train top layer SOM by all root nodes with appended input vectors
(b) Training Sequence

Figure 2. (a) Illustration of the Mapping of Tree-Structured Data into 3-Layer SOM. (b) Illustration of Training Sequence of MLSOM

Generally, an ordinary SOM is unable to process tree-structured data because of the flat vector-type feature representation. To solve this problem, a MLSOM with 3 layers is used with each layer processing the nodes of the corresponding level of the tree. For MLSOM the local features of the 2nd and the 3rd level nodes of a tree are compressed and appended to the global features of the root node of the tree. Therefore, the input to the top SOM layer is a flat feature containing both global and local features that can describe the flower images effectively. The compression of the 2nd and the 3rd level nodes is conducted by the 2nd and the 3rd layer SOM. Figure 2 illustrates the processing of the 3-level nodes by using the 3-layers MLSOM. The processing is performed in a bottom-up fashion starting with the 3rd level nodes. All the 3rd level nodes are mapped into the neurons after the completion of training of the 3rd layer SOM. Thus each 3rd level node has an associated neuron and the position vector [x, y] of that neuron on the SOM map is a compressed representation of the node's features. These compressed 2-D vectors from 3rd level nodes are appended to the feature vectors of their parent nodes in 2nd level, forming a hybrid-input vector for the 2nd layer SOM. Similar to 3rd level, 2nd level of nodes are mapped to the neurons after the completion of training of the 2nd layer SOM. Then the 2-D positions of neurons in the output space of the 2nd layer SOM are compressed vectors of all the 2nd level nodes. These compressed 2-D vectors are appended to the feature vectors of the top-level nodes as the hybrid-input vectors of the top layer SOM.

In our case, a tree structure is represented by a set of nodes as $\{X_1^1, X_2^2, \ldots\ldots\ X_r^3\}$, where r is the total number of nodes of the tree and X_i^j is node i at the jth level of the tree. A node X^k at the kth level is represented by a vector $X^k = \left[f_{1x},\ f_{2x}, \cdots f_{mx},\ n_{1x}, n_{2x}, \cdots n_{qx}\right]^T$, where f_i represents the ith feature of node X^k, $n_{ix}\,(=[x_i, y_i]^T, x_i \in [0,1], y_i \in [0,1])$ is a normalized 2-D position vector, m is the number of features, and q is the maximum number of child nodes at kth level. Some of the position vectors can contain zero vectors, when the number of child nodes of a node X is less than q. Section 3.3 discusses more details about position vectors. It should be noted that f_{ix} corresponds to F^1, F^2 or F^3 depending the level of the node and maximum number of feature (m) is 48, 10 and 10 for level 1, 2 and 3 respectively. For a 3-layer MLSOM, the weight vector W^k of a neuron at the kth layer is represented by $W^k = \left[f_{1w},\ f_{2w}, \ldots, f_{mw},\ n_{1w}, n_{2w}, \ldots, n_{qw}\right]^T$, where f_{iw} is the ith weight and $n_i\ (=[x_i, y_i]^T)$ is a 2-D vector.

To find the best-matching neuron, the following similarity measure is used between a node X^k at level k and the weight vector W^k of a neuron at layer k:

$$S\left(X^k, W^k\right) = \mu \sum_{i=1}^{m} w_{f_i} . \left\{1 - abs\left(f_{ix} - f_{iw}\right)\right\} + (1-\mu)\frac{1}{q}\sum_{j=1}^{q}\left\{1 - d\left(n_{jx} - n_{jw}\right)\right\}, \qquad (7)$$

where w_{f_i} is the normalized weight associated with the ith feature as empirically specified in Table 1, and μ is a parameter to control weight between global and local features. $d(.)$ is the Euclidean distance between two position vectors. The first and second parts on the right hand side of the equation represent the global and local similarity respectively. For example of a root node, the first part represents the similarity in terms of histograms (F^1). The second part represents the overall similarity in terms of second level nodes whose characteristics (F^2 and F^3) are compactly represented by the associated position vectors $n_i\ (=[x_i, y_i]^T)$ from the output of 2nd level SOM. It should be noted that w_f in the first part of the above function acts in a way similar to the mean operator in the second part because $\sum_{i=1}^{m} w_{f_i} = 1$. The parameter μ is the weight assigned to the features of the node, while $1-\mu$ is the weight assigned to the compressed position features. In this paper, the value of μ was set to 0.8, which was found to deliver satisfactory results in our experiments.

3.3. Training Algorithm of MLSOM

The training sequence is as follows. First, the bottom SOM layer is trained by all the bottom-level nodes. When the 2nd level nodes are processed, position vectors (as required to describe their child nodes at 3rd level) are already prepared to be used in Eq. (7). Then, the second SOM layer is trained by all the 2nd-level nodes of the trees with the appended input vectors. Finally, the top SOM layer is trained by using all the root nodes with the appended input vectors. These are illustrated in Figure 2. The training algorithm for MLSOM can be summarized as follows:

```
Initialize SOM for all layers
Loop from the third to first layer
    Select all the nodes of all images at the same level as the SOM layer
    Create the input vector for each node
    Loop for training SOM
        Randomly select an input vector
        Find a winner neuron for the input vector on the current SOM layer
        Update the current SOM layer
    End
    Find the final winner neuron at the current SOM layer for each node at the
    current level and append the winner neuron’s position vector to the input
    vector of its parent node.
End
```

The basic steps involved in the above training procedure are elaborated below:

Step 1) Initialization: First, the three SOM layers are initialized. The feature dimension for nodes at 1st, 2nd, and 3rd layer are 48, 10 and 10 respectively. Therefore, the dimension of the weight vectors at 1st, 2nd, and 3rd layer of SOM are $(48+2\times q_1)$, $(10+2\times q_2)$ and 10 respectively, where q_1 and q_2 are the maximum numbers of child nodes a node can have at 1st and 2nd level. All the weight vectors are randomly initialized and the values in each dimension are in a range between 0 and 1. Also, normalize the 2-D positions of neurons in each SOM layer. Set current level k to 3 and set iteration t to 0.

Step 2) Input preparation: Collect all the nodes in the current level k. Before serving a node to a SOM layer, the input vector of a node X^k at the *k*th (*k*<3) level is made up by combining a feature vector with position vectors. A position vector $n_i = [x_i, y_i]^T$ contains the spatial position of the winner neuron on the *(k+1)*th SOM layer for node X^k’s *i*th child node. The position vectors $\{n_1, n_2, ..., n_c\}$ are also sorted before they are appended to the input vector of node X^k. The sorting technique is described at the end of this section.

Step 3) Input matching: Randomly select an input X^k from the current level k. Find the winner neuron on the kth SOM layer by using the following maximum similarity criterion:

$$c = \arg \max_i S\left(X^k,\ W_i^k\right),\ i = 1, 2, ..., M\ , \tag{8}$$

where *c* is the index number of the winner neuron at the *k*th SOM layer, W_i^k is the *i*th weight vector at the *k*th layer SOM and *M* is the size of the *k*th SOM layer and *S* is the similarity measure defined in (7).

Step 4) Updating: After the winner neuron *c* is found for the input node X^k at the *k*th level, the weight vectors at the *k*th SOM layer are updated as follows:

$$W_i^k(t+1) = W_i^k(t) + \eta(t)h_{ic}(t)[X^k(t) - W_i^k(t)],\ i = 1, 2, ..., M\ , \tag{9}$$

where t is the iteration number, $\eta(t)$ is the learning rate parameter and $h_{ic}(t)$ is the neighborhood function centered on winning neuron *c*. Both $\eta(t)$ and $h_{ic}(t)$ decrease gradually during the learning process. The neighborhood function $h_{ic}(t)$ can be defined by

$$h_{ic}(t) = \exp\left(-d(n_i, n_c)^2 / 2\sigma^2(t)\right), \tag{10}$$

where $\sigma(t)$ is the width of the neighborhood function, $d(n_1, n_2)$ is the Euclidean distance between two neurons with position vectors n_1 and n_2. $\sigma(t)$ decreases with time and can be the following form:

$$\sigma(t) = \sigma_0.\exp\left(-t/\tau_1\right), \tag{11}$$

where σ_0 is the initial width, $\tau_1 = \left(-\tau_2/\log\sigma_0\right)$, τ_2 is the maximum number of iterations.

The learning rate $\eta(t)$ can be defined by:

$$\eta(t) = \eta_0.\exp\left(-t/\tau_2\right), \tag{12}$$

where η_0 is the initial learning rate.

Step 5) If maximum iteration is reached go to step 6. Otherwise increase iteration t by 1 and go to step 3.

Step 6) Set current level by $k = k - 1$, set iteration t to 0 and go to step 2, unless current k is 1 for which training will stop.

Sorting of position vectors for input preparation of level-1 and level-2 is performed by using a 1-D SOM [29, 30]. For example, the maximum number of child nodes at level-3 (for any parent node in level-2) is 6. So we use a 1-D SOM consisting of 6 neurons, and train it with the position vectors of all child nodes at level-3 from training dataset. After the training is complete, the neurons act like 6 major centers of clusters that are also spatially ordered. Now consider sorting of three child nodes before appending their position vectors (n_A, n_B, and n_C) to their parent node. As illustrated in Figure 3, these three position vectors are compared to the trained neurons, and found that 3rd, 2nd, and 6th neurons are the closest match respectively. Thus, an sorted array of position vectors [**0**, n_B, n_A, **0**, **0**, n_C] is formed, where **0** is a zero vector [0,0]. The purpose of this sorting lies in the similarity matching between two position sets of children nodes that appears in the second part of Eq. 7. By performing sorting, the two position sets can be compared by a meaningful and fast one-to-one matching, instead of many-to-many matching [12].

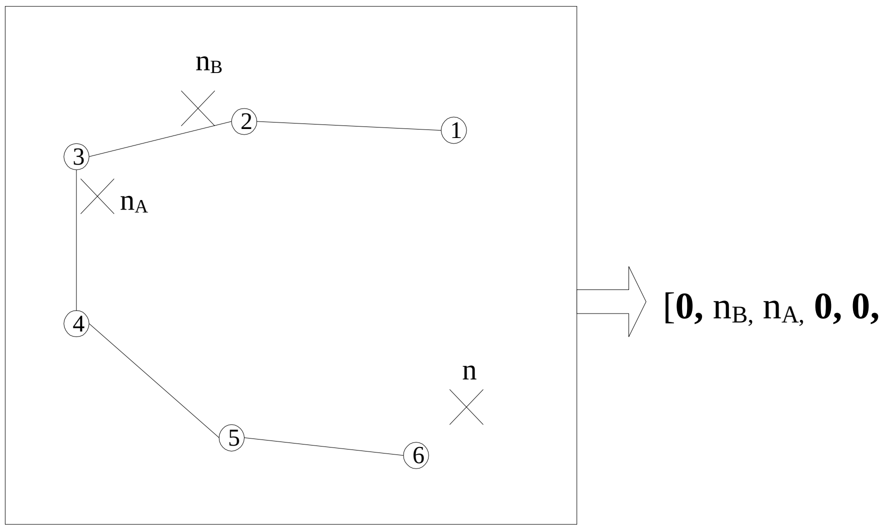


Figure 3. Illustration of the Ordering of Three Child Nodes A, B and C

3.4. Visualization and Retrieval of Flower Images

After all the SOM layers have been trained, each training image can be associated with a neuron on the top SOM layer and thus can be used for visualization. The best-matched neuron is found by processing nodes of its tree structure in a bottom-up fashion using the following procedures:

Procedure for finding the best-matched neuron for an image data:

```
Loop from the third to first layer
        1. Select all the nodes of all images at the same level as the SOM layer
        2. Create the input vector for each node
        3. Find the final winner neuron at the current SOM layer for each node at the
           current level and append the winner neuron's position vector to the input
           vector of its parent node.
End
```

Thus each flower image is associated with a neuron at the top SOM layer. The SOM map can be visualized by showing flower species with associated neurons. Some useful information, such as clustering tendency, can be further detected from the output maps as shown in Figure 6. After the image association with neurons, the image retrieval task can be done by the following procedures:

Procedure for image retrieval for a query image:

1. Extract the tree-structured features for the query image.

2. Process the nodes of the tree structure in a bottom-up fashion.

3. Sort the neurons of the top SOM layer in descending order according to similarity with the root node of the query tree-data.

4. Using those sorted neurons, stack names of the associated images in descending order until *N* number of images are found. *N* is the number of images to be retrieved defined by users. Return the *N* images to users.

4. Experimental Results

To evaluate the proposed method for flower image retrieval, a flower data set with 36 flower species was used in this experiment. 40 sample images were used for each flower species. The pictures are taken using both single and multi flower views, with a variety of natural backgrounds including leaves, stem, soil *etc*. Typical images of each flower species are shown at Figure 4. The database was divided into two parts, namely training and testing sets. The training set, consists of 1260 images, was used to train the SOM. The rest 180 images, constituting the testing set, were used to test the generalization of SOM-based retrieval performance. Images were retrieved only from training set. All the images were resized into 107×80 before processing. The performance of the proposed method was compared with that of SOM-SD [23] with the tree-structured features. We also included comparative results from using SOM with flat features that are combination of color histogram [6] and Gabor texture [4]. The color histogram feature in this case is the same as the one used in the tree-structured features. However, in this study, the Gabor texture features appeared to be more effective as global features compared with the one described in Section 2.3 used for region feature. For Gabor texture features, 4 scales and 6 orientations were used. Means and standard deviations of the output from Gabor filters were used. Thus, the total flat features used for SOM are 96 (16×3+4×6×2). The values of the flat features, tree-structured features and the weight vectors in each SOM layer were normalized. The position vectors were also normalized in a way that the values of positions in each dimension lie within [0,1]. The procedures of image association and retrieval described in Section 3.4 were used to retrieve flower images for all the three SOM methods. For all the three SOM methods, 10 iterations were used for training one-layer SOM. In this study, one iteration means that all the training images are used once for MLSOM, SOM and SOM-SD. The initial learning rate was set to 0.5 and the initial radius of the neighborhood function was set to half of the lattice dimension. The performance of the proposed method was also compared with IRM method [9] that has been successfully used for general image retrieval [10]. To implement the IRM method [9], we have used the features described in Section-2.3: color moments (means), texture moments (means) and shape features.

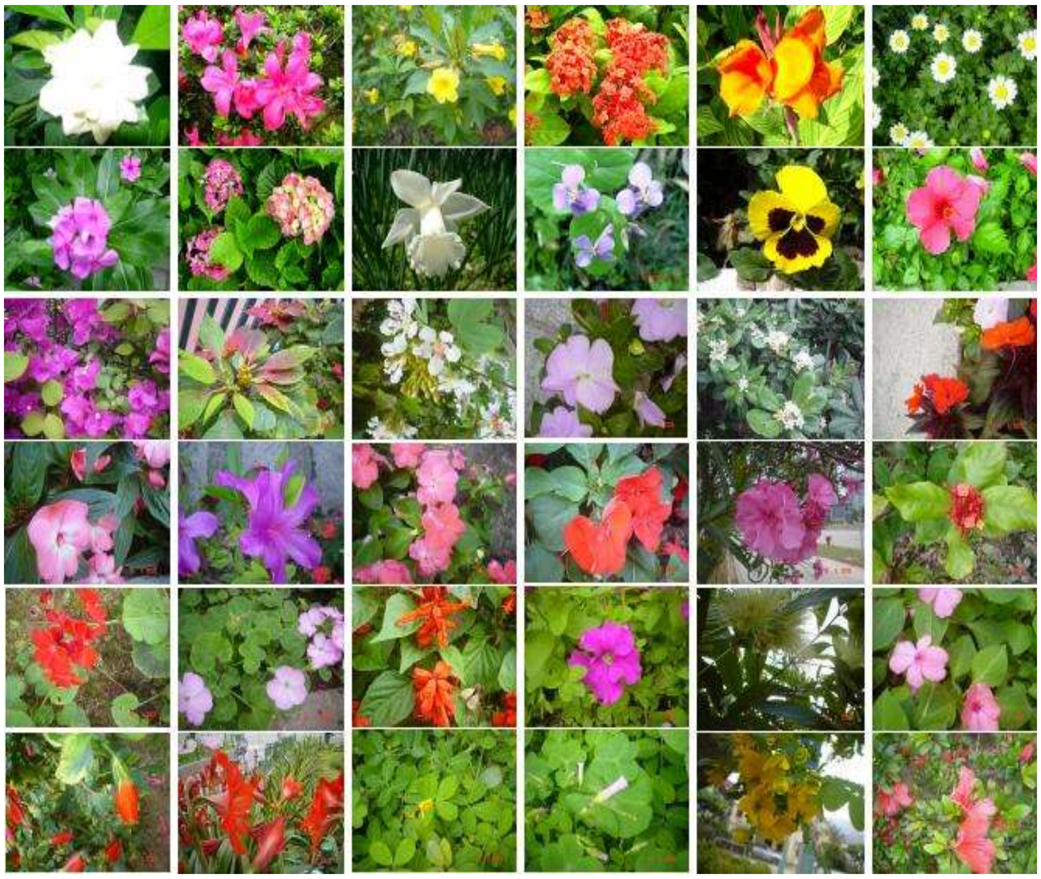


Figure 4. Samples of Flower Images from Different Species

We have investigated different SOM sizes for training SOM, SOM-SD and MLSOM. Figure 5 shows the retrieval performance of the three methods evaluated under different SOM sizes. The retrieval performance is defined by the precision of image retrieval as follows:

$$precision = \frac{\text{Number of relevant images retrieved}}{\text{Total number of images retrieved}}.$$

The retrieval precision in Figure 5 was averaged over all data for retrieving 20 images. The obtained result shows that the proposed method delivers the best performance under different SOM sizes compared with other two methods. The results also show that for all the three SOM methods, the performance was improved when the number of neurons was increased from a small number. It is because a small number of neurons can cause more neurons to be associated with different flower species. Figure 5 also shows that SOM-SD requires relatively more neurons to obtain satisfactory results because of the architecture of SOM-SD. But more neurons mean more computational burden. From the results shown in Figure 5, it appears that a 20×20 SOM size is reasonable for the proposed method in each SOM layer and the traditional SOM. However, a relatively large SOM size of 65×65 was used for SOM-SD [24] because the number of nodes in an image tree is large.

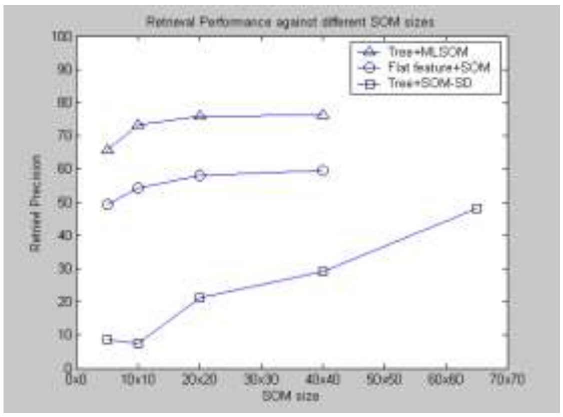
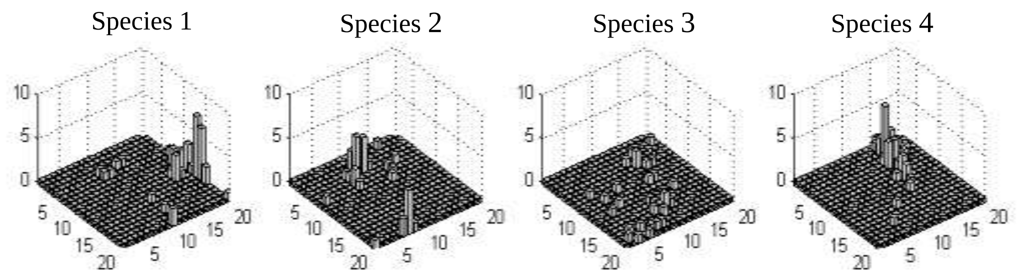
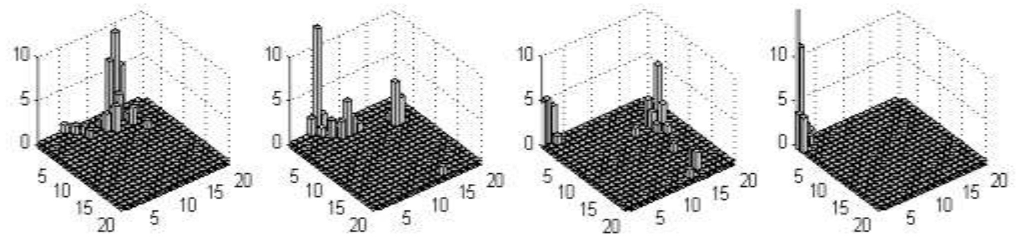


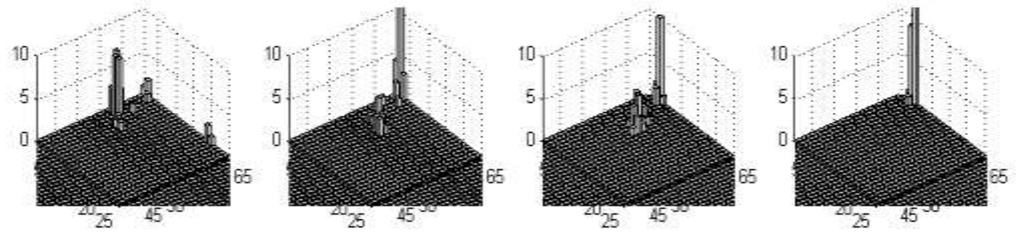
Figure 5. The Average Flower Retrieval Precision by the Three Methods Against Different SOM Sizes



(a) SOM with Flat Image Features



(b) The Proposed Method with Tree-Structured Features

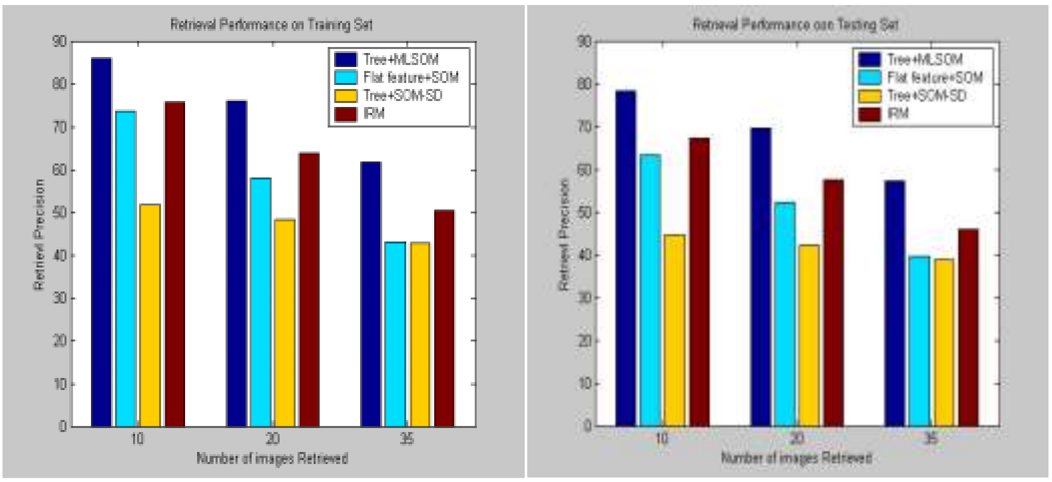


(c) SOM-SD with Tree-Structured Features

Figure 6. The Visualization of Four Flower Species on the SOM Map from the Three Approaches. Each Figure Shows Distribution of Flower Images of a Specific Species on the SOM Map, where the Bars Represent the Number of Images Associated with a Neuron. Flower Images from a Species are Comparatively more Scattered on the SOM Map that used Flat Image Features. On the other Hand Different Species are Comparatively much Overlapped on the Map of SOM-SD

After the completion of training SOM, SOM-SD and MLSOM, the visualization of four different flower species on the output map is shown in Figure 6. It should also be noted for MLSOM that only the top SOM layer is used and shown for visualization. The clustering tendency of different species can be easily visualized from these maps. In general, better visualization means more solidity of the cluster for a particular data class as well as better separation of clusters of different classes. In this aspect, flower species are better grouped by the proposed method compared with others as shown in Figure 6. In the case of SOM with flat image features, images from a flower species are comparatively more scattered, and different flower species are mixed up to some extent. It is probably because the global representation of flower images using the flat features cannot reflect the local objects satisfactorily for better separation of classes. However, using the tree-structured data, SOM-SD provides worse visualization than the proposed method because different flower species are mixed up on the output map by sharing same neurons. The visualization is similar to that reported in [24], because the same SOM layer in SOM-SD processes all the nodes of the 3-level trees, and most neurons are representative for non-root nodes. Only few neurons, being associated with root nodes, are used for visualization of flower images. Thus, clustering of SOM-SD is strong in the sense that root nodes are clearly isolated from the child nodes at level-3 and level-2. However, for this application different classes need to form clearly separated clusters, and clustering of SOM-SD is very poor in that sense.

Finally, image retrieval results using IRM and three SOM methods (using 20×20 size for SOM and MLSOM, and 65×65 size for SOM-SD) are summarized in Figure 7. For a given query image, images are retrieved from the training set containing 35 images per species. Figure 7 shows the average retrieval precision for retrieving the first 10, 20 and 35 images, calculated by averaging over all query images. Retrieval performance on the training and the testing sets are shown in Figure 7a and Figure 7b respectively. It should be noted that images of the testing set are not used in the training phase of SOM methods. Similar results from both training and testing data sets reflect the generalization of the SOM methods. Considering the results in Figure 7, MLSOM with tree-structured features performs better than SOM with flat features and SOM-SD with tree-structured features. The better result obtained by the proposed method is clearly related to its better clustering results of flower species on the output map shown in Figure 6. The proposed approach also shows better performance compared with IRM method, a purely region based image comparison method. Thus we can conclude that the proposed approach can deliver superior results by integrating both global and region-based properties of a flower image.



(a) Retrieval Precision on Training Set (b) Retrieval Precision on Testing Set

Figure 7. The Average Retrieval Precision over 36 Flower Species by the Four Methods, when the Total Numbers of Retrieved Images are 10, 20 and 35 Respectively

We have not implemented a direct method for tree-structured data that might provide even better results in exchange of an exhaustive computational demand. It should be noted that in our proposed method the purpose of SOM is not to improve the retrieval accuracy. We have used the SOM technique as a clustering tool to speed-up the retrieval task and it reduces the retrieval precision as a trade-off. Comparing first two SOM-based methods, it is obvious that tree-structured feature is more effective than traditional flat features, specially when considering clustering technique like SOM. Considering, the two tree-based methods, our proposed MLSOM based technique outperforms the SOM-SD based technique. Overall, our proposed method, combining both the tree-structured data and MLSOM, is more effective in handling such application of data clustering, visualization and retrieval.

Table 2. The Computational Time for Different Methods

Methods	Retrieval time (per query)
SOM (20×20) with flat image features	0.015 seconds
MLSOM (20×20 for each layer) with tree-structure	**0.016** seconds
SOM-SD (65×65) with tree structure	0.097 seconds
IRM	7.620 seconds

The computational time for the task of image retrieval depends on the code implementation and the programming language. However, for a brief comparison, the computational times for the three methods, which use 20×20 size for SOM and MLSOM, and both 20×20 and 65×65 size for SOM-SD, are shown in Table 2. All the times were calculated by using MATLAB 7.6 in an Intel core-2 2GHz PC with 2GB memory. The 20×20 MLSOM needs less computational time in training compared to the 20×20 SOM-SD with the same tree-structured data. This is mainly due to the fact that the generated input vector of the second and third level nodes for SOM-SD is longer than that for MLSOM. Clearly SOM-SD needs more computational time when a 65×65 size is used to achieve reasonable retrieval results. It is noticed that the retrieval time for a query is relatively short for all the three SOM methods, whilst a much longer retrieval time is usually required for the IRM method. This is due to the algorithmic procedures of the IRM method, which needs to compare the query image with every image of the database. The retrieval time can be a critical issue when database size is huge, and a clustering technique is then necessary to maintain an acceptable speed. The Table 2 clearly indicates that the retrieval time can be significantly reduced when clustering technique like SOM is used.

In real scenario, the images of a same scene can be different because of the camera settings and environmental conditions. Thus, performance evaluation was also conducted under different kinds of image alterations using the software “FotoCanvas, version 1.1, ACD system”. To evaluate the retrieval performance under these image alterations, one image sample from each species were randomly selected. The average retrieval precision on each alteration was then calculated for the 20 retrieved images. These altered images were not used to train the SOM again. Figure 8 shows the performance of the flower retrieval using MLSOM against blurring, brightening, contrast increase and darkening. These results corroborate that the proposed method is reasonably robust against image alterations except the case of brightness variation. It is also verified that proposed method is also robust against image rotation, which is obvious due to the fact that the features are not dependent on the orientation information.

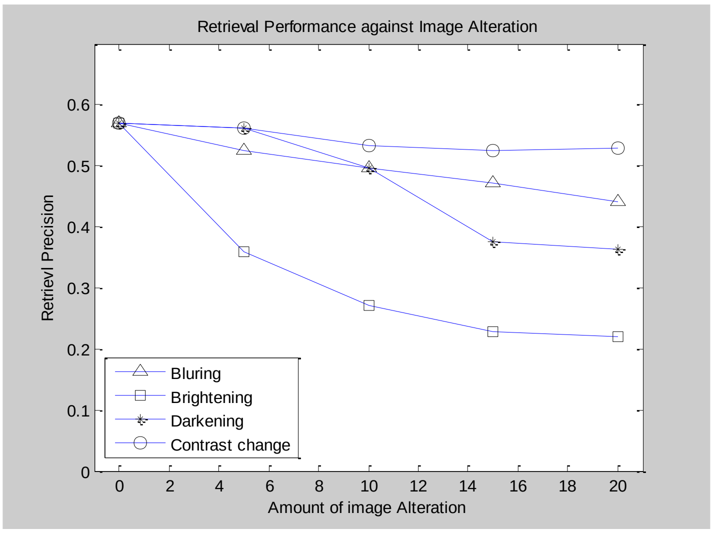


Figure 8. Retrieval Performance under Different Image Modifications: Image Blurring, Brightening, Darkening and Contrast Change

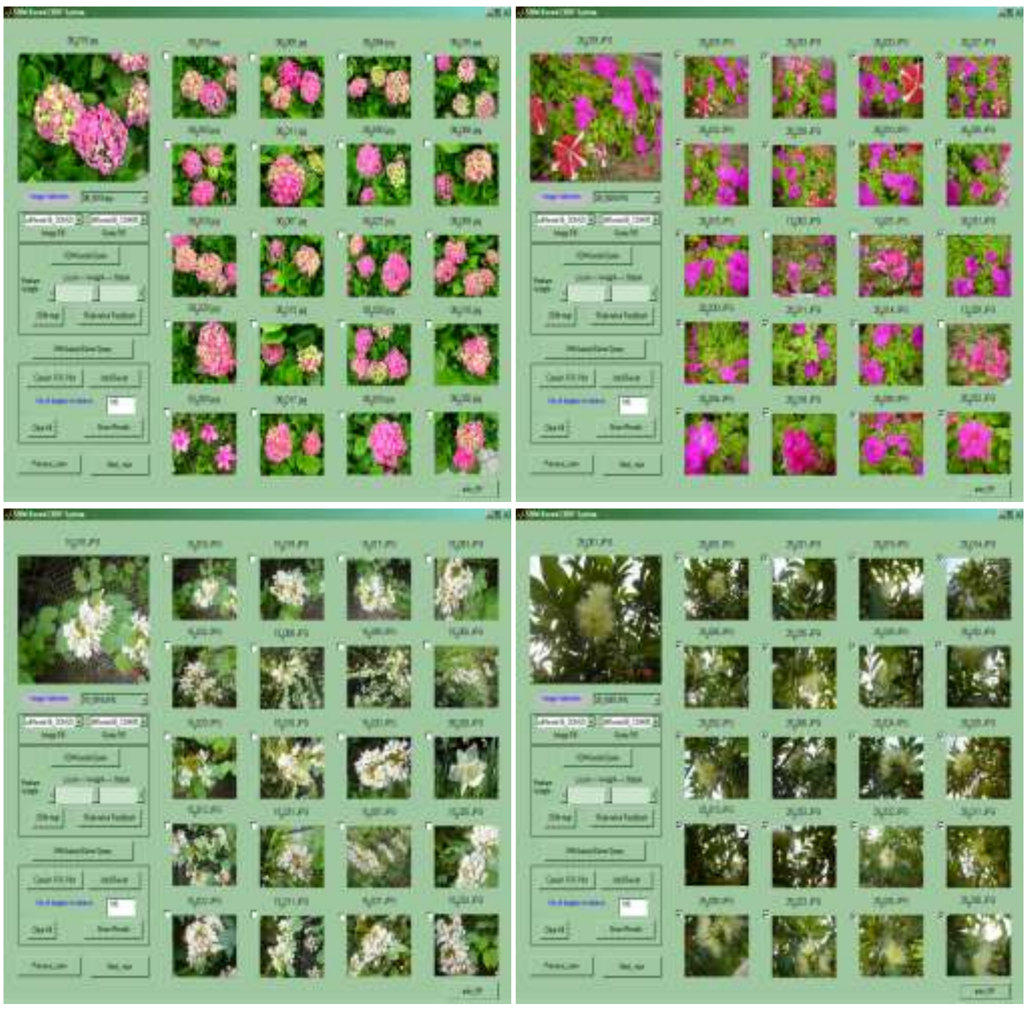


Figure 9. Four Retrieval Examples are shown from Different Species. The Query Image is shown at Left-Top Corner, and First 20 Images of Retrieved Images are Shown

Figure 9 shows typical retrieval examples using the proposed approach. Four sample queries are presented in Figure 9 including multi-color flowers and different background. Variation of flower color within the same species affects the retrieval performance, although the texture features of leaves are mostly unique for the same species. In the case of scale variations, which were obtained by using close-up and long shots, the proposed tree-structured feature exhibits advantage over the traditional global color or texture feature. For example, in the case of global flat feature, less important regions like background can become dominating in image similarity measure because of their relatively large size. In contrast, all regions are almost equally weighted in the similarity measure when region-based tree-structure feature is used. Figure 10 shows two cases of scale variation in flower image through a precision-recall characteristic. Precision-recall plot is a common way to visualize retrieval performance in each individual case. The recall is defined as follows

$$recall = \frac{\text{Number of relevant images retrieved}}{Number\ \ of\ \ total\ \ image\ \ in\ \ same\ \ class}\,.$$

Ideally, the precision can always be kept at 1 even when the recall increases. In real applications, the precision drops when more images are retrieved or the recall increases. Figure 10 shows precision-recall characteristic for retrieving unto 35 images. Two case studies of scale variations in flower images are shown in Figure 10: (a) close shot query image, where most images in that species are long shot; (b) long shot query image, where most images in that species are close shot. It is quite common in real world to have such scale variation between query and most of database images in the same category. The results shown in Figure 10 indicate that the proposed tree-structured feature is able to deliver higher retrieval precision compared with flat global feature.

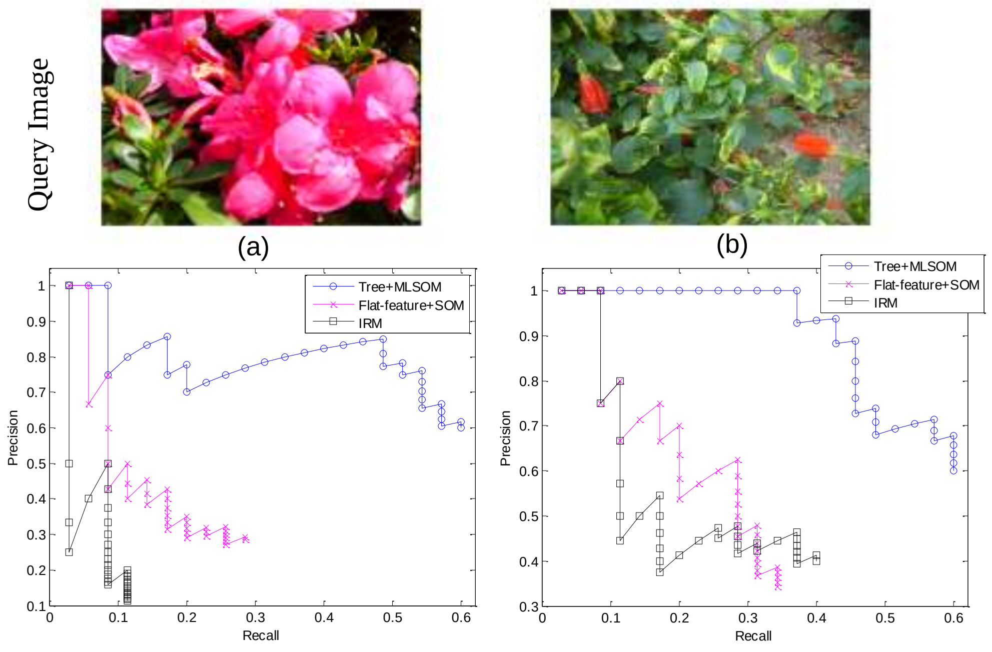


Figure 10. Two Case Study of Scale Variation in Flower Image through Precision-Recall Characteristic for Retrieving unto 35 Images. (a) Close Shot Query Image, where most Images in that Species are Long Shot; (b) Long Shot Query Image, where most Images in that Species are Close Shot

5. Conclusion

In this paper, a MLSOM based method for flower image retrieval is proposed by using a new tree-structured image representation. A 3-level tree is used to organize the global and the local features in a hierarchical fashion. This type of representation enables object-oriented image features to be better encoded. As a result, better retrieval performance can be achieved compared with that of flat image features. The tree-structured data of flower images is then processed by MLSOM. Once the training is complete for MLSOM, flower images are organized on the output map of MLSOM. As a result, flower image retrieval can be performed in a much efficient way compared with direct method of image retrieval. The performance of image retrieval is related to the visualization of a SOM map. The comparative experimental results show that the proposed method delivers better data visualization of different flower species than SOM with flat image features and SOM-SD with tree-structured features. The proposed method consistently delivers higher retrieval accuracy compared with other SOM approaches and region-based IRM method. Furthermore, the image retrieval performance is robust under different kinds of image alterations. The proposed method is effective and promising for flower image retrieval.

References

[1] M. Das, R. Manmatha and E. M. Riseman, “Indexing Flower Patent Images Using Domain Knowledge”, IEEE Intelligent Systems, vol. 14, no. 5, **(1999)**, pp. 24-33.

[2] G. W. A. M. Heijden and A. M. Vossepoel, “A Landmark-Based Approach of Shape Dissimilarity”, In Proc. of ICPR’ 96, **(1996)**, pp. 120-124.

[3] T. Saitoh and T. Kaneko, “Automatic Recognition of Wild Flowers”, In Proc. of 15th International Conference on Pattern Recognition, vol. 2, **(2000)**, pp. 507-510.

[4] J. G. Daugman, “Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression”, IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 36, no. 7, **(1988)**, pp. 1169-1179.

[5] V. Ogle and M. Stonebraker, “Chabot: Retrieval from a relational database of images”, IEEE Computer, vol. 28, no. 9, **(1995)**, pp. 40-48.

[6] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, “The QBIC project: querying images by content using color, texture, and shape”, In Proceeding of SPIE - Int. Soc. Opt. Eng., in Storage and Retrieval for Image and Video Database, vol. 1908, **(1993)**, pp. 173-87.

[7] E. Saber and A. M. Tekalp, “Integration of color, edge, shape, and texture features for automatic region-based image annotation and retrieval”, Journal of Electronic Imaging, vol. 7, no. 3, **(1998)**, pp. 684-700.

[8] G. Pass and R. Zabih, “Histogram refinement for content-based image Retrieval”, In Proc. of the IEEE Workshop on Applications of Computer Vision, Sarasota, Florida, **(1996)**.

[9] J. Li, J. Z. Wang and G. Wiederhold, “IRM: Integrated region matching for image retrieval”, In Proc. of ACM Multimedia, **(2000)**.

[10] J. Z. Wang, J. Li and G. Wiederhold, “SIMPLIcity- Semantics sensitive Integrated Matching for Picture Libraries”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, **(2001)**, pp. 947-963.

[11] M. K. M. Rahman and T. W. Chow, “Image retrieval using an improved similarity measure: SRIC Similarity with Region Importance and Consistency”, In Proc. of 17th International Conference on Computer and Information Technology (ICCIT), **(2014)**, pp. 336-343.

[12] T. W. S. Chow, M. K. M. Rahman and S. Wu, “Content-based image retrieval by using tree-structured features and multi-layer self-organizing map”, Pattern Analysis & Applications, vol. 9, no. 1, **(2006)**, pp. 1-20.

[13] M. K. M. Rahman, W. P. Yang, T. W. S. Chow and S. Wu, “A Flexible Multi-Layer Self-Organizing Map for Generic Processing of Tree-Structured Data”, Pattern Recognition, vol. 40, no. 5, **(2007)**, pp. 1406-1424.

[14] T. W. S. Chow and M. K. M. Rahman, “A New Image Classification Technique using Tree-Structured Regional Features”, Neurocomputing, vol. 70, no. 4-6, **(2007)**, pp. 1040-1050.

[15] A. Sanfeliu, R. Alquézar, J. Andrade, J. Climent, F. Serratosa and J. Vergés, “Graph-based representations and techniques for image processing and image analysis”, Pattern Recognition, vol. 35, **(2002)**, pp. 639-650.

[16] H. Radha, M. Vetterli and R. Leonardi, “Image compression using binary space partitioning trees”, IEEE Transactions on Image Processing, vol. 5, no. 12, **(1996)**, pp. 1610-24.

[17] P. Salembier and L. Garrido, “Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval”, IEEE Transactions on Image Processing, vol. 9, no. 4, **(2000)**, pp. 561-76.
[18] S. Y. Cho and Z. Chi, “Genetic evolution processing of data structures for image classification”, IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 2, **(2005)**, pp. 216-231.
[19] A. R. Weeks and G. E. Hague, “Color segmentation in the HIS color space using the K-means algorithm”, In Proc. of SPIE, Nonlinear Image Processing VIII, vol. 3026, **(1997)**, pp. 143-154.
[20] R. M. Haralick and L. G. Shapiro, “Image segmentation techniques”, Computer Vision, Graphics & Image Processing, vol. 29, **(1985)**, pp. 100-132.
[21] N. R. Pal and S. K. Pal, “A review on image segmentation techniques”, Pattern Recognition, vol. 26, no. 9, **(1993)**, pp. 1277-1294.
[22] Daubechies, Ten lectures on wavelets, Capital City Press, **(1992)**.
[23] M. Hagenbuchner, A. Sperduti and A. C. Tsoi, “A Self-Organizing Map for Adaptive Processing of Structured Data”, IEEE Trans. Neural Networks, vol. 14, **(2003)**, pp. 491-505.
[24] M. Hagenbuchner, “Extension and evaluation of adaptive processing of structured information using artificial neural networks”, PhD Dissertation, Faculty of Informatics, University of Wollongong, **(2002)**.
[25] T. W. S. Chow and M. K. M. Rahman, “Multi-Layer SOM with Tree Structured Data for Efficient Document Retrieval and Plagiarism Detection”, IEEE Transactions on Neural Networks, vol. 20, no. 9, **(2009)**, pp. 1385-1402.