Annales Univ. Sci. Budapest., Sect. Comp. 53 (2022) 3–28

A SEMANTIC-BASED IMAGE RETRIEVAL SYSTEM USING NESTED KD-TREE STRUCTURE

Nguyen Thi Dinh and 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. Semantic-based image retrieval system has relied on many different methods. In this paper, a semantic-based image retrieval method using Nested KD-Tree structure and ontology is proposed. The Nested KD-Tree structure is built depending on an image classification method based on KD-Tree. Firstly, KD-Tree is built as a balanced multi-branch tree; Secondly, some leaf nodes on KD-Tree are grown into SubTrees to create a Nested KD-Tree structure. Therefore, an algorithm of Nested KD-Tree construction and a method of image classification are performed. Each query image is extracted to a feature vector to perform image classification and retrieve similar images based on Nested KD-Tree. Then, a SPARQL query is generated to retrieve similar images by semantic-based on ontology. On the basis of this theory, a semantic-based image retrieval model is proposed to build an experiment. The precision on experimental image data sets including COREL, Wang, and Caltech101 of 81.19%, 80.29%, 72.55%, respectively. The experiment results are compared to the published works on the same data set to demonstrate the efficiency of our proposed method.

1. Introduction

Over the decade, the image retrieval problem attracted many researchers owing to the enormous increase in digital images and digital image repositories. According to statistics from the International Data Group (IDC) [1], the

Key words and phrases: Nested KD-Tree, image classification, image retrieval, similar image, ontology.

The ACM Computing Classification (1998): H.2.8, H.3.3.

increase in digital image data has brought many challenges and opportunities for image retrieval systems. Some of the criteria to evaluate the performance of the image retrieval system include storage capacity, query time, feasibility, and others. Therefore, a data structure to improve the capacity of data storage and enhance performance for the image retrieval problem is necessary. In this paper, a Nested KD-Tree structure is proposed and applied to a semantic-based image retrieval system. Firstly, KD-Tree is built to perform image classification by multidimensional feature vectors and create clusters at the leaf nodes. Secondly, some leaves are grown into SubTrees to form a Nested KD-Tree structure.

Semantic-based image retrieval system has been implemented by many methods such as using machine learning techniques, an ontology, related feedback methods, and others. In this paper, a semantic-based image retrieval method based on ontology is approached. Firstly, building a Nested KD-Tree structure to store and perform image classification; Secondly, retrieving similar images based on the Nested KD-Tree; Finally, extracting a set of visual words, creating SPARQL query to retrieve a set of similar images by semantic. The semantic-based image retrieval system based on the Nested KD-Tree and ontology has been evaluated as effective.

The contribution of the paper includes (1) proposing a method of image classification based on the Nested KD-Tree; (2) proposing an algorithm to build Nested KD-Tree, a method for assigning a label to leaf, weight training algorithm, and retrieval algorithm based on the Nested KD-Tree; (3) proposing a semantic-based image retrieval model based on Nested KD-Tree and ontology; (4) building experiment and proving the efficiency of the proposed method on COREL [2], Wang [3], Caltech101 [4] image data sets.

The rest of this paper is as follows: In Section 2, we survey and analyze related works of image classification and semantic-based image retrieval systems. In Section 3, we present Nested KD-Tree construction algorithm, a method for assigning a label to leaf, weight training algorithm, and retrieval algorithm; Section 4, the general architecture of SB-NKDT is described. In Section 5, we build an experiment and evaluate the effectiveness of the proposed method. Conclusions and future works are presented in Section 6.

2. Related works

Image classification method applied to the image retrieval problem that has been published by many works based on machine learning techniques such as using the SVM classification algorithm (Support Vector Machine) [6], the k-Nearest Neighbors (k-NN) algorithm [7], the decision tree (Decision Tree) [8]; the KD-Tree combined with the k-NN algorithm [9], the convolution neural

network (CNN) [10] and others. The results of image classification are applied to the image retrieval problem that has brought high performance. In addition, many semantic-based image retrieval systems meet the increasing popular user demands and have applied to many fields such as object recognition, pattern analysis, information retrieval based on images, etc.

Yuqian Zhang et al. (2016) [13] used an image classification method for face recognition. In this work, at the training phase, each image in the data set is divided into several regions; the KD-Tree was built based on the partitioned image data set to perform image classification. At the testing phase, each image was divided into multiple partitions and the KD-Tree was used to retrieve the k-Nearest Neighbor (k-NN) by matching the query image regions with the original image. Besides, the KD-Tree was built as a multidimensional index structure to reduce the retrieval time. Finally, the authors experimented on CUFS image data set (Chinese University Face Sketch) to prove this proposed method for face recognition was effective.

Hakan Cevikalp et al. (2017) [14] retrieved the image using a Graph-Cut graph and hierarchical binary tree. The images are performed classification according to the SVM algorithm based on the low-level features. However, this work did not perform the semantics of image classification. M. Jiu and H. Sahbi (2017) [15] used a multilayer neural network based on different nonlinear activation functions at each layer combined with the SVM technique. This work was applied to perform image classification at the output layer and to define the semantics for the set of similar images. M. Tzelepi and A. Tefas (2018) [16] proposed a method to train a CNN network for a content-based image retrieval system based on the Caffe Deep Learning framework. In this paper, the authors performed image classification by low-level features based on relevant feedback to apply to the semantic-based image retrieval problem.

M. Bennet Rajesh et al. (2020) [25] implemented a content-based image retrieval system using Co-occurrence of Edges and Valleys with Support Vector Machine algorithm. The proposed system was tested on Caltech 101 data set with an accuracy of 63.14%. The proposed system obtained better efficiency than state-of-the-art methods.

Applying image classification results to the image retrieval problem has been published by many works. These works of semantic-based image retrieval have been interested in many research groups recently. One of the approaches to the semantic-based image retrieval problem is using ontology. To retrieve the set of similar images by semantic, SPARQL query [11, 12] has been created to retrieve on ontology.

V. Vijayarajan et al. (2016) [17] performed image retrieval based on natural language analysis to generate SPARQL queries. The image retrieving process depends on analyzing the grammar of the language to form keywords that describe the image content. Content-based image classification was not performed

by color characteristics and spatial features to generate keywords for retrieval. Therefore, retrieving an image hasn't been performed from an input image yet.

M.N. Asim et al. (2019) [18] implemented an information retrieval method based on ontology applied to text queries, multimedia data (images, videos, audio). In this work, the authors used the RDF triple language to retrieve similar images based on ontology and simultaneously compares its performance with the others of previous approaches. This work was evaluated as effective.

N.T.U Nhi, V.T. Thanh, T.M Le (2020) [19] implemented an image retrieval method based on balanced clustering tree structure C-Tree; after that, the semantic-based image retrieval system was implemented on ontology by SPARQL query language. Firstly, the authors performed input image classification and experimented on the Wang data set with the classification efficiency of 68.93%. This classification result generated a SPARQL query to retrieve similar images on ontology. The experimental results of semantic-based image retrieval on the Wang data set with the precision of 60.72% compared with the works on the same data set. This work was evaluated as effective.

Nguyen T. D., Van T. T., and Le M. T. (2021) [27] have implemented an image classification method by the KD-Tree, which is feasible and effective. However, there are some limitations: (1) The KD-Tree is a binary tree, the classification accuracy is not good if the image classification process of an image is wrong at the root node; (2) The height of this tree is large if training data sets have large classifiers that effects on the speed of image retrieval; (3) The KD-Tree was not able to increase to the number of classifiers to meet the requirements of increasing the training data sets; (4) Some leaves in the KD-Tree has contained too much data.

In this paper, an image classification method is proposed using Nested KD-Tree and applied to a semantic-based image retrieval system. The Nested KD-Tree has performed as a multi-layer data classification for an object many times. Some leaves can be grown into SubTrees to continue performing image classification once. After conducting image classification, a process of content-based image retrieval is performed based on Nested KD-Tree; a set of visual words is extracted; a SPARQL query is created to retrieve a set of similar images by semantics based on ontology.

3. Nested KD-Tree structure for image retrieval

The paper presents a semantic-based image retrieval system with KD-Tree for image classification in section 3.2. Phase 1 builds a balanced multi-branch KD-Tree that is illustrated by figure 2. Phase 2 builds SubTree from some leaves on KD-Tree. The experimental process shows that some leaves contain too many image data belonging to many different classes; so the image classification

performance on KD-Tree is not good. The process of division leaf to SubTree is necessary. Because, the process of division leaf makes the leaf partition with a large number of elements, which helps to partition these elements again, so the image classification accuracy will increase. Simultaneously, this process helps to reduce the cost of weight training on KD-Tree. Experimental results show that the result of image classification on Nested KD-Tree is better than KD-Tree. In this paper, two parameters M, N are conditions to build SubTree at $Leaf_k$ after building KD-Tree. M is a number of elements at $Leaf_k$. N is a number of labels at $Leaf_k$. The threshold values M and N depend on the experimental data set and present in Subsection 3.2. The process of building SubTree is only performed at some leaves that satisfy the conditions in table 1 and the other leaves remain unchanged. The Nested KD-Tree is built by the following steps:

1) Building a KD-Tree.

2) Assigning a label to the leaf on KD-Tree.

3) Training weight on KD-Tree.

4) Finding leaves that satisfy the conditions in table 1 to built SubTree.

5) Assigning a label to the leaf on Subtree.

6) Training weight on Subtree.

7) Linking KD-Tree and SubTree to create Nested KD-Tree.

3.1. General description of Nested KD-Tree structure

Multidimensional data structure plays an important role in storing image data and affects to query time of the image retrieval system. Some multidimensional data structures such as KD-Tree [20], R-Tree [21], M-Tree [22], and others are used for image retrieval systems. In this paper, the multidimensional data structure KD-Tree [20] is developed into a Nested KD-Tree structure and applied to a semantic-based image retrieval problem. The Nested KD-Tree structure contains some following components:

1) A root node ($Root$) stores weight vector (w_0), has a set of children nodes $\{children\}$ and a level: $Root =< w_0, \{children\}, level >$. Because level of root is the depth of a node then it is always 0.

2) An internal node ($Node_i$) is a node that has a parent node, stores a weight vector (w_i), has a set of children nodes $\{child\}$ and a level: $Node_i = =< parent, w_i, \{child\}, level >$

3) A leaf node ($Leaf_i$) is a node that has a parent node, has no children nodes, stores a set of multidimensional vector $F = \{f_1, f_2, \ldots, f_k\}$, has a level and is assigned a label: $Leaf_i =< parent, F, level, label >$

Based on the components of Nested KD-Tree, there are three properties of the Nested KD-Tree, such as:

1) Two nodes, $Node_i$ and $Node_j$ have a brotherly relation if $Node_i.parent = Node_j.parent$
2) Two nodes, $Node_i$ and $Node_j$ have a filiation if $Node_j.parent = Node_i$ or $Node_i.parent = Node_j$
3) Two nodes, $Node_i$ and $Node_j$ are peers relation if $Node_i.level = Node_j.level$

There are many activation functions such as Linear, Sigmoid, Tanh, Sign function, etc. In this paper, at each node on KD-Tree, the output value is used Sigmoid activation function. This activation function is chosen because the value of the Sigmoid function belongs to the range [0..1] and it is the integrity constraint for the problem. Besides, the weight training process is done by the method of taking the Gradient opposite derivative of the Sigmoid function. The weight training process is to adjust the weight vectors at nodes to reduce the error when classifying images. Therefore, the Sigmoid activation function is the best choice in this problem. The output value (y_{ij}) of a vector f_j is determined by the formula (3.1). The activation function Sigmoid(x) is determined by the formula (3.2).

$$y_{ij} = Sigmoid(w_i * f_j), \tag{3.1}$$

$$Sigmoid(w_i * f_j) = \frac{1}{1 + e^{-(w_i * f_j)}}. \tag{3.2}$$

The components of a root, node, and leaf are illustrated by Figure 1.

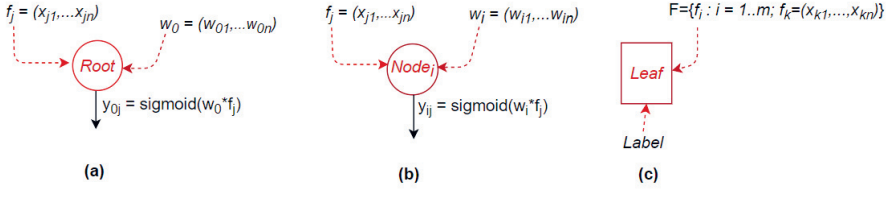

Figure 1: Illustrate components of Root **(a)**, Node **(b)** and Leaf **(c)** on Nested KD-Tree

Figure 1 (a) describes a *Root*, weight vector w_0 is stored at *Root*; vector f_j is inserted into the Nested KD-Tree; y_{0j} is output value of f_j at Root. Figure 1 (b) describes $Node_i$, weight vector w_i is stored at $Node_i$, y_{ij} is output value of f_j at $Node_i$. Figure 1 (c) describes a *Leaf*. Each *Leaf* contains a set of image vectors F and is assigned a label.

3.2. Principles of building Nested KD-Tree structure

The KD-Tree structure is illustrated by Figure 2.

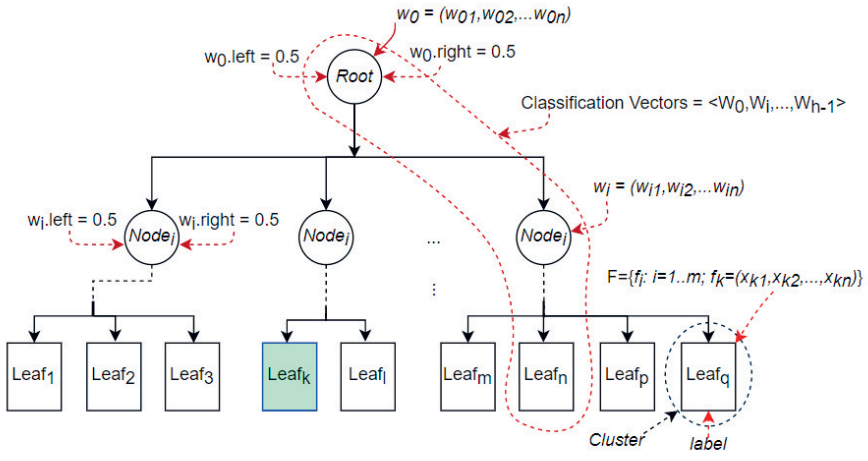

Figure 2: Illustration of KD-Tree structure

The image classification process based on Nested KD-Tree is performed many times. At each layer on Nested KD-Tree is performed once to classify the input image. Therefore, this process is considered as image classification according to a multi-layer model. A Nested KD-Tree is built in two phase:

Phase 1: Building a balanced multi-branched tree (KD-Tree) to perform image classification from root to leaf. The original KD-Tree is a binary tree. In the work [27], the authors published the results of a content-based image retrieval system based on the binary KD-Tree tree. In this paper, the KD-Tree is improved to a balanced multi-branch tree. Then, some leaves are grown into SubTree to form a Nested KD-Tree. Nested KD-Tree is an extension of the KD-Tree structure. Thus, image classification based on KD-Tree structure is perform with the following steps:

Step 1: Initializing height of tree (h), a maximum number of branches (n) at each $Node_i$. Then, a maximum number of leaves is n^h

Step 2: Initializing set of random weight vectors $W =< w_0, \ldots, w_{h-1} >$, where $w_i = (w_{i1}, w_{i2}, \ldots, w_{in})$ is stored at $Node_i$ of the data set layer.

Step 3: At each $Node_i$, initializing thresholds are $w_i.left = 0.5$, $w_i.right = = 0.5$. So, KD-Tree is balanced.

Step 4: Output value y_{ij} of f_j vector at $Node_i$ is calculated by formula (3.1). The direction of f_j vector for next branch is determined with following rules:

a) If $y_{ij} > w_i.right$ then a right sub-branch is created;

update $w_i.right = y_{ij}$

b) If $y_{ij} < w_i.left$ then a left sub-branch is created;

update $w_i.left = y_{ij}$

c) If $w_i.left <= y_{ij}$¡ and $y_{ij} <= w_i.right$ then path for f_j is determined the shortest distance (d_{min}) from position y_{ij} to values in $[w_i.left, w_i.right]$.

Step 5: **Step 3** and **Step 4** are repeated until $leaf_k$ is reached. Then f_j vector is inserted into $leaf_k$

Phase 2: Growing a leaf of the KD-Tree into SubTree.

KD-Tree structure is built by the image classification method from root to leaf. However, some leaves contain many vectors. They can be able to build SubTree. The SubTree is illustrated in Figure 3 as a balanced multi-branch KD-Tree. The purpose of the SubTree is to create a smoother classification process and better classification performance. The conditions for building a SubTree at $Leaf_k$ are (Table 1):

1) A number of elements in $Leaf_k$ is greater than the M threshold.
2) A number of classes in $Leaf_k$ is greater than the N threshold.
3) Height of SubTree is h and maximum number of branches is n.

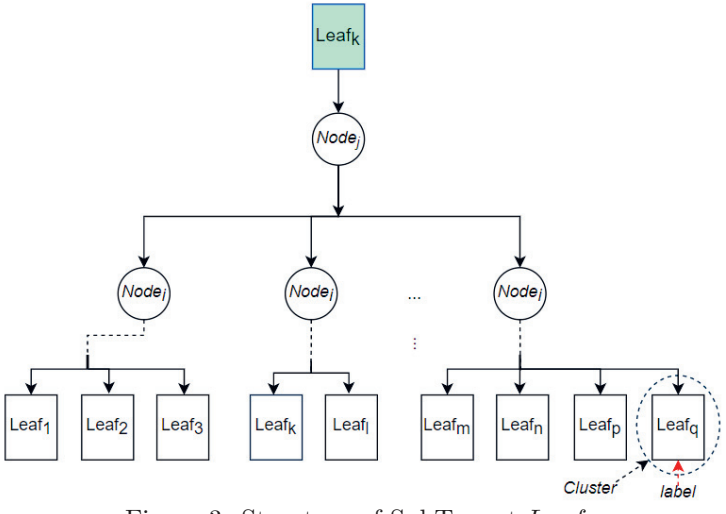

Figure 3: Structure of SubTree at $Leaf_k$

A Nested KD-Tree is illustrated in Figure 4. The Nested KD-Tree includes KD-Tree and SubTree. The **Algorithm 1** and Figure 4 are illustrative of a node. The conditions for building SubTrees are M, N values at $Leaf_k$. The threshold values M, N depend on each experimental data set. So, the threshold values M, N are presented in Table 1.

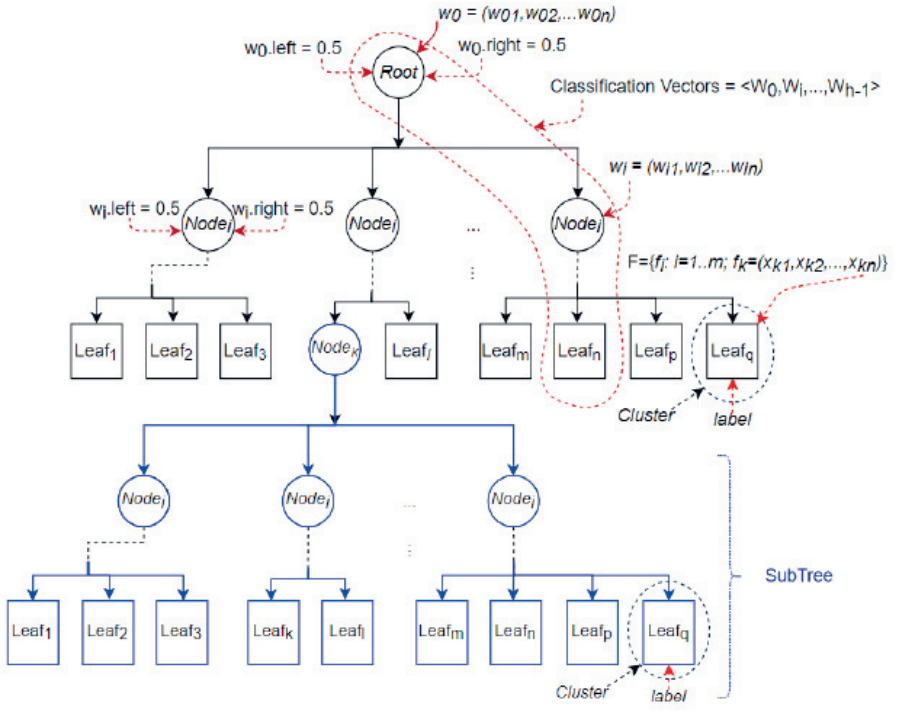

Figure 4: Structure of Nested KD-Tree

In Figure 4, the value of left and right are only illustrative of a node type. All other nodes are also initialized with left and right weights (excluding leaf). In **Algorithm 1**, the initialization value is epsilon that is equal to 0.5.

Data set	Conditions for thresholds M, N
COREL	M > 200 and N > 2
Wang	M > 250 and N > 2
Caltech101	M > 400 and N > 2

Table 1: The results of threshold values M, N on experimental image data sets

3.3. Nested KD-Tree construction algorithm

Nested KD-Tree construction **Algorithm 1 (CNKDT)** is divided into two steps: Step 1 builds a balanced multi-branch KD-Tree; Step 2 searches leaves that meet the conditions in Subsection 3.2 to build Subtrees.

Algorithm 1: Nested KD-Tree tree construction algorithm -**CNKDT**

Input:Vector set $F = \{f_i : f_i = (x_{i0}, x_{i1}, \ldots, x_{in}); i = 1...k\}$;
Initialized set of weights $W_{kt} = \{W_i : W_i = (x_{i0}, x_{i1}, \ldots, x_{in}); i = 0, \ldots, h-1\}$; // the set W_{kt} had initialized to store in all the Nodes of the KD-Tree
Output: NKD-Tree
Function: **CNKDT** (F, W_{kt}, h, n)
begin
  Step 1: Create KD-Tree
  Int h, n; NKD-Tree = ϕ; // h is height, n is number of branch of KD-Tree
  for *(int* $i = 0; i \leq h-1; i++$*)* **do**
    W_i = ***GenerateInitWeight***(); // initialize weight vector
    $W_i.left = Epsilon; W_i.right = Epsilon$; //The value of epsilon is 0.5
  end
  foreach *(*$f_i \in F$*)* **do**
    if *(NKD-Tree =* ϕ*)* **then**
      CNKDT $(f_1, W_i, h, 1)$;// Tree has one branch, height is h
    end
    else
      for *(int* $j = 0; j <= h-1; j++$*)* **do**
        if *(*$Sigmoid(f_i, w_j) < w_j.left$*)* **then**
          CreateLeftBranch();
          $w_j.left = Sigmoid(f_i, w_j)$;
        end
        if *(*$Sigmoid(f_i, w_j) > w_j.right$*)* **then**
          CreateRightBranch();
          $w_j.right = Sigmoid(f_i, w_j)$;
        end
        if *(*$w_j.left < Sigmoid(f_i, w_j))$ *and* $Sigmoid(f_i, w_j) < w_j.right$*)* **then**
          SelectBestBranch(); // Branch has the shortest distance
        end
      end
    end
    if *(*$Node_i.level = h-1$*))* **then**
      if *(*$Leaf_k.parent = Node_i$*)* **then**
        foreach *(*$f_j \in Leaf_k$*)* **do**
          if *(*$f_i.label = f_j.label$*)* **then**
            Insert$(f_i, Leaf_k)$ // if $f_i.label$ is the same as $f_j.label$
          end
          else
            CreateLeaf $(Leaf_{k+1})$;
            Insert $(f_i, Leaf_{k+1})$;
          end
        end
      end
    end
    Step 2: Create SubTree at $Leaf_k$
    Double M = 200, N = 2; // M, N depend on data sets
    foreach *(*$Leaf_k \in SetofLeaf$*)* **do**
      m = CountVector $(f, Leaf_k)$; // Count vector in $Leaf_k$
      n = CountLabel $(label, Leaf_k)$; // Count label in $Leaf_k$
      if *(*$m >= M$ *and* $n >= N$*)* **then**
        CNKDT (F_k, W_{kt}, h, n);
      end
    end
  end
  Return NKD-Tree;
end

Proposition 1. *The complexity of* **CNKDT** *algorithm is* $O((m + M) * h)$*;* m*,* M *are the number of elements to build* KD-Tree *and* SubTree*.* M *is less than* m *and the largest of* M *is* m*;* $(M + m)$ *is assigned* n*.* h *is the height of* KD-Tree *and* h *is a constant. So, the complexity of* **CNKDT** *algorithm is* $O(n)$*.*

Proof. The **CNKDT** algorithm browsing from root to leaf, since KD-Tree is a balanced tree, the **CNKDT** algorithm browsing the height h of the tree (h is a constant). The **CNKDT** algorithm inserts m elements into the KD-Tree and M elements into the SubTree. In the worst case, M is equal to m ($2m = n$). Therefore, the complexity of **CNKDT** algorithm is $O(n)$. ■

3.4. Assigning a label to leaf

After the KD-Tree structure is built, each $leaf_i$ stores a set of feature vectors that belongs to many different classes. Therefore, each $leaf_i$ needs to be assigned a label according to a maximum frequency of occurrence. An optimal method to assign a label to $leaf_i$ is to create a matrix with following steps:

Step 1: Each $leaf_i$ is counted as the frequency of occurrence of all labels in the image data set. Labels do not appear in $leaf_i$ which assigned of frequency of occurrence as 0.

Step 2: The occurrence frequency of $label_i$ in $leaf_j$ is presented by a matrix $MT(x, y) : x = NumberofLabel; y = NumberofLeaf$

Step 3: A maximum value of the matrix $MT(x, y)$ is $VT_{i,j}$ at row_i and $column_j$. Assign $label_i$ to $leaf_j$ is $Value(VT_{i,j})$

Step 4: All values in row i, column j are converted to 0.

Step 5: Repeat step 3, step 4 until all labels are assigned to all *leaves*.

The result of assigning a label to a leaf will create the classifiers. Image classification performance is calculated by the number of vectors with the same class label at a leaf that is divided by the total number of vectors with the same class label in the original data set. Therefore, classification performance on Nested KD-Tree is calculated by the formula

$$P_i = AVG\left(\frac{\sum(f_{ki})}{\sum(f_{mi})}\right), \tag{3.3}$$

where $\sum(f_{ki})$ is a number of vectors with the same label belonging to $leaf_i$, $\sum(f_{kj})$ is the number of vectors with the same label in the original data set.

After assigning a label to leaf and calculating classification performance on the Nested KD-Tree. A process of weight adjustment is called a weight-training and is described in Subsection 3.5.

3.5. Weight training process

Algorithm 2: Weight training algorithm - **TNKDT**

Input: Set of initialized weights
Output: Set of training weights
Function: TNKDT $(InitWeight, Ebol_i)$
begin
  $Weight$ = *InitWeight()*; //initialize random weights
    repeat
      CNKDT$(Ebol_i, InitWeight, h, n)$;
      SetLabel2Leaf (*KD-Tree, ListLabels*); //assigning a labeling to leaf
      // P_i Classifier performance before adjusting weight
      $P_i = SumofVectorRightLabel()/SumofVectorinEbol_i()$;
      Road (f.wrong) = LeafWrong.Road; // Find wrong road of f vector
      Road (f.right) = LeafRight.Road; // Find right road of f vector
      $Get(Node_w)inRoad_{f,w}$; // get $Node_w$ in road of f vector
      //function of weight adjustment at $Node_w$
      $NewWeight =$ **UpdateWeight**$(Node_w, Road_{f,w})$;
      CNKDT $(Ebol_i, NewWeight, h, n)$;
      SetLabel2Leaf (*KD-Tree, ListLabels*);
      // P_j Classifier performance after adjusting weight
      $P_j = SumofVectorRightLabel()/SumofVectorinEbol_i()$;
    until $(P_j < P_i)$;
  $Weight = NewWeight$;
  //Training weight in SubTree
  foreach *(SubTree)* **do**
    F = Find$(f_j$ **in** *SubTree*$)$;
    WeightSub = TNKDT (*InitWeight, SubTree.F*);
  end
  $Weight = NewWeight \cup WeightSub$;
  Return *Weight*;
end

A weight training process of the Nested KD-Tree is divided into two phases: **Phase 1** trains weights of KD-Tree; **Phase 2** trains weights of SubTree at a $leaf_k$. The formula of adjustment w_i weight at $node_i$ (3.4) is determined by the opposite derivative of the Gradient direction with respect to variable w_i and η being a given constant. The weight training process is conducted on each Ebol data set in the experimental image data sets.

$$W_i = W_i - sign(\sigma(W_i * f_i)) * \eta * \frac{\partial(\sigma(W_i * f_i))}{\partial(W_i)} = \tag{3.4}$$
$$= W_i - sign(\sigma(W_i * f_i)) * \eta * \sigma(W_i * f_i) * (1 - \sigma(W_i * f_i)).$$

Proposition 2. *The complexity of* **TNKDT** *algorithm is* $O(p*h*m)$*, where* p *is number of times of adjust weight vector* (p *is a constant*)*,* h *is height of tree* (h *is a constant*)*,* m *is number of elements to build the tree. So, the complexity of* **TNKDT** *algorithm is* $O(m)$*.*

Proof. Weight training process on the Nested KD-Tree is calculated by number of times weight updates to create the tree and assign labels to leaves. Therefore, the complexity of **TNKDT** algorithm is a sum of the complexity of building the tree and assign labels to leaves; in which many labels and number of leaves are constant (p, h are constant). So, the complexity of **TNKDT** algorithm is $O(m)$. ■

In this paper, the Nested KD-Tree has been built and trained weights on a training data sets consisting of COREL, Wang, and Caltech101. The image classification results on data sets are presented in Table 2.

Data set	The number of classes	Classifier performance
COREL	10	83.25%
Wang	80	82.05%
Caltech101	102	75.52%

Table 2: The performance of image classification on Nested KD-Tree

3.6. Retrieval on Nested KD-Tree

The algorithm of retrieving similar images **(SNKDT)** is performed as **Algorithm 3**.

Proposition 3. *The complexity of* **SNKDT** *algorithm is* $O(h*b)$*, where* h *is height of the* Nested KD-Tree (h *is a constant*)*,* b *is a maximum number of branches at* $node_i$ *on the* Nested KD-Tree (b *is a constant*)*;* $h*b$ *is assigned by* c (c *is a constant*)*. So, the complexity of* **SNKDT** *algorithm is* $O(c)$*.*

Proof. The **SNKDT** algorithm is browsing from root to leaf of the Nested KD-Tree. At each $node_i$ of the layer is browsing the list of b sub-branches of $node_i$ (h*b as c is a constant). Therefore, the complexity of **SNKDT** algorithm is $O(c)$. ■

After building the Nested KD-Tree to perform image classification and store image data. Each input image is extracted as a feature vector and retrieved a set of similar images by content based on the Nested KD-Tree. Then, a set of visual words and SPARQL queries are created to retrieve a set of similar images by semantic on ontology. A semantic-based image retrieval model is presented in the Section 4.

Algorithm 3: The content-based image retrieval algorithm–SNKDT

Input: f_j, feature vector of image I_j, Nested KD-Tree (NKD-Tree)
Output: The set of similar images CI
Function: **SNKDT** (f_j, NKD-Tree)
begin
  $CI = \varnothing$;
  for *(int* $i = 0; j < h - 1; i++$*)* **do**
    m = $w_i.left$; n = $w_i.right$; // initialize the left, right value of w_i
    $LeftRightofW_i = [m..n]$; // store the left, right values of w_i
    $ChildofW_i = [Node_m..Node_n]$; //a set of childs of w_i
    $S = Sigmoid(Product(w_i, f_j)$;
    foreach *(k in* $LeftRightofW_i$*)* **do**
      $d(S,k) = TinhKC(S,k)$;//calculate the distance from S to k
      if *(*$d(S,k) = d_{min}$*))* **then**
        // finding branch that has shortest distance
        SNKDT $(f_j, NKD - Tree.Node_k)$;
      end
    end
  end
  if *(*$f_j \in Leaf_k.\{F\}$*)* **then**
    CI = CI $\cup Leaf_k.\{F\}$;
  end
  Return CI;
end

4. A model of the semantic-based image retrieval based on nested KD-Tree and ontology

4.1. Description of experimental data

To demonstrate the effectiveness of the proposed method, in this paper, the data sets have been used for this experiment including COREL, Wang, Caltech101. These data sets are described in Table 3. The COREL data set has 1,000 images and is divided into 10 subjects, each subject has 100 images. The Wang data set has 10,800 images and is divided into 80 subjects, each contains from 100 to 400 images. Caltech101 data set has 9,144 images and consists of pictures of objects belonging to 102 classes. Each image is labeled with a single object. Each class contains from 40 to 800 images. Images have variable sizes, with typical edge lengths of 200–300 pixels.

Data sets	Numbers of images	Numbers of subclasses
COREL	1,000	10
WANG	10,800	80
Caltech101	9,144	102

Table 3: Description of experimental image data sets

The image retrieval problem is conducted with feature vectors. In this paper, each image is extracted into an 81-dimensional feature vector consisting of low-level features describing the image content such as object color, shape, texture, etc. To extract image features, firstly, each input image is segmented into many small regions. Secondly, the methods of extracting color, shape, and texture features are performed by the work [27].

4.2. Creating ontology of image data set

In this paper, an ontology is built by protégé software with follows steps:

(1) preparing sample data for classes,

(2) hierarchy of classes and attributes,

(3) preparing literals for individual classes,

(4) preparing image data with class definitions for the image data set,

(5) building ontology by protégé.

Figure 5 is an example of an N3 language ontology with the Wang data set. Figure 6 is an ontology for the Wang image data set built by protégé.

OntologyWANG.n3

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix sbir: <http://sbir-hcm.vn/>.
<http://sbir-hcm.vn/ANIMAL> <http://sbir-hcm.vn/anoDescription> "a living organism characterized by voluntary movement";
                            <http://sbir-hcm.vn/anoFilename> "../../Dictionary/ANIMAL.txt";
                            <http://sbir-hcm.vn/anoURI> "http://sbir-hcm.vn/ANIMAL";
                            a <http://www.w3.org/2002/07/owl#Class>.
<http://sbir-hcm.vn/anoDescription> a <http://www.w3.org/2002/07/owl#AnnotationProperty>;
                                    <http://www.w3.org/2000/01/rdf-schema#range> <http://www.w3.org/2001/XMLSchema#string>.
<http://sbir-hcm.vn/anoFilename> a <http://www.w3.org/2002/07/owl#AnnotationProperty>;
                                 <http://www.w3.org/2000/01/rdf-schema#range> <http://www.w3.org/2001/XMLSchema#string>.
<http://sbir-hcm.vn/anoURI> a <http://www.w3.org/2002/07/owl#AnnotationProperty>;
                            <http://www.w3.org/2000/01/rdf-schema#range> <http://www.w3.org/2001/XMLSchema#string>.
<http://sbir-hcm.vn/ART_1> <http://sbir-hcm.vn/anoDescription> "the products of human creativity; works of art collectively";
                           <http://sbir-hcm.vn/anoFilename> "../../Dictionary/ART_1.txt";
                           <http://sbir-hcm.vn/anoURI> "http://sbir-hcm.vn/ART_1";
                           a <http://www.w3.org/2002/07/owl#Class>.
<http://sbir-hcm.vn/ART_ANTIQUE> <http://sbir-hcm.vn/anoDescription> "any piece of furniture or decorative object or the like produced in a f
                                 <http://sbir-hcm.vn/anoFilename> "../../Dictionary/ART_ANTIQUE.txt";
                                 <http://sbir-hcm.vn/anoURI> "http://sbir-hcm.vn/ART_ANTIQUE".
<http://sbir-hcm.vn/ART_ANTIQUES> a <http://www.w3.org/2002/07/owl#Class>;
                                  <http://www.w3.org/2000/01/rdf-schema#subClassOf> <http://sbir-hcm.vn/ART_1>.
<http://sbir-hcm.vn/ART_CYBR> <http://sbir-hcm.vn/anoDescription> "a genre of fast-paced science fiction involving oppressive futuristic comp
                              <http://sbir-hcm.vn/anoFilename> "../../Dictionary/ART_CYBR.txt";
```

Figure 5: Illustrating ontology in N3 language

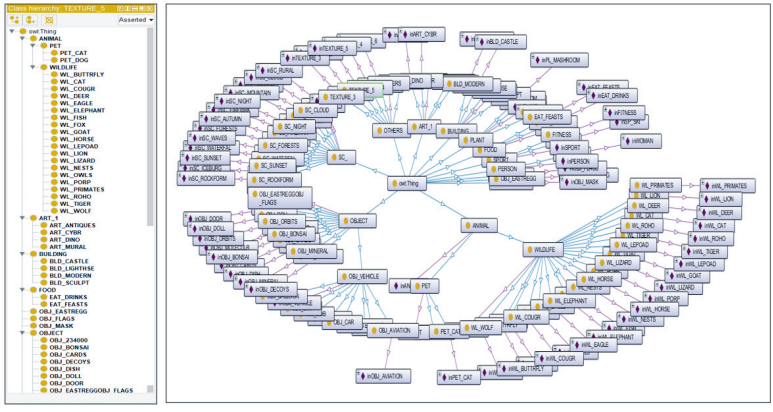

Figure 6: Ontology of Wang image data set built on protégé

After building ontology, a set of similar images by semantic are retrieved by SPARQL query language. An ontology of the data set is built based on the RDF language. RDF is used to name images, describe image attributes and relations [23]. Ontology is used to define the properties and relations of objects and their classes. Ontology is stored in N3 notation.

Ontology is built on triple-language and queried in many different languages. SPARQL is an ontology-based query language and based on the triple. Simultaneously, a SPARQL query is used to get high-level semantics of images on ontology. Therefore, SPARQL is a convenient query language for experiments and the best choice for this paper. In this paper, a set of similar images by content of the input image based on the Nested KD-Tree is extracted. Based on a set of similar images of an input image is retrieved on the Nested KD-Tree. A set of visual words is extracted. Figure 7 illustrates process of building a set of visual words. Figure 8 illustrates the SPARQL query that is created from a set of visual words "Beach", "Peoples" in two ways: "UNION Query" or "AND Query".

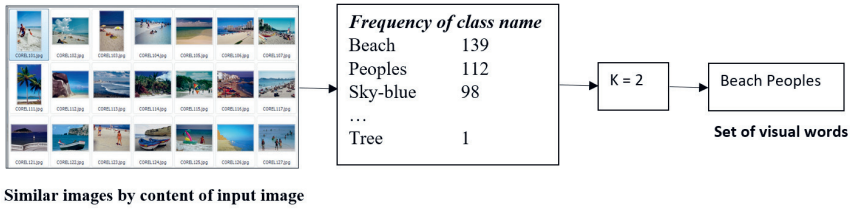

Figure 7: An example of building a set of visual words

The set of visual words is built by taking the frequency of the most occurrences of K class names in a set of similar images by content of input image.

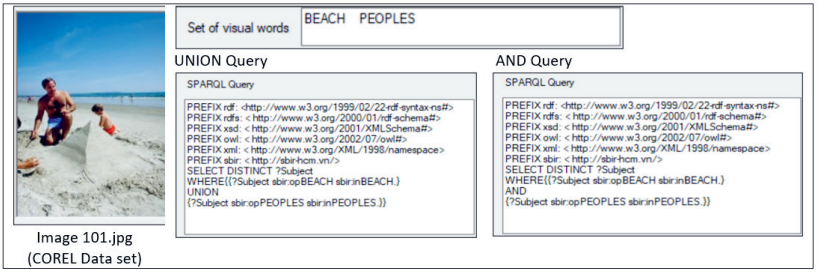

Figure 8: An example of creating a SPARQL query from a set of visual words

4.3. A model of semantic-based image retrieval

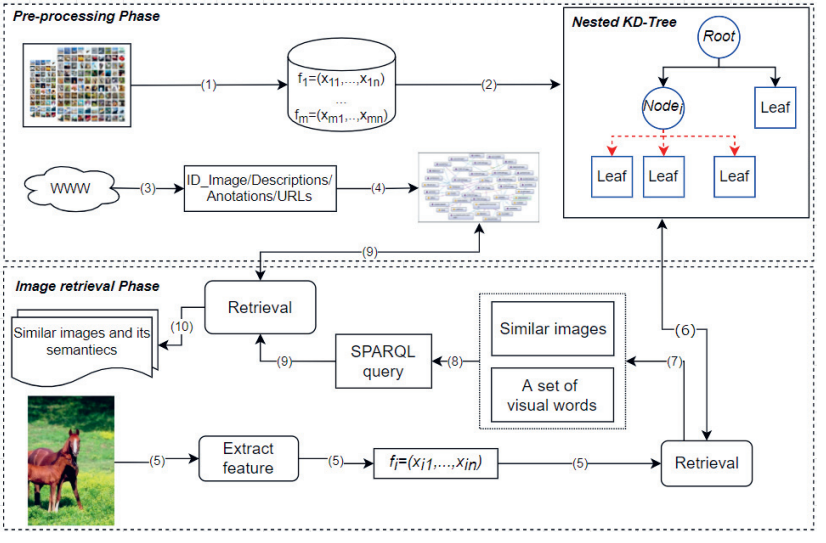

Figure 9: A model of semantic-based image retrieval SB-NKDT

A model of semantic-based image retrieval SB-NKDT consists of two phases:

Preprocessing phase: Extracting the feature vectors, building Nested KD-Tree and ontology include the following steps:

Step 1: Extract feature vectors of image data sets (1).

Step 2: Build Nested KD-Tree by training weight vectors at nodes and storing the image data at leaves (2).

Step 3: Build ontology on protégé with experimental set of images and images from WWW (3), (4).

Retrieval phase: Retrieve a set of similar images based on Nested KD-Tree structure, extract a set of visual words and retrieve a set of similar images by semantics on ontology using SPARQL query.

Step 4: Extract the feature vector of an input image (5) and retrieve a set of similar images by content based on the Nested KD-Tree (6).

Step 5: Create a set of visual words (7) and SPARQL query (8).

Step 6: Retrieve a set of similar images by semantics on ontology (9).

Step 7: Extract a set of similar images by semantics and its semantics (10).

4.4. The semantic-based image retrieval algorithm

When a query image is inserted, a set of similar images by content is retrieved on the Nested KD-Tree. Based on the set of similar images, a set of visual words is extracted. So that, SPARQL query is generated and retrieved a set of similar images by semantics based on ontology. Therefore, an algorithm of semantic-based image retrieval **(SOKDT)** is presented as follows:

Algorithm 4: A semantic-based image retrieval algorithm– **SOKDT**

Input: Visual Word vector of I_i image
Output: A set of similar images (SI) and semantics classification of images I_i (SC)
Function SNKDT (VisualWordVector)
begin
  $SI = \varnothing; SC = \varnothing;$
  $SPARQL(I_i) = CreateSPARQL(VisualWordVector);$
  $(SI, CS) = Query(SPARQL(I_i), \text{Ontology});$
  Return $\{SI, SC\};$
end

5. Experiments

5.1. Experimental environment

The experiment on building Nested KD-Tree and image classification is illustrated by Figure 10. The SB-NKDT system is built on the dotNET Framework 4.5 platform, the *C#* programming language. The graphs are built on MathLab. The SB-NKDT system is performed in two phases: Preprocessing phase and retrieval phase, which are implemented on computers with Intel(R) Core™ i7-5200U processors, CPU 2.70GHz, RAM 16GB, and Windows 10 Professional operating systems. Figure 11 describes the SB-NKDT system for semantic image retrieval. In this paper, three data sets COREL, Wang, and Caltech101 are used for an experiment.

Figure 10 describes a method of building Nested KD-Tree including three sets of images COREL, Wang, Caltech101. Each image data set is built a tree that determines the maximum height and number of branches. P1 is a clas-

sification performance before training weight (Classification Performance P1), P2 is a classification performance after training weight (Classification Performance P2). After creating the KD-Tree (Create KD-Tree) assigning a label to leaf (Set Label); training weights for KD-Tree (Weight Training KD-Tree); creating SubTrees at some leaves (Create SubTree) and training weights for SubTree (Weight Training SubTree).

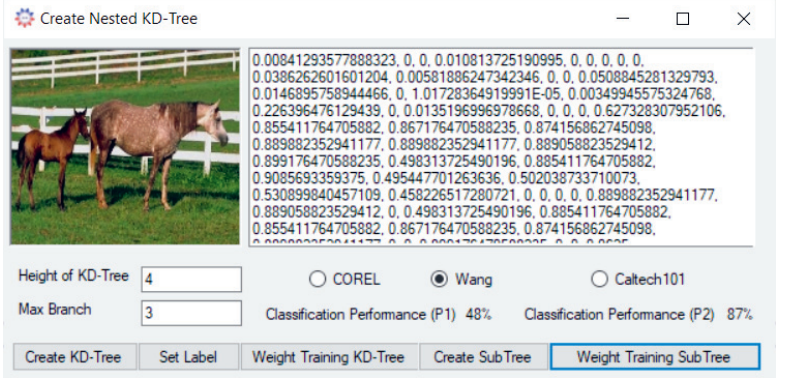

Figure 10: Experiment of construction Nested KD-Tree

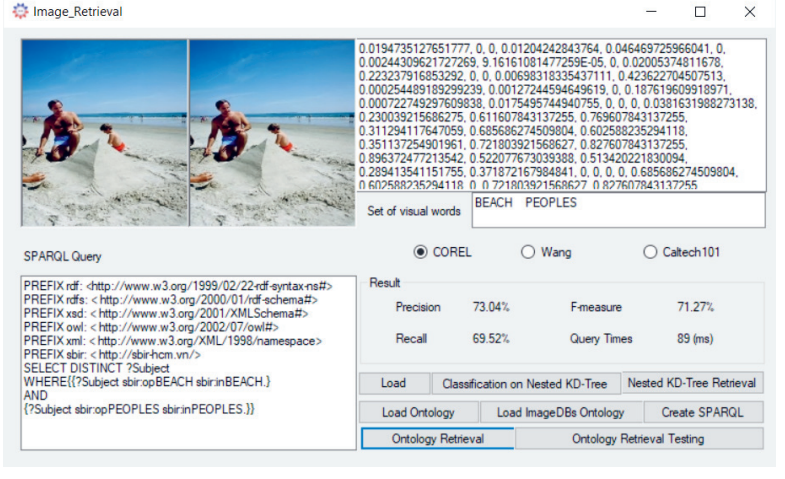

Figure 11: The semantic-based image retrieval system SB-NKDT

Figure 11 describes an experiment of semantic image retrieval of 101.jpg image in COREL data set. After loading an image (Load Image), performing image classification on the Nested KD-Tree (Classification on Nested KD-Tree) and retrieving a set of similar images by content (Nested KD-Tree Retrieval). Then, a SPARQL query is created (Create SPARQL) and a set of similar images

by semantic are retrieved (Ontology Retrieval). The result of the semantic-based image retrieval of 101.jpg, COREL data set is illustrated in Figure 12.

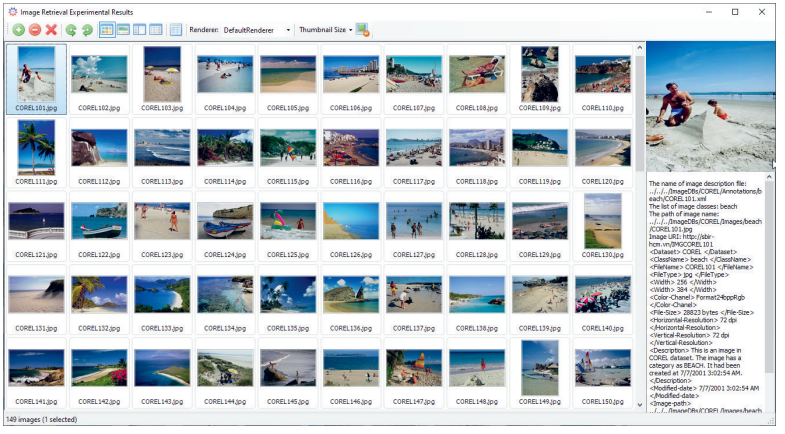

Figure 12: The result of SB-NKDT for 101.jpg image (COREL)

5.2. Experimental results and evaluation

The experimental results of the proposed method are presented in Table 4. Each image data set has a different number of classes, so the height and number of branches on the Nested KD-Tree are different. Therefore, the query performance and retrieval time of the system also differ from the image data sets. The experimental results of the COREL data set, the height of the tree is two, the number of branches is five, so the retrieval time is faster than that of Wang, Caltech101 data sets as a height or number of branches greater than three.

Data set	Average precision	Average recall	Average F-measure	Average query time (ms)
Corel	81.19%	70.83%	75.65%	39
Wang	80.29%	66.28%	72.61%	76
Caltech101	72.55%	61.93%	66.82%	148

Table 4: Image retrieval performance of the proposed method on data sets

The semantic-based image retrieval performance of the proposed method is quite high, because of the following reasons: (1) The Nested KD-Tree is built using the image classification method; (2) The weight training process on Nested KD-Tree is expensive but image classification efficiency is high; (3) The classification results are used for content-based image retrieval, then semantic-based image retrieval based on ontology is performed. Therefore, the results of the proposed method are evaluated as effective.

Figure 13–18 show the curves of Precision-Recall and ROC for the COREL, Wang, Caltech101 data sets. Each curve describes a set of query images, which are retrieved. The graph shows that the precision of the COREL image data set concentrated in the [0.68, 1.0] region, Wang image data set concentrated in the [0.61, 1.0] region, and the precision of the Caltech101 in the region [0.5, 1.0]. The ROC curve graph shows true positive and false positive values according to Recall coverage, the values concentrated on the baseline, the numerous values in the true-positive region than in the false-positive region.

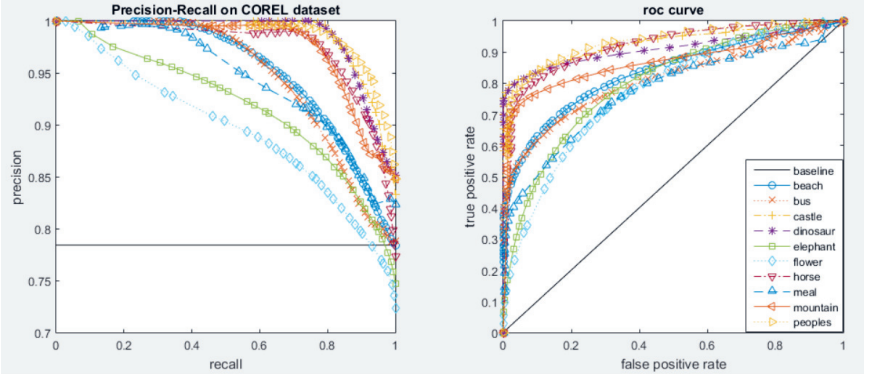

Figure 13: Precision-Recall and ROC curve of COREL data set

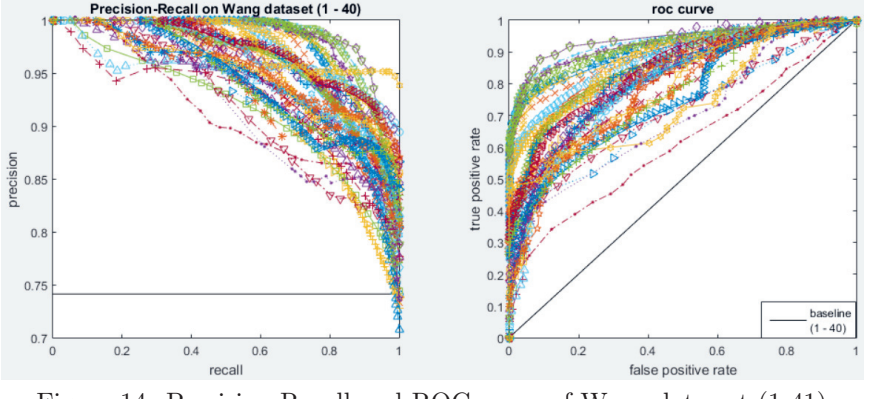

Figure 14: Precision-Recall and ROC curve of Wang data set (1-41)

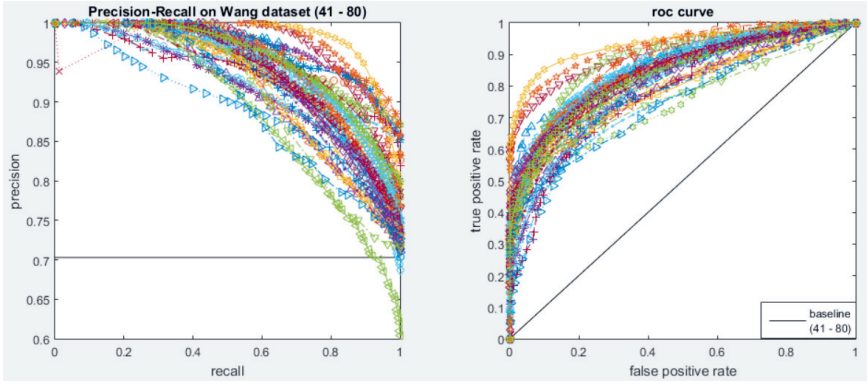

Figure 15: Precision-Recall and ROC curve of Wang data set (41-80)

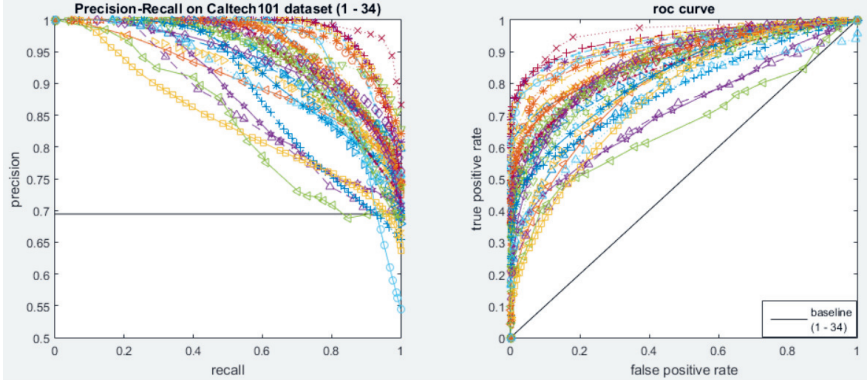

Figure 16: Precision - Recall and ROC curve of Caltech101 (1-34)

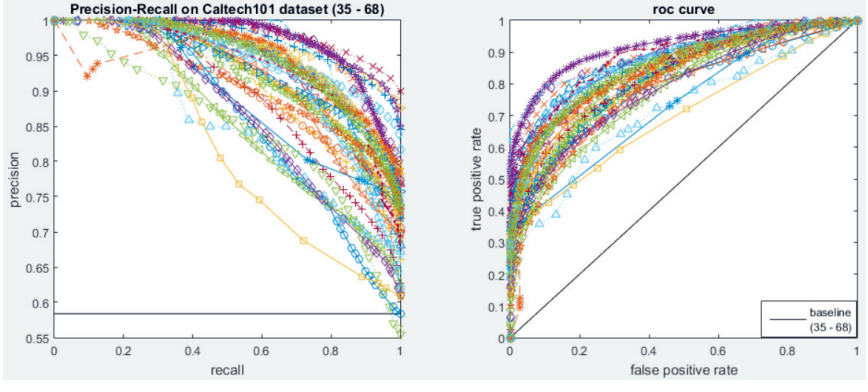

Figure 17: Precision-Recall and ROC curve of Caltech101 (35-68)

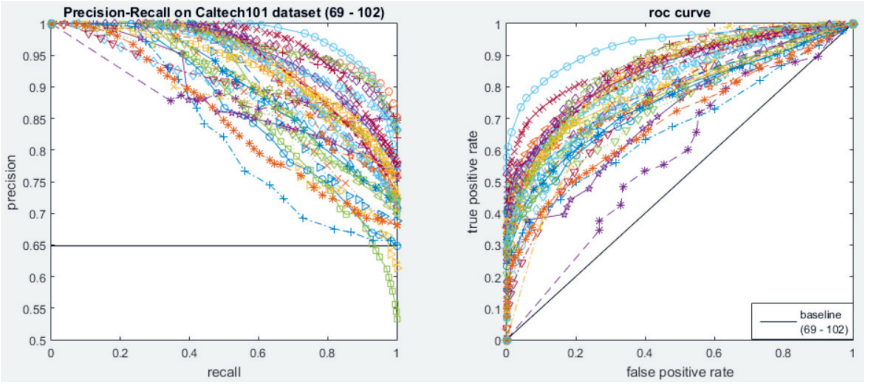

Figure 18: Precision-Recall and ROC curve of Caltech101 (69-102)

The experimental results show the performance of SB-NKDT system on the data sets COREL, Wang, Caltech101 are higher than other works because of the following reasons: **(1)** The SB-NKDT performed image classification based on the Nested KD- Tree structure before retrieving the similar images; **(2)** The Nested KD-Tree is built as a multi-layer classification method for object and weight vectors at the nodes are trained, so the accuracy of image classification is high; **(3)** The combination of Nested KD-Tree multidimensional data structure and ontology have improved the performance of semantic-based image retrieval system.

Method	Data set	Average accuracy
B-SHIFT, 2016 [15]	COREL	72.00%
CBIR-CKDT, 2021 [27]	COREL	76.40%
SB-NKDT	**COREL**	**81.19%**
P. CHHABRA, 2020 [24]	Wang	63.20%
CBIR-CKDT, 2021 [27]	Wang	73.27%
SB-NKDT	**Wang**	**80.29%**
M. Bennet Rajesh, 2020 [25]	Caltech101	63.14%
S. Sathiamoorthy1, 2020 [26]	Caltech101	69.42%
SB-NKDT	**Caltech101**	**72.55%**

Table 5: Comparison of mean average precision (MAP) of methods on data sets

In addition, the precision of works [15], [24], and [25] were lower than the semantic-based image retrieval system SB-NKDT because of the following reasons: (1) The works [15, 24, 25] have not built a data structure for storing image data; (2) The works [15, 24] haven't performed image classification; (3) The works [15, 24, 25] haven't done semantic-based image retrieval system; (4)

The works [15, 24] have not combined many machine learning techniques into the problem to improve the retrieval performance; (5) The work [24] used SIFT feature of eight lengths, the accuracy is 86.2% but the coverage was not high because the author took the top 20 images; (6) The work [25] has combined Co-occurrence of Edges and Valleys with Support Vector Machine classification algorithm but has not performed semantic-based image retrieval. Therefore, SB-NKDT is a combination of a multi-layered data structure for objects with semantic-based image retrieval based on ontology is one of the approaches to improve retrieval performance.

6. Conclusions and development

In this paper, a semantic-based image retrieval system SB-NKDT was built based on Nested KD-Tree structure and ontology. The experiment was performed on COREL, WANG, and Caltech101 image data sets with precision values of 81.19%, 80.29%, 72.55%, respectively. To evaluate the efficiency of the proposed method, the experimental results were compared with other methods on the same image data set. The results show that the proposed method is effective and can be applied to image retrieval systems in different fields. In our new development, we will build a random forest model based on the KD-Tree to conduct a performance of image classification and apply to a semantic-based image retrieval system.

Acknowledgment. The authors would like to thank the Faculty of Information Technology, University of Science - Hue University for their professional advice for this study. We would also like to thank HCMC University of Food Industry, University of Education, and research group SBIR HCM, which are sponsors of this research. We also thank anonymous reviewers for their helpful comments on this article.

References

[1] **Patrizio, A.,** Data center explorer:
https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html
[2] Corel 1k Database: http://wang.ist.psu.edu/docs/related/
[3] Wang Database: http://wang.ist.psu.edu/docs/home.shtml
[4] Caltech101:
https://www.vision.caltech.edu/datasets

[5] **Agarap, A.F.,** An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification, arXiv preprint (2017). https://arxiv.org/pdf/1712.03541.pdf

[6] **Gao, Dan, Y.-X. Zhang and Y.-H. Zhao,** Support vector machines and kd-tree for separating quasars from large survey data bases, *Monthly Notices of the Royal Astronomical Society*, **386(3)** (2008), 1417–1425.

[7] **Hou, W. et al.,** An advanced k nearest neighbor classification algorithm based on KD-tree, in: *2018 IEEE International Conference of Safety Produce Informatization* (*IICSPI*), 2018.

[8] **Khotimah, W.N., et al.,** Tuna fish classification using decision tree algorithm and image processing method, in: *2015 International Conference on Computer Control Informatics and its Applications* (*IC3INA*), 2015.

[9] **Setiawan, D.,** Mosque search with Kd-Tree Nearest Neighbor case studies and field district of johor, 2020.

[10] **Harsányi, K., A. Kiss, A. Majdik and T. Szirányi,** A hybrid CNN approach for single image depth estimation: A case study, in: *International Conference on Multimedia and Network Information System - Springer Champp.*, (2018), 372–381.

[11] **Gombos, G., G. Rácz, and A. Kiss,** Spar (k) ql: SPARQL evaluation method on Spark GraphX, in: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops* (*FiCloudW*), (2016), pp. 188–193.

[12] **Gombos, G. and A. Kiss,** P-Spar (k) ql: SPARQL evaluation method on Spark GraphX with parallel query plan, in: *2017 IEEE 5th International Conference on Future Internet of Things and Cloud* (*FiCloud*), (2017), pp. 212–219.

[13] **Zhang, Y. et al.,** Fast face sketch synthesis via kd-tree search, in: *European Conference on Computer Vision - Springer Cham*, 2016.

[14] **Cevikalp, H., E. Merve and O. Savas,** Large-scale image retrieval using transductive support vector machines, *Computer Vision and Image Understanding*, **173** (2018), 2–12.

[15] **Jiu, M. and H. Sahbi,** Nonlinear deep kernel learning for image annotation, *IEEE Transactions on Image Processing*, **26(4)** (2017), 1820–1832.

[16] **Tzelepi, M. and A. Tefas,** Deep convolutional learning for content based image retrieval, *Neurocomputing*, **275** (2018), 2467–2478.

[17] **Vijayarajan, V., M. Dinakaran, P. Tejaswin and M. Lohani,** A generic framework for ontology-based information retrieval and image retrieval in web data, *Human-centric Computing and Information Sciences*, **6(18)** (2016), 1–30.

[18] **Asim, M.N., et al.,** The use of ontology in retrieval: a study on textual multilingual and multimedia retrieval, *IEEE Access*, **7** (2019), 21662–21686.

[19] **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.
[20] **Bentley, J.L.,** Multidimensional binary search trees used for associative searching, *Communications of the ACM*, **18(9)** (1975), 509–517.
[21] **Ningning, C., X. Zhong and L. Li,** Research on optimized R-Tree high-dimensional indexing method based on video features, in: *IEEE 2nd International Conference on Cloud Computing and Big Data Analysis* (*ICCCBDA*) *Chengdu*, 2017, pp. 93–97.
[22] **Gombos, G., J.M. Szailai-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.
[23] **Gombos, G.,** Semantic data evaluation on federated and distributed systems, 2018.
[24] **Chhabra, P., N.K. Garg and M. Kumar,** Content-based image retrieval system using ORB and SIFT features, *Neural Computing and Applications*, **32(7)** (2020) 2725–2733.
[25] **Das, R., S. Thepade and S. Ghosh,** Novel feature extraction technique for content-based image recognition with query classification, *International Journal of Computational Vision and Robotics*, **7(1-2)** (2017), 123–147.
[26] **Jiu, M. and H. Sahbi,** Nonlinear deep kernel learning for image annotation, *IEEE Transactions on Image Processing*, **26(4)** (2017), 1820–1832.
[27] **Nguyen, T.D., T.T. Van and M.T. Le,** Image classification using Kd-Tree for image retrieval problem, *Journal of Research and Development on Information and Communication Technology*, **1** (2021), 33–44.

Nguyen Thi Dinh
HCMC University of Food Industry
HoChiMinh city, Vietnam
University of Science, Hue University
Hue, Vietnam
dinhnt@hufi.edu.vn

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

Thanh Manh Le
University of Science
Hue University
Hue, Vietnam
lmthanh@hueuni.edu.vn