Annales Univ. Sci. Budapest., Sect. Comp. 53 (2022) 29–55

AN IMPROVEMENT OF R-TREE FOR CONTENT-BASED IMAGE RETRIEVAL

Le Thi Vinh Thanh (Ba Ria-Vung Tau, VietNam)
Thanh The Van (Ho Chi Minh city, VietNam)
Thanh Manh Le (Hue, VietNam)

Communicated by Attila Kiss

(Received August 16, 2021; accepted January 20, 2022)

Abstract. The problem of image retrieval from a large image dataset has been having many challenges. Therefore, indexing and searching images on a data structure are important requirements to improve retrieval effectiveness. In this paper, an improved structure based on R-Tree, named R^S-Tree (Region Sphere Tree), is proposed to enhance the accuracy of the content-based image retrieval on different image sets. The improvements on R^S-Tree include: (1) representing image feature vectors in the form of spheres to optimize storage space; (2) improving operations on R^S-Tree such as adding, deleting, and splitting nodes to enhance retrieval efficiency. The result of this processes creates the balanced clustering tree. Since then, content-based image retrieval model is built rely on R^S-Tree. On the base of proposed theory, the experiment is performed on data-sets including COREL, Wang, Oxford Flowers-17 with precision values of 76.29%, 73.16%, and 78.69%, respectively. The experimental results are compared to related works on the same dataset to demonstrate the effectiveness of the image retrieval model based on R^S-Tree.

1. Introduction

The problem of the retrieving relevant images from a large image dataset is a challenging research problem. Many models of content-based image retrieval

Key words and phrases: R^S-Tree, CBIR, image retrieval, similar images, clustering.
The ACM Computing Classification (1998): H.2.8, H.3.3.

(CBIR) is applied in the image retrieval systems [1, 2, 3]. Many different methods have been proposed to enhance retrieval effectiveness. There are two main aspects to perform image retrieval included (1) extracting the visual content of the image; (2) describing the visual content in the form of indexes [4, 5, 6]. Currently, there are many the indexing methods for multidimensional data such as KD-Tree, Quad-Tree, M-Tree, R-Tree, etc. [6, 7, 8]; with R-Tree is the popular structure for storing indexes based on data partition [10]. R-Tree is a balanced multi-branched tree and the data elements is stored in the leaves of the tree. The data is partitioned into blocks that can be nested or overlapped (introduced by Guttman in 1984) [9]. Different variants of R-Tree include: R*-Tree, SS-Tree, SR-Tree, X-Tree, M-Tree, etc. are effectively applied in image retrieval systems [10, 11, 12].

The R^S-Tree is an improved structure based on R-Tree to enhance image retrieval performance. In R^S-Tree, the feature vector of image is represented in the form of spheres and stored in the leaves of tree. To enhance the effectiveness of storage and retrieval on the R^S-Tree, a threshold θ $(0 < \theta < 1)$ is proposed. In addition, the splitting algorithm based on a measure is proposed to improve retrieval efficiency. The result of this process is the creation of a balanced clustering tree and clustering the data elements in the leaves to enhance the accuracy of the retrieval system.

The main contributions of the paper included (1) improving the principle of the building R^S-Tree based on the threshold θ to cluster similar data; (2) improving the algorithm of node splitting based on the different measures of elements to enhance the accuracy of the retrieval system; (3) building a clustering model named R^S-Tree rely on R-Tree to store similar images; (4) proposing the content-based image retrieval model based on R^S-Tree; (5) building the experiment and evaluating the results of the proposed method using three image sets including COREL, Wang, Oxford Flowers-17.

The rest of this paper is organized as follows. In section 2, we survey and analyze related works. In section 3, we propose the clustering structure based on R-Tree, named R^S-Tree. In section 4, we present the image retrieval model based on the R^S-Tree. In section 5, we build experimental applications and evaluates the results based on the proposed theory. Conclusion and future works are presents in section 6.

2. Related works

In recent years, many researchers have proposed different methods to enhance efficiency for the image retrieval systems with specific as follows:

Milind V. Lande et al. (2014) proposed the method of extracting color, texture, and shape features from an image dataset. Then, the authors combined

these features to ensures higher retrieval efficiency. For extraction of color features, images were divided into non-overlapping blocks, and the dominant color of each block is determined using the k-means algorithm. To extract texture features, the authors used a gray-level co-occurrence matrix (GLCM). The work's experiment was performed on the Wang dataset. The experimental results show that the proposed method was effective. However, the results also were experimented with Matlab. The authors have not proposed a data structure to improve retrieval performance yet [13].

Haldurai et al. (2015) proposed the content-based image retrieval system on R-Tree structure. This work used image features such as color and texture for image retrieval. These features were extracted using fuzzy approaches and stored in the R-Tree structure. An image retrieval process is performed on the tree to enhance the retrieved performance. Experimental results on the COREL image set show that the proposed method is effective [14]. Abd Aziz et al. (2018) proposed the method to reduce the dimensionality of data by using S-Map. The image retrieved process was performed based on R-Tree. The experimental results on image and video datasets show that the proposed method improves retrieval performance [15].

Vanitha et al. (2017) proposed an SR-Tree index structure to apply for a content-based image retrieval system. In this study, features such as color, spatiality were extracted and stored on SR-Tree. The experimental results on the COREL image set show that SR-Tree works better than other indexing structures [16]. However, in SR-Tree the inserting algorithm needs to update both spheres and rectangles. Thus, creating and updating are complicated and expensive costs. Besides, since the SR-Tree contains both spheres and rectangles, its size is larger.

Shama, P. S. et al. (2015) proposed an image retrieval system on the R*-Tree for plant images. The authors use the co-occurrence matrix method and Gabor filtering to extract image features. The experiment was performed on 300 vegetation images [17]. Alfarrarjeh et al. (2020) proposed an image retrieval system with street image data [18]. The experimental results of these works show that the image retrieval system based on the R*-Tree is effective. However, in the R*-Tree, the re-insert algorithm when encountering an overflow node, leads to reorganizing the tree, and increasing tree building costs. In addition, the authors only performed on a small and specialized image dataset.

Payal Chhabra et al. (2020) presented a content-based image retrieval system based on low-level features of the image. In this study, the authors used the Oriented Fast and Rotated BRIEF (ORB) and the Scale-Invariant Feature Transform (SIFT) method to extract image features, etc. The experimental results show that the proposed method is reliable [20]. However, the authors did not suggest a data structure to store images to increase retrieval performance.

Related works show that the method of image retrieval based on the indexing structure is very effective. On the basis of inheriting and overcoming limitations of related works, we propose the improved structure R^S-Tree based on R-Tree to be used for content-based image retrieval systems. The R^S-Tree is built based on partitional clustering and hierarchical clustering method, each leaf stores similar data elements, thus improving image retrieval efficiency. From then, the content-based image retrieval model was built based on the R^S-Tree. The experiment was performed experimented on image set Wang (including 10,800 images, 80 classes), COREL (1000 images, 10 classes), and Oxford Flower-17 (1360 images, 17 classes) to demonstrate the effectiveness of the image retrieval based on the R^S-Tree.

3. Cluster structure of spatial data R^S-tree

3.1. R-Tree structure

In the original R-Tree structure [9], each internal node is of the form $\langle MBR, p\rangle$ that Minimum Bounding Rectangle(MBR) is the minimum rectangular spatial region contained sub-spatial regions inside and p is the link point to child nodes. Each leaf is of the form $\langle MBR, oid\rangle$ that MBR the minimum rectangular spatial regions contained the data objects and oid is the object identifier. Each leaf in the tree which has a maximum number of elements is M and a minimum number of elements is m. Each leaf is a data cluster containing objects in multi-dimensional space.

Deleting an element from R-Tree can be rebuilt the tree. When a leaf has m elements, removing an element from the leaf lead to the once is not existence; therefore, the remaining elements must be redistributed on the tree. Besides, when an element is added to R-Tree, the closest leaf is be selected by similarity measure. In the worst case, the element is completely different from the other elements in that leaf at the same time the number of elements in this leaf is less than M, then the leaf is not split. Therefore, this leaf is not well-partition, so the accuracy is reduced in searching problem.

The original R-Tree structure has two important drawbacks: (1) The execution of a point location query in an R-tree may lead to the investigation of several paths from the root to the leaf level. This characteristic may lead to performance deterioration, specifically when the overlap of the $MBRs$ is significant; (2) A few large rectangles may increase the degree of overlap significantly, leading to performance degradation during range query execution, due to empty space [10].

One of the variants of R-Tree is used in the problem of image retrieval as Similarity Search Tree (SS-Tree) structure [10]. SS-Tree is an index structure

for multidimensional data, which is proposed by White and Jain [26]. Each node on SS-Tree is represented in the form of a sphere including center and radius. The center of the sphere is the centroid of the underlying point. In this structure, the insert algorithm determines the most suitable subtree to accommodate the new entry by choosing a subtree whose centroid is the nearest to a new entry. It is an improvement of the R*-Tree and enhances the performance of nearest neighbor queries. However, the operation of re-inserting entries is performed on SS-Tree unless reinsertion has been made at the same node or leaf. This promotes the dynamic reorganization of the tree structure [27]. On the other hand, SS-Tree is only performed by nearest neighbor queries, i.e. it can not be performed the queries of spatial region. Therefore, in the paper, an improved structure named R^S-Tree is built to enhance retrieval precision by combining nearest neighbor and spatial region queries.

3.2. R^S-Tree structure

R^S-Tree is built based on SS-Tree structure. The center vector of the sphere of the leaf and node on R^S-Tree is similar to SS-Tree. The improvements of R^S-Tree including (1) creating a sphere structure to store feature vector of an image; (2) improving the node splitting process based on the difference measure; (3) proposing a theta threshold to cluster similar data; (4) combining nearest neighbor and spatial region queries.

R^S-Tree is a balanced multi-branch tree used for similar image retrieval. The process of data clustering is performed on each node of R^S-Tree based on the Euclidean distance and threshold θ. This process creates a balanced multi-branch clustering tree to enhance the accuracy of the retrieval system and reduce the retrieval time. R^S-Tree is a data partition structure including a root, a set of nodes, and a set of leaves.

The node denoted S_{node} is of the form $\langle MBS, p\rangle$. Where MBS is a sphere that has center denoted $\overrightarrow{c}_{node}$, and radius r_{node}, p is the link to the child nodes. This sphere covers the spheres of nodes in each subbranch of the tree. Each S_{node} has a minimum element of 2 and a maximum of N.

The leaf denoted S_{leaf} is of the form $\langle MBS, element\rangle$. Where MBS is a sphere that has center denoted $\overrightarrow{c}_{leaf}$ and radius r_{leaf} contains a set of elements. Each element $spED$ is of the form $\langle MBS, oid\rangle$. Where MBS is a sphere that has center denoted $\overrightarrow{c}_{sp}$, and radius r_{sp}, contains the object space, oid is identifier $\overrightarrow{f} = (v_1, v_2, ..., v_d)$. Each leaf S_{leaf} has the maximum number of elements M and the minimum $m(1 < m < M/2)$.

An image I with feature vector $\overrightarrow{f_I} = (v_{I1}, v_{I2}, ..., v_{Id})$, each sphere MBS of the element $spED$ illustrated in Figure 1 is the sphere that contains identifier $\overrightarrow{f_I}$ with the center vector $\overrightarrow{c}_{sp}$ and the radius r_{sp} is described as follows:

1) The sphere center of the element:

$$\overrightarrow{c}_{sp} = (c_{I1}, c_{I2}, ..., c_{Id})$$

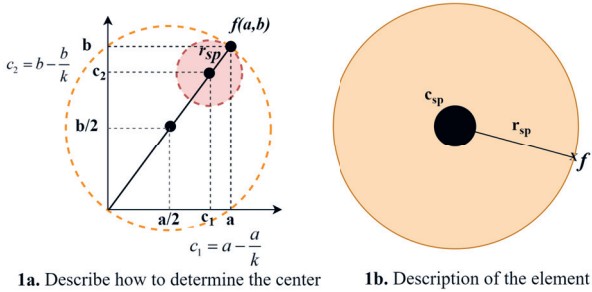
with:

$c_{\mathrm{Ij}} = Max_{\mathrm{j=1..d}}\{0, v_{\mathrm{Ij}} - a_{\mathrm{Ij}}\}$

$a_{\mathrm{I1}} = \frac{v_{I1}}{k}, a_{\mathrm{I2}} = \frac{v_{I2}}{k}, ..., a_{\mathrm{Id}} = \frac{v_{Id}}{k}$, k is a parameter,$k \geq 2$

2) The sphere radius of the element:

$$\mathrm{r}_{sp} = \sqrt{\frac{\sum_{j=1}^{d} (c_{\mathrm{Ij}} - v_{\mathrm{Ij}})^2}{d}} \tag{3.1}$$



1a. Describe how to determine the center **1b.** Description of the element

Figure 1. Description of the spherical structure of the data element

Figure 1a describes how to determine the center $\overrightarrow{c}_{sp} = (c_1, c_2)$ and the radius r_{sp} of the feature vector $f(a, b)$ in 2D space. Where, $c_1 = a - \frac{a}{k}$, $c_2 = b - \frac{b}{k}$ and the radius $r_{sp} = d_E(\overrightarrow{c}, \overrightarrow{f})$, with $k \geq 2$. Figure 1b describes the center $\overrightarrow{c}_{sp}$, the radius r_{sp} of feature vector f.

A sphere MBS of the leaf S_{leaf} illustrated in Figure 2 is the minimal sphere that covers all sphere elements included the center vector $\overrightarrow{c}_{leaf}$ and the radius r_{leaf} described as follows:

1) The sphere center of the leaf S_{leaf}:

$$\overrightarrow{c}_{leaf} = \frac{1}{k} \sum_{i=1}^{k} sp_i.\overrightarrow{c}_i \tag{3.2}$$

Where $sp_1, sp_2, sp_3, ..., sp_k$ are sphere elements inside the S_{leaf} and $sp_i.\overrightarrow{c}$ is the center vector of the sphere sp_i, with $1 < i < k$.

2) The sphere radius of the leaf S_{leaf}:

$$r_{leaf} = Max_{i=1..d}\{d_E(\overrightarrow{c}_{leaf}, sp_i.\overrightarrow{c_i}) + sp_i.r_i\}, \tag{3.3}$$

Where $d_E(\overrightarrow{c}_{leaf}, sp_i.\overrightarrow{c_i})$ is the Euclidean distance from the center vector of the leaf node S_{leaf} to the center vector of the i^{th} sphere element and $sp_i.r_i$ is the radius of the i^{th} sphere element.

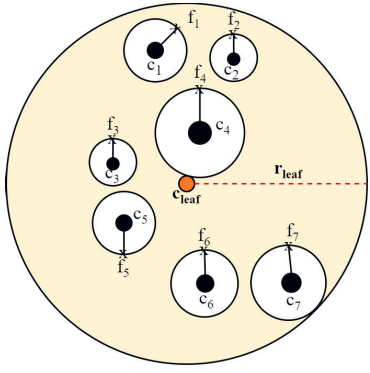


Figure 2. Description of the leaf on R^S-Tree

A sphere MBS of the node S_{node} is illustrated in Figure 3 is the minimal sphere that covers all spheres of the nodes in the subbranch of the tree included a center vector $\overrightarrow{c}_{node} = (c_1, c_2, ..., c_d)$ and a radius r_{node} described as follows:

1) The center sphere of the internal node S_{node}:

$$c_i = \frac{\sum_{j=1}^{k} S_j.\overrightarrow{c}_j.x_i \times S_j.\mathrm{w}}{\sum_{j=1}^{k} S_j.\mathrm{w}}, i = 1..d, \tag{3.4}$$

where $S_1, S_2, ..., S_k$ are the children nodes of the node S_{node}; d is the dimensionality of feature vectors; $\overrightarrow{c}_j.x_i$ is the i^{th} feature of the center vector $\overrightarrow{c}_j$; S_j.w is the number of elements of the node S_j.

2) The radius sphere of the internal node S_{node}:

$$r_{node} = Max_{j=1..k}\{d_E(\overrightarrow{c}_{node}, S_j.\overrightarrow{c_j}) + S_j.r_j\}. \tag{3.5}$$

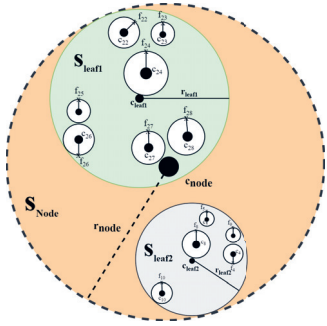


Figure 3. Description of the node on R^S-Tree

Theorem 1. *The sphere S_{leaf} is created from the center $\overrightarrow{c}_{leaf}$ and the radius r_{leaf} is the smallest sphere containing the spheres of the current leaf.*

Proof. i) This sphere containing the spheres of the current leaf. Since

$$r_{leaf} = Max_{\mathrm{i=1..d}}\{d_E(\overrightarrow{c}_{leaf}, sp_i.\overrightarrow{c_i}) + sp_i.r_i\},$$

infer

$$r_{leaf} \geq d_E(\overrightarrow{c}_{leaf}, sp_i.\overrightarrow{c_i}) + sp_i.r_i, \quad \forall \mathrm{i} = 1..\mathrm{d}.$$

Therefore, this sphere contains all spheres of the current leaf.

ii) This sphere is the smallest sphere. Suppose exist a sphere S'_{leaf} has center $\overrightarrow{c}_{leaf}$, and has a radius $r'_{leaf} < r_{leaf}$ containing the spheres of the current leaf. Since $r_{leaf} = Max_{\mathrm{i=1..d}}\{\mathrm{d_E}(\overrightarrow{c}_{\mathrm{leaf}}, \mathrm{sp_i}.\overrightarrow{c_i}) + \mathrm{sp_i}.\mathrm{r_i}\}$, i.e.

$$\exists sp_k \in S_{leaf} \| d_E(\overrightarrow{c}_{leaf}, sp_k.\overrightarrow{c}_k) + sp_k.r_k = r_{leaf},$$

infer sphere S'_{leaf} can not contain the element sp_k. Therefore, the sphere S_{leaf} which has center $\overrightarrow{c}_{leaf}$ with radius r_{leaf} is the smallest sphere. ■

Theorem 2. *Let $\overrightarrow{f_I} = (v_{I1}, ..., v_{Id})$, $\overrightarrow{c}_{sp} = (c_{I1}, ..., c_{Id})$ be two vectors of the data object and the center of the sphere in R^d space, respectively. Then*

(i) *Two vectors $\overrightarrow{f_I}$ and $\overrightarrow{c}_{sp}$ are parallel and have the same direction.*

(ii) *The terminal point of the vector $\overrightarrow{c}_{sp}$ lies on the line pass through initial point and the terminal point of the vector $\overrightarrow{f_I}$ in the R^d space.*

(iii) *The terminal point of the vector $\overrightarrow{f_I}$ lies on the circle with center $c_{sp}(c_{I1}, ..., c_{Id})$ of radius r_{sp} in R^d space.*

Proof. (i) Since vector $\overrightarrow{f_I}$ is normalized in $[0, 1]$. So, $\forall v_{Ii} \in \overrightarrow{f_I}, v_{Ii} \geq 0$, $i = 1..d$; $\forall c_{Ii} \in \overrightarrow{c}_{sp}, c_{Ii} = Max_{i=1..d}\{0, v_{Ii} - a_{Ii}\}$; with $a_{Ii} = \frac{v_{Ii}}{k}, k \geq 2$,

then $Max\{0, v_{Ii} - a_{Ii}\} = Max\{0, v_{Ii} - \frac{v_{Ii}}{k}\} = (v_{Ii} - \frac{v_{Ii}}{k})$. Therefore, $\forall c_{Ii} \in \in \overrightarrow{c}_{sp}, c_{Ii} = (v_{Ii} - \frac{v_{Ii}}{k}) = (1 - \frac{1}{k})v_{Ii}$. Let $t = (1 - \frac{1}{k}), t > 0$ and $t < 1$, infer $\overrightarrow{c}_{sp} = t(v_{I1}, ..., v_{Id}) = t\overrightarrow{f_I}$. Therefore, two vectors $\overrightarrow{f_I}$ and $\overrightarrow{c}_{sp}$ are parallel and have the same direction.

(ii) In R^d space, the vector $\overrightarrow{f_I}$ has a initial point of origin O and an terminal point of $f_I(v_{I1}, ..., v_{Id})$. The vector $\overrightarrow{c}_{sp}$ has the initial point of origin O and the terminal point of $c_I(c_{I1}, ..., v_{Id})$. So two vectors $\overrightarrow{f_I}$, $\overrightarrow{c}_{sp}$ have the same initial point and the same direction, infer the terminal point of the vector $\overrightarrow{c}_{sp}$ lies on the line pass through initial point and the terminal point of the vector $\overrightarrow{f_I}$ in the R^d space.

(iii) In R^d space, take the terminal point of the vector $\overrightarrow{c}_{sp}$ as the center of a circle with radius r_{sp}. In which, r_{sp} is the Euclidean distance between $c_I(c_{I1}, ..., v_{Id})$ and $f_I(v_{I1}, ..., v_{Id})$. Therefore, the terminal point of the vector $\overrightarrow{f_I}$ lies on the circle with center $c_I(c_{I1}, ..., v_{Id})$ of radius r_{sp}. ■

The process of creating spheres for image feature vectors improve the retrieving precision because of two reasons as follows: (1) In the process of building R^S-Tree, a data sphere is added to choose a leaf with the smallest extended spatial region, so the leaf contains the most similar elements; (2) In the query process, with an input data sphere, a leaf sphere with the largest intersection spatial region and closest center is found, thus finding the most similar elements.

3.3. The principles of the operations on the tree

To ensure the storage of image data objects increasing from time to time, and enhance the performance of image retrieval systems, the R^S-Tree structure must satisfy the following two requirements: (1) having the ability to grow with the height of the tree; (2) having the ability to partition and cluster similar data. Based on these needs, adding $spED$ elements to the R^S-Tree must be done from the root node with the following principles:

Principle 1: If $root = null$

Making a $root$ is $rootLeaf$, putting in the sphere $spED$ at $rootLeaf$, updating the center and radius of the sphere.

Principle 2: If $root \neq null$ and $root$ is $leaf$

The Euclidean distance-based measuring:
$dist = d_E(spED.\overrightarrow{c}_{sp}, rootLeaf.\overrightarrow{c}_{leaf}) + spED.r_{sp}$

• If $dist \leq \theta$

+ If $root.count < M$:

Putting $spED$ into $rootLeaf$, updating the center and radius of the sphere $rootLeaf$.

+ If $root.count = M$, performing node splitting.

• If $dist > \theta$

Making a $NewLeaf$ to store $spED$ and a new root called $rootIn$ linked to $rootLeaf$ and $NewLeaf$.

Principle 3: If $root \neq null$ and $root$ is not $leaf$

Choosing the path from the current node to node that the nearest branch based on similar measure to go until seeing the leaf denoted $leaf_{crt}$.

Measuring $dist = d_E(spED.\overrightarrow{c}_{sp}, leaf_{crt}.\overrightarrow{c}_{leaf}) + spED.r_{sp}$

• If $dist \leq \theta$

+ If $leaf_{crt}.count < M$. Putting $spED$ into the current leaf, update the center, radius of the current leaf and then update back to the root.

+ If $leaf_{crt}.count = M$. Choosing the sphere element with Euclidean distance-based measure $dist = M\text{ax}\{d_E(spED_k.\overrightarrow{c}_{sp}, leaf_{crt}.\overrightarrow{c}_{leaf}) + spED_k.r \| k = 1..leaf_{crt}.count+1\}$. Then, choosing the neighbor node denoted S_{leafNN} of $leaf_{crt}$ to meet conditions: $S_{leafNN}.parent = leaf_{crt}.parent, S_{leafNN}.count < M$ and $dist_{\min}(d_E(S_{leafNN}.\overrightarrow{c}_{leaf}, spED_k.\overrightarrow{c}_{sp}) + spED_k.r) < \theta$. Adding the element $spED$ into S_{leafNN}. If there is no neighbor node that satisfies these conditions, split the node.

• If $dist > \theta$

Making new leaf node denoted $Newleaf$ store the element $spED$, updating the center and radius of the sphere from the current leaf along to the root node.

The process of adding the element $spED$ into the tree illustrated in Figure 4. The element $spED$ follows the branch S_{node} with the smallest distance measure $d_{\min}\{d_E(S_{node}.\overrightarrow{c}_{node}, spED.\overrightarrow{c}_{sp})\}$. When meeting the node near the leaf, choosing the suitable leaf S_{leaf} based on the criteria for finding the optimal spatial region is presented in Section 3.4.

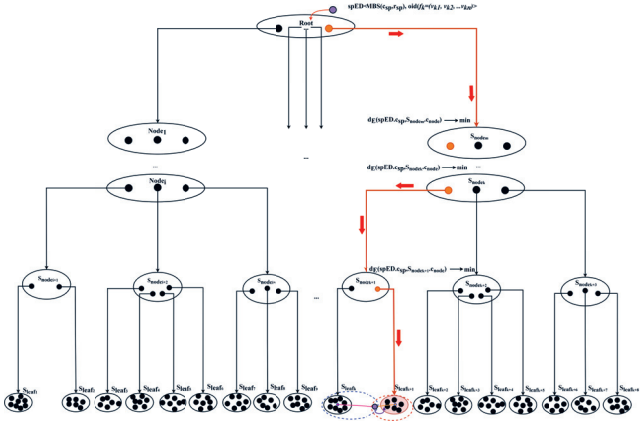


Figure 4. The illustration on the process of adding an element to R^S-Tree

3.4. Improve the operations on R^S-Tree structure

3.4.1. The operations of choosing the suitable leaf

To choose a suitable spatial region to add element $spED$ into the leaf with the following criteria satisfied:

Criteria 1. Choosing the leaf on the minimum extended spatial region.

When the element $spED$ is added, supposedly two leaves S_{leaf1}, S_{leaf2} are illustrated in Figure 5a. Because the extended spatial region of the leaf S_{leaf1} denoted $IncreaseArea(S_{leaf1}, spED)$, is smaller than of the leaf S_{leaf2} denoted $IncreaseArea(S_{leaf2}, spED)$. Therefore, S_{leaf1} is selected.

Criteria 2. Choosing the leaf with smaller spatial region.

When the element $spED$ is added, such as $IncreaseArea(S_{leaf1}, spED) = = IncreaseArea(S_{leaf1}, spED)$ illustrated in Figure 5b. At this time, the spatial region of the leaf node S_{leaf1}, denoted $Area(S_{leaf1})$, is smaller than of the leaf S_{leaf2}, denoted $Area(S_{leaf2})$. Therefore, $spED$ is added into S_{leaf1}.

Criteria 3. Choosing the leaf with the shortest distance from the center to the element added.

When the element $spED$ is added, such as $IncreaseArea(S_{leaf1}, spED) = = IncreaseArea(S_{leaf2}, spED) \wedge Area(S_{leaf1}, spED) = Area(S_{leaf2}, spED)$ illustrated in Figure 5c. At this time, we have $d_E(S_{leaf1}.\overrightarrow{c}_{l1}, spED.\overrightarrow{c}_{sp}) < < d_E(S_{leaf2}.\overrightarrow{c}_{l2}, spED.\overrightarrow{c}_{sp})$. Therefore, S_{leaf1} is selected.

Criteria 4. Choosing the leaf with the smaller number of elements.

When the element $spED$ is added, such as $IncreaseArea(S_{leaf1}, spED) = = IncreaseArea(S_{leaf2}, spED) \wedge Area(S_{leaf1}, spED) = Area(S_{leaf2}, spED) \wedge \wedge d_E(S_{leaf1}.\overrightarrow{c}_{l1}, spED.\overrightarrow{c}_{sp}) = d_E(S_{leaf2}.\overrightarrow{c}_{l2}, spED.\overrightarrow{c}_{sp})$ illustrated in Figure 5d. At this time, the leaf node S_{leaf2} has the number of elements less than the leaf S_{leaf1}. Therefore, $spED$ is added into S_{leaf2}.

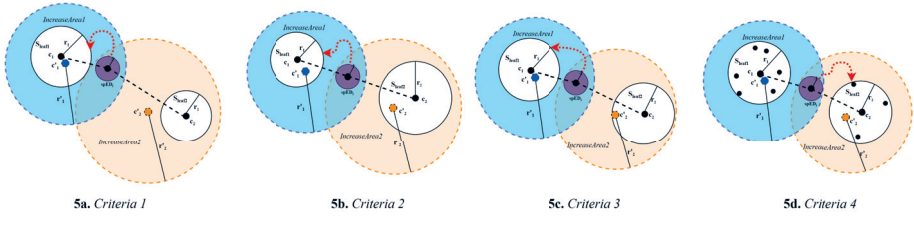


Figure 5. Description of the criteria to select the element distribution

The appropriate leaf is selected to cluster similar elements based on the criteria 1, 2, 3, 4 as above. On that basis, the suitable leaf choosing algorithm is presented as **Algorithm 1**.

Algorithm 1: SBLRST

Input:The leaf set S_L, $spED$
Output: $S_L[k]$ is the suitable leaf selected, $k \in [1, count\{S_L\}]$
Function: **SelectBestLeaf** (S_L,$spED$)
begin
  pos = $\arg\min(IncreaseArea(S_L[i], spED))$;
  $U_{Sk} = \{S_L[k] | S_L[k] \in \{S_L\}, IncreaseArea(S_L[k], spED) = IncreaseArea(S_L[\text{pos}], spED)\}$;
  if $U_{Sk}.size == 1$ **then**
    return $S_L[pos]$;
  else
    pos = $\arg\min\{S_L[\text{j}].\text{area}\}$;
    $U_{St} = \{S_L[\text{t}] | S_L[\text{t}] \in U_{Sk}, S_L[\text{t}].\text{area} = S_L[\text{pos}].\text{area}\}$;
    if $U_{St}.size == 1$ **then**
      return $S_L[pos]$;
    else
      $pos = \arg\min\{d_E(spED.\overrightarrow{c}, S_L[m].\overrightarrow{c})\}$;
      $U_{Sm} = \{S_L[m] | S_L[m] \in U_{St}, d_E(spED.\overrightarrow{c}, S_L[m].\overrightarrow{c}) = d_E(spED.\overrightarrow{c}, S_L[pos].\overrightarrow{c})\}$;
      if $U_{Sm}.size == 1$ **then**
        return $S_L[pos]$;
      else
        $pos = \arg\min\{S_L[r].size\}$;
        return $S_L[pos]$;
      end
    end
  end
end

Call M is the maximum amount of elements of a node on R^S-Tree. In the worst case, Algorithm SBLRST must be browsed M leaves. Therefore, the complexity of the Algorithm SBLRST is $O(M)$.

3.4.2. The operation of updating the center and radius

The process of adding, deleting, and splitting nodes on R^S-Tree leads to the storage space is changed to cover all the sub-space. Therefore, updating the center and radius of the nodes on R^S-Tree is performed. Let S_{sp} be a node on the R^S-Tree structure. In case S_{sp} is a leaf, let $S_{sp}.spED[i]$ be a sphere containing the data of the i^{th} child node of leaf S_{sp}, otherwise let $S_{sp}.spEC[i]$

be the sphere of the i^{th} child node of node S_{sp}. The algorithm of updating the center and radius presented as **Algorithm 2**.

Algorithm 2: USPCR

Input:S_{Node},R^S-Tree.
Output: R^S-Tree after updated
Function: **UpdateSphere**(S_{Node})
begin
  $Stack == null$;
  $Push(S_{Node}, Stack)$;
  while !$Empty(Stack)$ **do**
    $S_{sp} = Pop(Stack)$;
    if $S_{sp}.root == true$ **then**
      return R^S-Tree;
    end
    if $S_{sp}.leaf == true$ **then**
      $S_{sp}.\overrightarrow{c}_{sp} = avg(S_{sp}.SpED[i].\overrightarrow{c}_{spDi} | i = 1..S_{sp}.size)$;
      $S_{sp}.r_{sp} =$
        $Max\{\mathrm{d_E}(\mathrm{S_{sp}}.\overrightarrow{c}_{\mathrm{leaf}}, \mathrm{S_{sp}.SpED[i]}.\overrightarrow{c}_{\mathrm{spDi}}) + \mathrm{S_{sp}.SpED[i]}.\mathrm{r_{spDi}}\}$;
    else
      $S_{sp}.\overrightarrow{c}_{sp} = avg(S_{sp}.SpEC[i].\overrightarrow{c}_{spCi} \times S_{sp}.SpEC[i].w | i =$
        $1..S_{sp}.size)$;
      $S_{sp}.r_{sp} =$
        $Max\{\mathrm{d_E}(\mathrm{S_{sp}}.\overrightarrow{c}_{\mathrm{leaf}}, \mathrm{S_{sp}.SpEC[i]}.\overrightarrow{c}_{\mathrm{spCi}}) + \mathrm{S_{sp}.SpEC[i]}.\mathrm{r_{spCi}}\}$;
    end
    if $S_{sp}.parent \neq null$ **then**
      $S_{sp} = S_{sp}.parent$;
      $Push(S_{sp}, Stack)$;
    end
  end
end

Call n is the size of dataset. In R^S-Tree, each internal node of the tree stores up to N elements, each leaf node up to M and R^S-Tree with the height h. Because the data elements are only stored at the leaf layer of R^S-Tree, so we have $n \leq M \times N^h$. If $M \approx N$, the height of the tree is $h \approx log_M^{n-1}$. In the worst case, the Algorithm USPCR must update the cluster center from leaf to root and at each node browsing up to M element. Therefore, the complexity of the algorithm is $O(M \times log_M^n)$, with $log_M^n = log_M^2.log_2^n$. M is constant infer log_M^2 is also constant, $n \gg 2$. So, the complexity of the algorithm is $O(logn)$.

3.4.3. Node splitting algorithm

In the leaf splitting process, choosing element $spED$ to distribute into one of two new leaves is performed based on the different measure denoted d_{Dif}, illustrated in Figure 6.

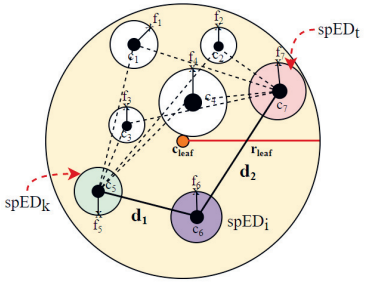


Figure 6. Description of the different measure of the element $spED_i$

In Figure 6, $spED_k$ and $spED_t$ is two elements selected to be the center of two new leaves. $spED_i$ is a element in remainder set of splitting leaf. The different measure d_{Dif} of the element $spED_i$ compared with $spED_k$ and $spED_t$, denoted S_{L1} and S_{L2} measured like $d_{Dif} = |d_1 - d_2|$, where $d_1 = = d_E(spED_i.\overrightarrow{c}_{spi}, spED_k.\overrightarrow{c}_{spk})$, $d_1 = d_E(spED_i.\overrightarrow{c}_{spi}, spED_k.\overrightarrow{c}_{spt})$. The bigger the difference measure is, the higher the probability of identifying the element belongs to either node.

R^S-Tree is a dynamically balanced tree structure growing from leaf to root. The overflowed node S_L is handled by creating two nodes S_{L1} and S_{L2} same level with S_L. $M + 1$ elements is distributed into the two nodes. This process can spread to the root, when the root is full, other root will be created and R^S-Tree grown up. The leaf splitting algorithm is presented as **Algorithm 3**.

Call n is size of the dataset, M is the maximum amount of the element in a node on R^S-Tree. When performing splitting the node, in the worst case, algorithm SBLRST must be performed from the leaf to the root. Each time of splitting, algorithm SpLRST must perform M comparison operations. On the other hand, every time a splitting node is performed, in the worst case, algorithm SpLRST must be called from the leaf to the root. M is constant. Therefore, the complexity of the algorithm SpLRST is $O((logn)^2)$.

Algorithm 3: SpLRST

```
Input:Leaf S_L.
Output: R^S-Tree after splitting
Function: SplitLeafRST(S_L)
begin
    CreateNewLeaf(S_L1);
    CreateNewLeaf(S_L2);
    spED1, spED2 = SelectedspEDs(S_L);
    Use function AddspED() to add a element into a node.
    S_L1.AddspED(spED1);
    S_L2.AddspED(spED2);
    S_L = S_L\{spED1, spED2};
    Use function SortListSpED() to arrange remainer elements follow
     different measure.
    SortListSpED(S_L, spED1, spED2);
    S_Lpr = S_L.parent;
    if S_Lpr! = null then
        Delete(S_L);
        S_L1.parent = S_Lpr;
        S_L2.parent = S_Lpr;
        if S_Lpr.size < N then
            UpdateSphere(S_Lpr);
        else
            Function SlpitNodeSRT() is used to split internal node.
            SlpitNodeSRT(S_Lpr);
        end
    else
        CreateRootInNode(S_Nr);
        S_L1.parent = S_Nr;
        S_L1.parent = S_Nr);
        UpdateSphere(S_Nr);
    end
end
```

3.4.4. The operation of adding an element

When the element $spED =< MBS(\vec{c}_{sp}, r_{sp}); f >$ is added in R^S-Tree, the insertion is done from the root, browsing all children nodes of the root and following the nearest branch based on similar measure until finding a node denoted S_{Nodek} next to leaf. After that, the element $spED$ is distributed on one of the child nodes of the current node with the condition $d_E(spED.\overrightarrow{c}_{sp}, S_{leaf}.\overrightarrow{c}_{leaf}) + spED.r_{sp} < \theta$ in **Algorithm 1**. SBLRST, otherwise, a new leaf $S_{newleaf}$ is created to store the element $spED$. The algorithm added elements is presented as **Algorithm 4**.

Algorithm 4: IspEDRST

Input:Leaf node S_L,threshold M, element $spED$.
Output: R^S-Tree after adding the element
Function: **InsertspEDintoRST**(S_L, $spED$,M)
begin
  if $S_L.leaf == true$ **then**
    if $S_N.size < M$ **then**
      $d = d_E(spED.\overrightarrow{c}, S_N.\overrightarrow{c}) + spED.r$;
      if $d < \theta$ **then**
        $S_N.AddSpED(spED)$;
      else
        $CreateNewLeaf(S_L)$;
        $S_L.AddSpED(spED)$;
      end
    else
      $SplitLeafRST(S_L)$;
    end
  else
    if $S_L.Child.leaf == false$ **then**
      Use function SelectBestBranch to select the best branch follow similar measure
      $S_k = SelectBestBranch(S_N, spED)$;
      $InsertspEDintoRST(S_k, spED)$;
    else
      $S_{leafk} = SelectBestLeaf(S_L, spED)$;
      if $S_{leafk}.size < M$ **then**
        $d = d_E(spED.\overrightarrow{c}, S_{leafk}.\overrightarrow{c}) + spED.r$;
        if $d < \theta$ **then**
          $S_{leafk}.AddSpED(spED)$;
        else
          $CreateNewLeaf(S_L)$;
          $S_L.AddSpED(spED)$;
        end
      else
        $SplitLeafRST(S_L)$;
      end
    end
  end
end

Call n is the size of dataset, M is the maximum number of the in the node of R^S-Tree. Algorithm IspEDRST performed browsing from root to leaf, each time to browse M element, and algorithm IspEDRST perform the center updating and node splitting from the leaf to the root, M is constant. Therefore, algorithm IspEDRST with the complexity is $O(logn)^3$.

4. Similar image retrieval model in R^S-tree structure

4.1. The proposed model of CBIR-RST

An query image I is extracted the feature vector and retrieved on R^S-Tree. The process of retrieving is performed on R^S-Tree until meeting the leaf, then the set of all data elements in the leaf is called a similar image set of the queried image. Content-based image retrieval model on R^S-Tree illustrated in Figure 7.

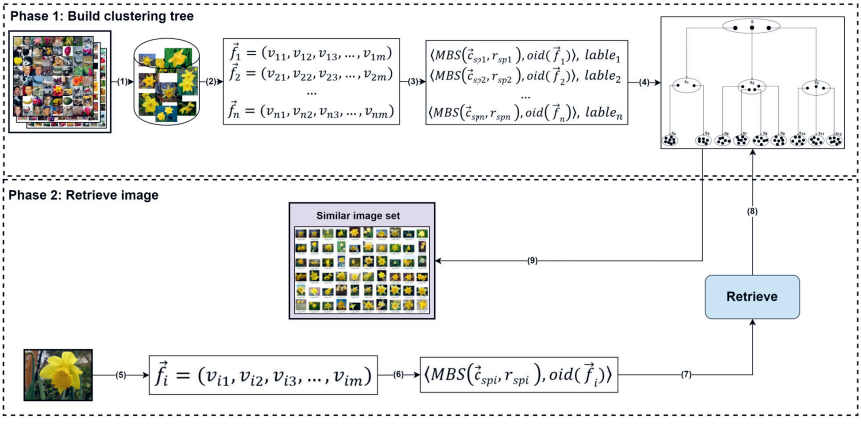


Figure 7. Model of content-based image retrieval of CBIR-RST on R^S-Tree

The image retrieval process is performed in two phases: the first phase performs clustering and stores the image data in R^S-Tree, the second phase performs to retrieval for similar images set from the queried image. The process is described as follows:

Build clustering tree. The process of building a clustered R^S-Tree based on the feature vector of the image dataset comprises the following steps:

Step 1. Extract the feature vectors of the image dataset.

Step 2. Represent the feature vector of image dataset in the form of sphere.

Step 3. Indexing clustered structure named R^S-Tree is trained based on the proposed similar measure and k-Mean clustering method. Each leaf of the tree is a set of the sphere element containing the vectors f_i to describe the visual features of the image.

Image retrieval. The retrieval phase include the following steps:

Step 1. From a query image Iq, the feature vector is extracted and convert into a spatial sphere.

Step 2. Perform similar image retrieval based on indexed clustering structure R^S-Tree.

Step 3. Arrange similar images by similarity measure of query image.

4.2. Image retrieval based on R^S-Tree

From indexing clustered structure R^S-Tree created, an algorithm for content-based similar images retrieval on R^S-Tree is proposed. The process of similar image retrieval on R^S-Tree is described as **Algorithm 5**.

Algorithm 5: RSIR

```
1  Input:element spED of the image retrieval, R^S-Tree.
2  Output: the similar image set SI
3  Function: RSIR(S_Nr, spED)
4  begin
5      S_Node = S_Nr;
6      if S_Node == null then
7          return null;
8      else
9          if S_N.Child.leaf == false then
10             S_Nk = SelectBestBranch(S_Node,spED);
11             RSIR(S_Nk,spED);
12         else
13             SI = S_Node.Child.spEC[i].ListSpED|i = 1..S_Node.size;
14         end
15     end
16     return SI;
17 end
```

Call n is size of the dataset, M is the maximum number of the element in a node of R^S-Tree. Algorithm RSIR browses from the root to the leaf. Each time of browsing, algorithm RSIR must be compared with M element of a node, M is constant. Therefore, the complexity of the algorithm RSIR is $O(logn)$.

5. Experiments and discussions

5.1. Extracting the Feature Vector

In this paper, feature of image were extracted including MPEG7 color feature, shape feature, texture feature, Laplacian of Gaussian for detecting object, object recognition based on boundary and surface with Sobel filter, enhancing pixel intensity with Gaussian filtering [29]. Experiment of feature extraction process 049.jpg image in Daffodil folder, Oxford Flowers 17 dataset is illustrated by Figure 8. In which (a) original image; (b) object mask image (ForeGround); (c) object image; (d) original photo wallpaper; (e) feature contour image by Laplace filter for feature mask image; (f) object contour image and object surface texture according to Sobel filter; (g) Contour image of the original image according to Laplace and Gaussian filters; (h) high-pass filter image for original image and feature vector consisting of 81 components. To extract image

features, firstly, each input image is segmented into many small regions. Secondly, the methods of extracting color, shape, and texture features [30] are performed. The image feature vector is extracted and inherited from the work [29, 30]. Then, the vectors are represented into the spatial sphere and stored on the R^S-Tree.

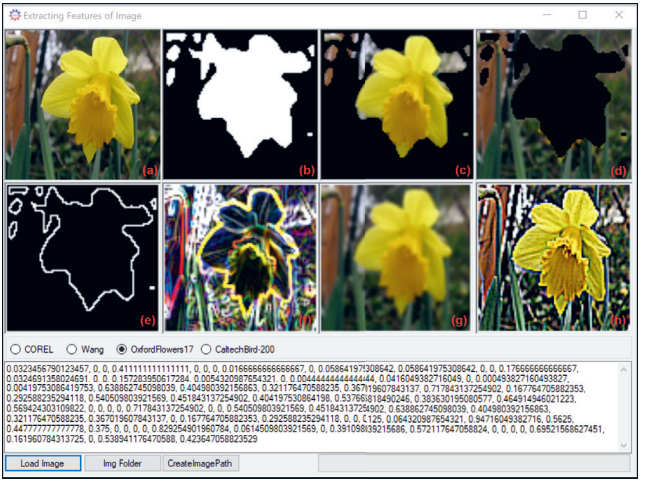


Figure 8. An example of the low-level feature extraction of the image

5.2. Experimental environment

The build clustering tree phase is performed on PC CPU 2.3GHz 8-core 9th-generation Intel Core i9, 16GB 2666MHz memory, 1TB flash storage. The image retrieval phase is examined on PC CPU Intel Core i7-6500U CPU @ 2.50GHz, 8.0GB RAM, the operating system of Windows 10 Pro 64 bit.

In this paper, we experiment on three image sets including COREL, Wang, Oxford Flowers 17. Where COREL has 1000 images divided into 10 classes [23], Wang has 10.800 images divided into 80 classes [24], Oxford Flowers 17 has 1360 images divided into 17 classes [25]. Experimental application is illustrated in Figures 9 and 10.

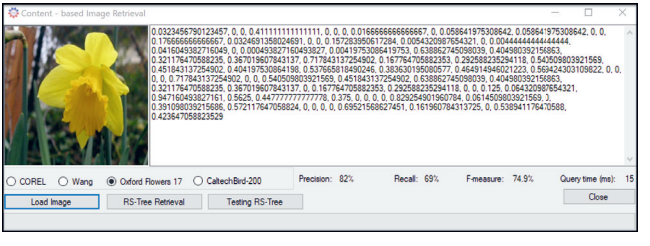


Figure 9. Image retrieval interface based on R^S-Tree

Figure 10. A result of image retrieval in R^S-Tree

5.3. Experimental evaluation

In this paper, the parameters M, m, N, θ are uesed in the process of building R^S-Tree to improve the retrieval precision. Experimental processing is conducted as follows: Let $nMax_{inclass}$ be the maximum number of elements in a class. We choose $M \in [nMax_{inclass} - \alpha, nMax_{inclass} + \alpha]$, α is constant; $N \in [2, M]$. Threshold θ to evaluate the similarity of elements belonging to a cluster. The number elements of each classifier estimates approximately 10% of the dataset. In the experiments of this paper, we experiment with the values $\theta \in [0.1 - \mu, 0.1 + \mu], \mu \in [0, 0.05]$ to cluster the data depending on the dataset. Optimal parameters in the experimental process, the Top of retrieved elements and R^S-tree build time are presented in **Table 1**.

Parameter	COREL	Wang	Oxford Flowers 17
M	120	100	80
m	1	1	1
N	20	30	20
θ	0.095	0.065	0.08
$Topk$	80	65	45
R^S-Tree building time (hours)	0.50	3.57	0.65

Table 1. Description of experimental parameters

To evaluate the effectiveness of proposed method, we used the following as evaluation metrics: precision, recall and F-measure. The experimental results are shown in Figures 11–14. Each curve on the graph describes retrieval results from the image class in the dataset of COREL, Wang, Oxford Flowers 17. In addition, the corresponding curve in the ROC graph shows the ratio of true and false retrieval results, that is, the area under this curve evaluates the correctness of retrieval results.

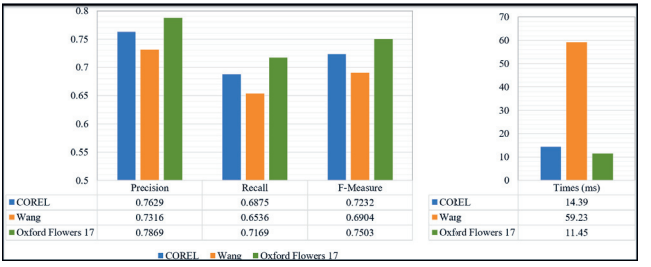


Figure 11. The average value of Precision, Recall, F-measure

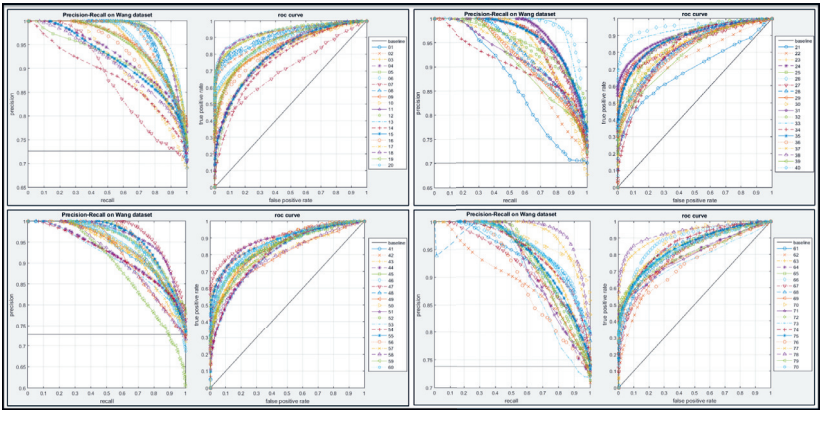


Figure 12. Precision-Recall and curving ROC of WANG dataset

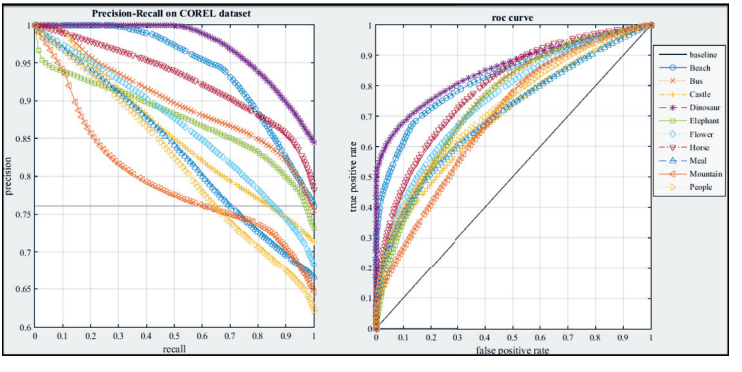


Figure 13. Precision-Recall and curving ROC of COREL dataset

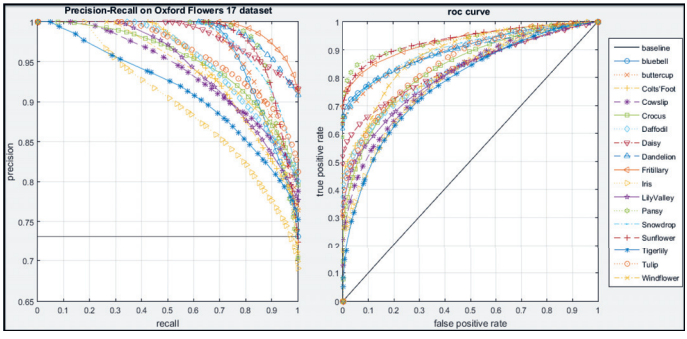


Figure 14. Precision-Recall and curving ROC of Oxford Flowers 17 dataset

The retrieval performance of proposed algorithm based on R^S-Tree is described in **Table 2**, **Table 3**, **Table 4**. To evaluate the performance of the proposed model, we compare with the results of previous related works on the same dataset described in **Table 5**.

Folders	Avg.recall	Avg.precision	Avg.F-measure	Avg.time (ms)
00-10	68.75%	76.29%	72.32%	14.39

Table 2. Performance of image retrieval of CBIR-RST on COREL dataset

Folders	Avg.recall	Avg.precision	Avg.F-measure	Avg.time (ms)
00-20	63.55%	69.94%	66.59%	56.21
21-40	64.54%	71.86%	68.00%	45.61
41-60	66.17%	73.98%	69.86%	65.41
61-80	67.18%	76.84%	71.70%	69.71
AVG	65.36%	73.16%	69.04%	59.23

Table 3. Performance of image retrieval of CBIR-RST on Wang dataset

Folders	Avg.recall	Avg.precision	Avg.F-measure	Avg.time (ms)
00-09	74.99%	81.99%	78.34%	10.78
10-17	67.98%	74.98%	71.30%	12.11
AVG	71.69%	78.69%	75.03%	11.45

Table 4. Performance of image retrieval of CBIR-RST on Oxford Flowers 17 dataset

The experimental result tables show that the performance of the proposed method is quite high, because of the following reasons: (1) the R^S-Tree is built rely on feature vectors in the form of spheres to optimize storage space; (2) a threshold θ ($0 < \theta < 1$) is proposed to cluster similar image; (3) the node splitting algorithm is improved to enhance retrieval efficiency.

Methods	MAP(%)	Dataset
L. Haldurai, 2015 [14]	73.88	COREL
Ahmed, 2019 [19]	72.10	COREL
CBIR-RST	**76.29**	**COREL**
Lande, Milind V, 2014 [13]	61.00	Wang
J. Vanitha, 2017 [16]	75.80	Wang
P. Chhabra, 2020 [20]	63.20	Wang
CBIR-RST	**73.16**	**Wang**
Ahmed, 2019 [19]	77.10	Oxford Flowers 17
S. Gao, 2014 [22]	73.43	Oxford Flowers 17
CBIR-RST	**78.69**	**Oxford Flowers 17**

Table 5. Comparison mean average precision (MAP) of methods on dataset

Table 5 shows the retrieval accuracy of the method proposed is quite effective. In work [14] used R-Tree to store image data. However, the node splitting process in R-Tree can have similar elements, but it still splits into two nodes. In the worst case, these elements are in two distant branches. In addition, R-Tree is stored in rectangular blocks, so it wastes the storage space and complex computation time. In work [19], the authors extracted color features from RGB images and used the gray level image for local features. However, in this study, the authors only performed indexing on a subjective basis and did not propose a data indexing structure to improve retrieval performance. In this work, the authors took the top 10 images with the precision of value 87,1% (COREL) and 95,5% (Oxford Flowers 17). However, with the top 40 images, precision is 72.1% (COREL) and 77,1% (Oxford Flowers 17). In work [20] used the methods ORB and SIFT to extract image features of eight length the accuracy is 86.2% but the coverage was not high because the author took the top 20 images. In addition, these two methods require huge storage space and computational. In work [13] used co-occurrence matrix and Fourier transform to extract image features. The studies used k-Mean clustering to partition the data and did not propose a storage structure to improve retrieval effectiveness. In work [16] used SR-Tree to store image data. However, the process of inserting on SR-tree needs to update both spheres and rectangles bounding. Thus, it is relatively complicated and expensive to create and update. The accuracy is 75.8% but the coverage was not high (16%). In [22], the authors have combined dictionary structure and the Support Vector Machine technique. However, this method is costly in training time, and the authors only focused on local features, did not combine advanced features to enhance image classification performance. In this paper, the improvement study proposed the threshold to cluster similar elements, and R^S-Tree stores the image elements by the sphere. In addition, the node splitting algorithm is improved based on a different measure. Therefore, retrieval results of the proposed method have higher accuracy and faster retrieval time.

6. Conclusions

In this paper, an improved R^S-Tree structure is built based on R-Tree and applied to the image retrieval systems. This structure ensures a large storage capacity since feature vectors are represented in the form of spheres. In addition, R^S-Tree is built based on partitional clustering and hierarchical clustering method, each leaf stores similar data elements, thus improving image retrieval efficiency. From then, a content-based image retrieval system, $CBIR\_RST$, was built based on the R^S-Tree. The experiment was performed on COREL, Wang, Oxford Flowers-17 image dataset with the precision value of 76.29%, 73.16%, and 78.69%, respectively. To evaluate the proposed methods, the experimental results were compared with other methods on the same image dataset. The comparison results indicate that the $CBIR\_RST$ system is effective. In the future work, we will build a Graph Convolution Network (GCN) from the leaves in R^S-Tree to optimize the algorithm of image retrieval based on similar clusters and create neighboring clusters on GCN graph.

Acknowledgment. The authors would like to thank the Faculty of Information Technology, University of Sciences - Hue University for their professional advice for this study. We would also like to thank HCMC University of Food Industry, Ba Ria - Vung Tau University, University of Education HCMC, and research group $SBIR\_HCM$, which are sponsors of this research. We also would like to express our sincere thanks to reviewers for helpful comments on this article.

References

[1] **Zhang, D., M.M. Islam and G. Lu,** A review on automatic image annotation techniques, *Pattern Recognition*, **45(1)** (2012), 346–362.

[2] **Liu, Y., D. Zhang, G. Lu and W.-Y. Ma,** A survey of content-based image retrieval with high-level semantics, *Pattern Recognition*, **40(1)** (2007), 262–282.

[3] **Mehmood, Z., F. Abbas, T. Mahmood, M.A. Javid, A. Rehman and T. Nawaz,** Content-based image retrieval based on visual words fusion versus features fusion of local and global features, *Arabian Journal for Science and Engineering*, **43(12)** (2018), 7265–7284.

[4] **Zhou, Wengang, H. Li, and Q. Tian,** Recent advance in content-based image retrieval: A literature survey, arXiv preprint (2017). https://arxiv.org/pdf/1706.06064.pdf

[5] **Ismail, M.M.B.,** A survey on content-based image retrieval, *Int. J. Adv. Comput. Sci. Appl.*, **8(5)** (2017), 159–170.

[6] **Shaik, A.N. Begum and K.P. Supreethi,** A survey on spatial indexing, *Journal of Web Development and Web Designing*, **3(1)** (2018), 1–25.

[7] **Szalai-Gindl, J.M., A. Kiss, G. Halász, L. Dobos, and I. Csabai,** Nd-gist: A novel method for disk-resident k-mer indexing, in: *World Conference on Information Systems and Technologies*, Springer, Cham, 2019, pp. 663–672.

[8] **Gombos, G., J.M. Szalai-Gindl, I. Donkó and A. Kiss,** Towards on experimental comparison of the m-tree index structure with bk-tree and vp-tree, *Acta Electrotechnica et Informatica*, **20(2)** (2020), 19–26.

[9] **Guttman, A.,** R-trees: A dynamic index structure for spatial searching, in: *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Boston, MA, 1984, pp. 47–57.

[10] **Manolopoulos, Y., A. Nanopoulos, A.N. Papadopoulos and Y. Theodoridis,** *R-Trees: Theory and Applications*, Springer, London, 2010, pp. 117–125.

[11] **Donkó, I., J.M. Szalai-Gindl, G. Gombos and A. Kiss,** An implementation of the M-tree index structure for PostgreSQL using GiST, in: *IEEE 15th International Scientific Conference on Informatics, IEEE*, 2019, pp. 189–194.

[12] **Chandresh, P.S.,** R-Tree implementation of image databases, *Signal and Image Processing: An International Journal* (*SIPIJ*), **2(4)** (2011), 89–108.

[13] **Lande, M.V., P. Bhanodiya and P. Jain,** An effective content-based image retrieval using color, texture and shape feature, *Intelligent Computing, Networking and Informatics*, Springer, New Delhi, 2014, pp. 1163–1170.

[14] **Haldurai, L. and V. Vinodhini,** Parallel indexing on color and texture feature extraction using R-Tree for content based image retrieval, *International Journal of Computer Sciences and Engineering*, **3** (2015), 11–15.

[15] **Abd, A., D.N.A. Binti, N. Higuchi and T. Shinohara,** A study on optimization of similarity search by R-tree using dimension reduction, *Information Processing Society of Japan*, (2018).

[16] **Vanitha, J. and M. S. Murugan,** An efficient content based image retrieval using block color histogram and color co-occurrence matrix, *International Journal of Applied Engineering Research*, (2017), pp. 15966–15971.

[17] **Shama, P.S., K. Badrinath and A. Tilugul,** An efficient indexing approach for content based image retrieval, *International Journal of Computer Applications*, **117(15)** (2015), 11–18.

[18] **Alfarrarjeh, A., S.H. Kim, V. Hegde, C. Shahabi, Q. Xie and S. Ravada,** A class of R*-tree indexes for spatial-visual search of geotagged street images, in: *IEEE 36th International Conference on Data Engineering* (*ICDE*)*, IEEE*, 2020, pp. 1990–1993.

[19] **Ahmed, K.T., S. Ummesafi and A. Iqbal,** Content based image retrieval using image features information fusion, *Information Fusion*, **51** (2019), 76–99.

[20] **Chhabra, P., N.K. Garg and M. Kumar,** Content-based image retrieval system using ORB and SIFT features, *Neural Computing and Applications*, 2020, 2725–2733.

[21] **Kumar, V., V. Tripathi and B. Pant,** Content based fine-grained image retrieval using convolutional neural network, in: *7th International Conference on Signal Processing and Integrated Networks* (*SPIN*)*, IEEE*, 2020, pp. 1120–1125.

[22] **Gao, S., I.W.H. Tsang and Y. Ma,** Learning category-specific dictionary and shared dictionary for fine-grained image categorization, *IEEE Transactions on Image Processing*, **23(2)**, (2014), 623–634.

[23] **Jia, L. and Z.W. James,** Automatic linguistic indexing of pictures by a statistical modeling approach, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25(9)** (2003), 1075–1088.

[24] **James, Z.W., L. Jia and G. Wiederhold,** SIMPLIcity: Semantics-sensitive integrated matching for picture libraries, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23(9)** (2001), 947–963.

[25] **Nilsback, M.-E. and A. Zisserman,** A visual vocabulary for flower classification, in: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (*CVPR'06*)*, vol. 2,. IEEE*, 2006, pp. 1447–1454.

[26] **White, D.A. and R.N. Jain,** Similarity indexing with the SS-tree, in: *Proceedings of the Twelfth International Conference on Data Engineering, IEEE*, 1996, pp. 516–523.

[27] **Katayama, N. and S.I. Satoh,** The SR-tree: An index structure for high-dimensional nearest neighbor queries, *ACM Sigmod Record*, **26(2)** (1997), 369–380.

[28] **Wu, M.K.,** *Evaluation of R-trees for Nearest Neighbor Search*, Doctoral dissertation, University of Houston, 2006.

[29] **Nhi, N.T.U., T.V. Thanh and T.M. Le,** Semantic-based image retrieval using balanced clustering tree, in: *WorldCIST*, **2** (2021), 416–427.

[30] **Chaki, J.,** Image color feature extraction techniques: Fundamentals and applications, Singapore, 2021, Springer, pp. 29–80.

Le Thi Vinh Thanh
University of Ba Ria Vung Tau
Ba Ria Vung Tau
Faculty of Information Technology
University of Sciences – Hue University
Hue
Vietnam
thanhltv@bvu.edu.vn

Thanh The Van
Faculty of Information Technology
HCMC University of Education
HoChiMinh city
Vietnam
thanhvt@hcmue.edu.vn

Thanh Manh Le
Faculty of Information Technology
University of Sciences – Hue University
Hue
Vietnam
lmthanh@hueuni.edu.vn