Proceedings of the Federated Conference on Computer Science and Information Systems pp. 613–621
DOI: 10.15439/2017F457
ISSN 2300-5963 ACSIS, Vol. 11

New Content Based Image Retrieval database structure using Query by Approximate Shapes

Stanisław Deniziak
Kielce University of Technology
al. Tysiaclecia Panstwa Polskiego 7, 25-314 Kielce, Poland
Email: s.deniziak@tu.kielce.pl

Tomasz Michno
Kielce University of Technology
al. Tysiaclecia Panstwa Polskiego 7, 25-314 Kielce, Poland
Email: t.michno@tu.kielce.pl

***Abstract*—The image retrieval from multimedia databases is a very challenging problem nowadays. Not only it requires the proper query form, but also efficient methods of data storage. The problem is important, because nowadays there are many different systems which needs image retrieval. As an example web searching engines may be given, which had to store a very huge amount of images and needs fast image retrieval of chosen ones. Also social media portals increasingly face the same requirements. This paper presents a new Content Based Image Retrieval database. It is based on new object representation which is based on approximation of objects by a set of shapes. The structure of the database is designed in order to reduce the number of comparisons using a tree structure. The main advantages of the proposed solution are: easy queries for users, faster image retrieval and ability to parallelize queries.**

I. Introduction

THE image retrieval from multimedia databases is a very challenging problem nowadays. Not only it requires the proper query form, but also efficient methods of data storage. The problem is important, because nowadays there are many different systems which needs image retrieval. As an example web searching engines may be given, which had to store a very huge amount of images and needs fast image retrieval of chosen ones by users. Also social media portals increasingly face the same requirements. Other examples may be monitoring systems which have to detect objects and then find them in the database in order to e.g. check if they are undesirable and additional actions have to be performed. Also number plate or face recognition systems have to perform database queries based on the data present in the image. There are also some attempts of sketch-based CBIR usage in conjunction with gesture recognition [1].

This paper presents a new Content Based Image Retrieval database which is based on our previous researches [2], [3], [4], [5]. The main idea of the Query by Approximate Shapes algorithm is based on a new object representation which consists of approximation of objects by a set of shapes. There are six base shapes defined [2], called primitives. Each primitive may contain not only information about its type, but also parameters which describes each single shape occurrence (e.g. a slope for lines or an angle for arches) and relations to other shapes. In order to store all information about the object, a graph of shapes is proposed. The structure of the database which stores such graphs in order to be efficient, have to reduce the number of comparisons, thus a tree structure is proposed. We defined two types of tree nodes:

- common nodes which are used to organize the data
- data nodes which only stores graphs

The main advantages of proposed solution are: easy queries for users (both images and graphs drawn by a human are accepted), faster retrieval of results thanks to the hierarchical structure and storing similar graphs in congruent nodes, ability to parallelize operations during query process and possibility to use different implementations of the database on the lower level (e.g. using NoSQL data stores, relational databases or containers).

The paper is organized as follows: the Section II presents related works in the area of Image Retrieval and database structures. The Section III contains our motivation and assumptions which are made for the system. The Section IV describes the object representation used in the database. The Section V is dedicated to the database structure and contains descriptions of inner structure, operations on nodes and queries. The Section VI shows initial experimental results. The Section VII presents the plans for the future works and conclusions. The last section contains bibliography.

II. Related works

The multimedia database Image Retrieval algorithms may be assigned to three types of algorithms:

- based on textual descriptions, most often keywords - Keywords Based Image Retrieval (KBIR) algorithms
- based on semantic information extracted from the image - Semantic Based Image Retrieval (SBIR) algorithms
- based on information which is present in the image - Content Based Image Retrieval (CBIR) algorithms

The Keyword Based Image Retrieval algorithms use textual annotations in order to describe the whole image or their parts. Most often descriptions are made by humans and the precision of keywords is limited to the knowledge and perception of a person [3]. For objects which are well known it is easy to represent them by annotations and the results of retrieval are very satisfactory. For example a car object may be named very precisely by the brand, model name, version, production year and color. When the object is not well known or it is not easy to describe it precisely by keywords, the results

may be imprecise. This is due to the fact that annotations are very subjective and different person may use different words as keywords for the same objects [6], [7]. For example a landscape with trees and water may be annotated by one person as a forest and a river, but by another one as trees and a lake. The third person contrary may use the name of place where the photo was taken. In this situation the results of a query may be imprecise and unsatisfactory for the user. Another disadvantage of the KBIR approach is that it is hard to automatically add keywords without human interaction.

The Semantic Based Image Retrieval algorithms are similar to Keyword Based Image Retrieval algorithms because they use also words to perform queries. However, contrary to them, they allows users to write queries as phrases which are more natural form for them. Such different query interface is used in order to overcome the so called 'semantic gap' which is a difference between what a human could describe and what is present in the image [7], [8]. After defining by an user, the phrases are mapped onto so called semantic features which are correlated with the content of the image [9]. The use of semantic based textual approach is more comfortable and easy for users but still if they does not have the full knowledge about searched images the results may be insufficient. There are also approaches which uses graphical queries which are then transformed into textual description. One of the most interesting research is [10] which uses a sketch as a query, then extracts textual annotations - semantic features and then finds 3D models of objects which are described by similar or the same set of features.

The Content Based Image Retrieval algorithms use information present in the image to perform queries [3]. In this area two types of algorithms could be distinguished: low-level and high-level [2]. The first type of algorithms are based on extraction of features for the whole image. There may be statistical image features used, e.g. a normalized color histogram [11]. Another methods may be a difference moment and entropy [12], a spatial domain image representation [13] or a bag of words histogram [14]. There are also approaches which use different MPEG-7 descriptors, e.g. shape and texture descriptors [15]. Since the features describes the whole image, the low-level CBIR algorithms provide very satisfactory results when a query is performed in order to find similar images. However, when an user would like to obtain images with the same object but with different backgrounds, the low level algorithms are not efficient and the results may be insufficient. The high-level CBIR algorithms are more suitable for that situations. Their main idea is to separate objects from the background and other parts of the image. Most often the region extraction method is used [2] which is based on gathering similar groups of pixels into uniform areas which are then transformed into a graph, storing the mutual relations between nodes. In order to extract regions, methods based on e.g. color thresholds, moment-based local operators [16] or fuzzy patterns recognition [17] may be used. The query process is strictly based on searching subgraphs between a graph which was extracted from the stored image and a graphs stored in the database. The main disadvantage of the region-based algorithms is the need of query image which have to store many details. If an user does not have a proper one, it must be prepared which may require drawing skills.

There are also CBIR algorithms which allows performing queries without having the full knowledge about the searched objects. There are algorithms which are low-level e.g. [18]. The approach presented by authors is based on human drawn sketches which are then transformed into lower resolution images and compared with sketches in the database using edge detection techniques. The method provides good results, but is oriented on finding similar paintings, which is not sufficient for querying by objects. [19]. Other researches uses global contour map and salient contour map in order to extract objects and compare them with images in the database. There are also researches which use additionally relevant feedback (e.g. SIFT algotihm) an re-ranking to improve the results precision [20]. In our previous researches [4], [2], [5], [3] we proposed a high level algorithm which is based on decomposing object into its approximated by predefined shapes representation (Query by Approximate Shape). The shapes are used for creating a graph which is then compared with other graphs stored in the database. In this paper we describe the database structure based on the Query by Approximate Shape method.

All images or objects representations have to be stored in an efficient way. Most often a structure based on cells or trees are used [21]. One of the most interesting approaches is [22] which is based on a tree storing cells. The similar images are stored in the same cells, but when the similarity between images is below the determined threshold, another cell is created (a process in the paper called a mitosis). In [4] we proposed the first attempts for the database structure for our object representation which was based on Scalable Distibuted Two-Layer Data Structures. The approach provided good results but was prone to rapid tree height increase. Moreover we would like to prepare the database structure which would be more universal and allow to use different data structures types in the lower implementation level (e.g. SD2DS, but also data containers or relational databases). This would increase the number of possible applications e.g. to use our database on devices with very small resources. Our database structure in some parts is based on the same ideas as [22] (e.g. storing similar data in the same part of the tree), but we use different inner structure and different methods of object representation and querying.

III. Motivation

The multimedia database structure as well as performing queries is a very broad problem. Not only querying by images may be complicated, but also storing a very huge amount of data, because images and their representation used for comparisons had to be kept. Some attempts for the database structure were presented in [4], but we would like to obtain more universal structure which could be used with different implementations for specific cases. This would allow implementing the Query by Approximate Shape database using for

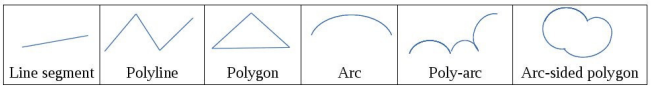

Fig. 1. The primitives used to describe objects [2].

Fig. 2. The attributes used for line segments (a) and arches (b).

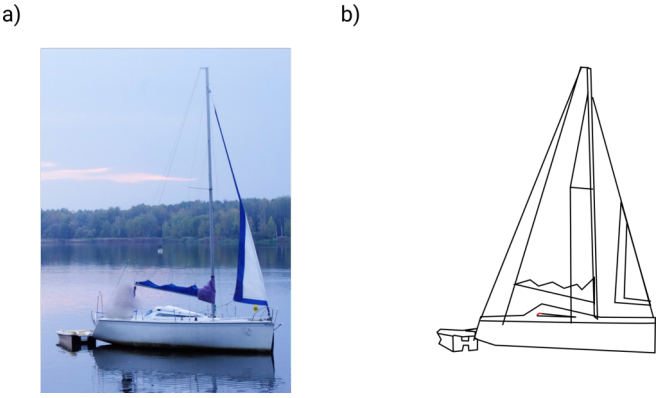

Fig. 3. The example sailboat object representation: a) an image, b) an object drawn with lines (black color) and arches (red color).

example Scalable Distibuted Two-Layer Data Structures for data servers, relational databases (e.g. MySQL) for desktops and containers like vectors or lists for tablets. As a result of our research we would like to obtain the system which fulfill the following requirements:

- ability to perform queries using graphs extracted from a query image or graphs drawn by a human (without need of drawing skills)
- easy addition of a new object without rebuilding or training the whole structure
- fast access to the data
- the database structure which allows parallelization in order to improve the efficiency of queries (and for example use different machines to process searching in different subtrees)
- the ability to set the minimum similarity between objects which is needed to add them to the result set
- higher level of database structure, which allows different implementations e.g. using NoSQL data stores, relational databases, containers etc.

IV. Object graph

The main idea of our algorithm is based on representing objects by shapes. Each object can be described using approximation by a set of geometrical shapes. The example representation was shown in a Fig. 3. In our previous research [2] we proposed to use following shapes, called primitives: line segments, polylines, polygons, arches, polyarches and arc-sided polygons (polygons constructed from arches) (Fig. 1). Each shape is defined by its type and its attributes:

- line segments are defined by the angle of its slope and optionally by their length (Fig. 2 a)
- polylines and polygons are defined by the number of line segments from which they are built and attributes of each of them
- arches are defined by the angle of the arc and optionally by its radius or diameter (Fig. 2 b)
- polyarches and arc-sided polygons are defined similarly like polylines and polygons by the number of arches from which they are built and attributes of each of them

The proposed object representation allows using two types of queries: a manually hand drawn sketches (e.g. using predefined shapes, like in vector graphics) or to automatically extract shapes from the query image. Thanks to that, the database human interface may be more universal and more suitable for people without drawing skills. Moreover automatically generated queries by e.g. by monitoring systems would be also easily performed. The manually drawn sketches may be prepared using a set of predefined shapes and they may be very schematic without many details. Due to that fact they may be prepared fastly and easilty without high drawing skills. The automatically extraction of shapes from images is based on line segments and arches detection e.g. using Line Segment Detector algorithm and Circular Hough Transform[2]. Firstly all lines and arches are detected and then if it is possible, they are joined constructing more complex shapes like polylines, polygons, poly-arches and arc-sided polygons. The extraction procedure is described with more details in [2].

When representing an objects by set of shapes, there may be also needed an information how they are positioned to each others or which of them are connected. In order to store such an information a graph may be used [2].

Each graph may contain not also a description used to comparisons with other graphs, but also some metadata which is useful when returning results. For example metadata may contain the image name, its description and image file path or image binary pixels data.

In order to compare graphs with each others, a coefficient called $similarity$ is used which describes how similar two graphs are. The values of $similarity$ are between 0 and 1. The value 1 means that the graphs are the same, 0 that they are completely different. When generating a results set, some minimal threshold should be used to mark which graphs are similar and should be taken into account.

V. The database structure

The database structure is based on a tree which is build from different types of nodes. There are two types of elements:

- common (graphs) nodes which are used to organize the data
- data nodes which only stores graphs

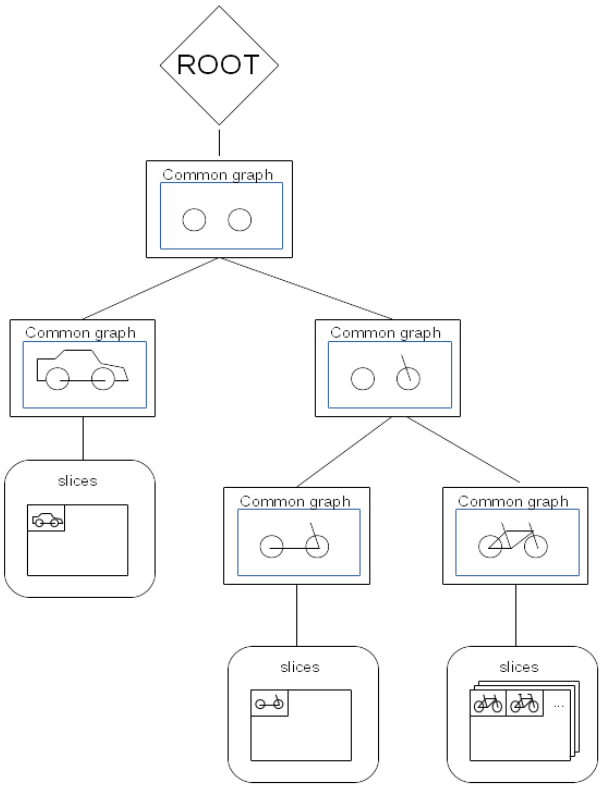

Fig. 4. The overview of the tree database structure

The database structure is partially based on our previous research [4]. In order to improve the time of comparisons, similar graphs are grouped into the same node or nodes, called data nodes. Moreover this could reduce greatly the height of the tree which was a problem in [4].

The root node is used as an entry point to the tree and does not contain a graph thus when compared with the query graph, it always returns the highest similarity. It may contain many children which are then compared on the next level of query.

The common graphs nodes are gathering similar parts of their children nodes as a graph. Therefore when the query graph is compared and does not have enough similarity, there is no need to compare other levels of tree and the whole subtree can be abandoned, which highly decrease the time of the query.

The data nodes does not have any children but contains one or more object graphs (which are correlated to images using metadata). They are used to gather similar graphs in the same node of the tree. Graphs are stored in a so called slice which is strictly a vector of graphs. The first vector element is the most similar graph to the common graph stored in the parent common node. Next graphs stored in the vector are compared to the first element and sorted from the most similar to the least. In order to improve the query time and to allow using parallelism or different machines to store some parts of graphs

Algorithm 1 Splitting a vector of graphs when the maximum size is reached

Ensure: T_s - maximum number of graphs in the slice; dh - data node, vec - vector with graphs
$vSize \leftarrow size(vec)$;
if $vSize > T_s$ **then**
  create new vector $vec2$;
  copy into $vec2$ graphs from $vec[T_s]$ to $vec[vSize]$;
  remove $vec[T_s]$ to $vec[vSize]$;
  add $vec2$ to dh;
end if

and images, there may be more than one slices of graphs stored in a one data node. Therefore, the first slice should store the most similar graph to the parent common graph and when a desired maximum number of graphs in a vector is achieved, the next vector slice is created (Alg. 1).

A. Inserting new graphs

Inserting a graph into the database is similar to the approach presented in [4]. Firstly the tree root is reached and then comparisons with all its children's graphs are performed. Comparisons are performed in order to find the best match between children's graphs nodes and the inserted graph (Fig. 5). When computing the similarity we divide the sum of found similarities between matched nodes by the minimum number of nodes in both graphs (Fig. 5 e) in order to avoid the situation when for the same matching, different similarity values are obtained (Fig. 5 c, d). If the similarity is high enough then its children are tested. If comparisons with two or more nodes returns high similarity, the node with the highest similarity value is used as a direction of the tree traversal. The new graph and common graphs comparisons are performed until the data node is reached. Then the graph insertion is performed, as described previously. Firstly the comparison of the first graph in the first slice is performed and then, based on the similarity result, the graph is put into the graphs vector in the position which is correlated with the sim value. If the maximum number of graphs in the vector is reached, the split operation is performed (Alg. 1). If during the tree traversal the comparison with the node's graph does not give enough similarity value, then a pair of new common and data nodes have to be created. The creation process is as follows: firstly a new graph which contains only common parts of the inserted graph and nodes's graph is being created and stored as a new common node. Next a new data node is created and the new graph is inserted into its data vector. Finally, the new common node is used as a parent for the node graph and new data node. The whole process is shown in the Fig. 6.

The graph insertion algorithm is presented in Alg. 2.

B. Querying the database

The database querying by a graph is much easier and faster with the proposed structure. Firstly, comparisons with all root children are performed. If similarity with one or more of

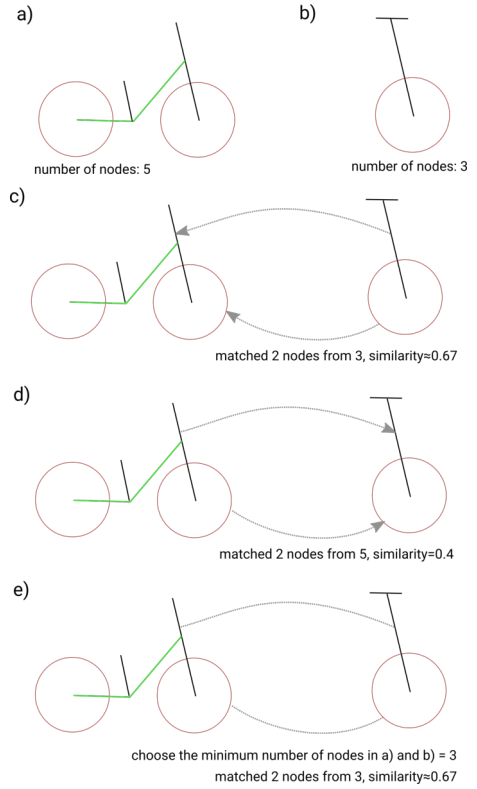

Fig. 5. Comparisons between graphs: a), b) example graphs with common nodes; c) comparison of a) with b); d) comparison of b) with a) e) the similarity computed using minimum number of nodes in graphs.

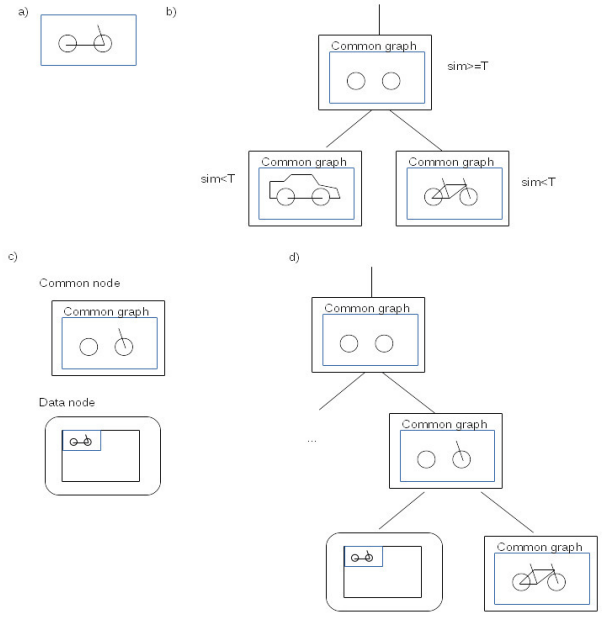

Fig. 6. The insertion of a new node with graph to the tree: a) the graph which has to be inserted, b) the tree structure c) two types of nodes after creation d) the tree after insertion of the nodes from c)

Algorithm 2 Inserting a new graph into the tree

Ensure: g - the graph which has to be inserted; T_{sim} - minimal similarity of graphs
$node \leftarrow root$;
while $node$ is not NULL **do**
  $traverse \leftarrow true$;
  if $node$ is a data node **then**
    compare g with first graph in first slice of $node$ [compare graphs using Alg. 3];
    insert g into $node$ in the position related to the similiarity;
    exit
  end if
  if $node$ is not a root **then**
    $sim \leftarrow$ compareCommon(g, $node->commonGraph$) [compare graphs using Alg. 3];
    if $sim < T_{sim}$ **then**
      $traverse \leftarrow false$;
    end if
  end if
  if $traverse$ **then**
    compare all $node$'s children common graphs with g, choose maximum similarity as sim [compare graphs using Alg. 3];
    if $sim < T_{sim}$ **then**
      $traverse \leftarrow false$;
    else
      $node \leftarrow$ child with max sim;
      continue;
    end if
  end if
  if !$traverse$ **then**
    $commGraph \leftarrow$ common part of g and $node$'s $commonGraph$;
    create new common node th
    insert $commGraph$ into th
    create new data node dh and add as child to th
    insert g into dh
    add $node$ as child to th
    exit
  end if
end while

them is high enough, each of them is tested until a common node with unsatisfactory similarity or a data node is reached. When a tree with not enough similarity is reached, the rest of the subtree is not tested. When a data node is reached, the comparisons within slices are performed. Firstly the similarity with the first and last graphs in slices are computed. If both are high enough, all graphs from slices are returned, if not, the proper range is specified using Algorithm 6.

The example querying process is shown in the the Fig. 7. The graph used for query is shown in Fig. 7 a). Firstly, the graph is compared with the root's child (id=2) which

Algorithm 3 Comparing a graph which has to be inserted with the graph in the tree node

Ensure: g_i - the graph which has to be inserted; g_{db} - graph which has to be matched to g_i (the graph which is stored in a tree); T_{conn} - minimal similarity threshold for connections test
$countNodes \leftarrow$ number of nodes in g_i;
for each $node_{gi}$ in g_i **do**
  for each $node_{gdb}$ in g_{db} **do**
    $sim_{gi,gdb} \leftarrow 0$
    if nodes types are different **then**
      continue;
    end if
    $simConn \leftarrow$ how many connections to other nodes in g_i has the same type as in g_{db};
    $simConn \leftarrow simConn \div countNodes$;
    if $simConn < T_{conn}$ **then**
      continue;
    end if
    $simPrim \leftarrow$ the similarity of primitives stored in nodes (returned by Alg. 4);
    try to match all connected nodes to $node_{gdb}$ onto the counterparts in $node_{gi}$ checking the similarity of primitives stored in nodes (by Alg. 4) and relative positions to other nodes, store the similarity result in $simPos$;
    $sim_{gi,gdb} \leftarrow simConn \cdot simPrim \cdot simPos$ store as similarity between $node_{gi}$ and $node_{gdb}$;
  end for
end for
$sim \leftarrow 0$
for each $node_{gi}$ in g_i **do**
  choose the match with nodes in g_{db} with highest $sim_{gi,gdb}$ value and add to sim ;
end for
$sim \leftarrow sim \div min($ number of nodes in g_i, number of nodes in g_{db};
return sim;

Algorithm 4 Comparing graphs nodes between each others. As a result the similarity coefficient is returned (values: <0,1>).

Ensure: pa, pb - primitives to compare;
if nodes types are different **then**
  return 0
end if
if nodes types are line segments **then**
  $diff \leftarrow |angle\ slope\ of\ pa - angle\ slope\ of\ pb|$
  return $sim \leftarrow 1 - diff$
end if
if nodes types are arches **then**
  $diff \leftarrow |angle\ of\ pa - angle\ of\ pb|$
  return $sim \leftarrow 1 - diff$
end if
if nodes types are polylines, polygons, polyarches or arc-sided polygons **then**
  $diff \leftarrow |number\ of\ segments\ in\ pa - number\ of\ segments\ in\ pb|$
  try to match all segments between pa and pb, choosing the smallest difference of their attributes, sum all corresponding differences and add to $diff$
  $sim \leftarrow (1 + minimum\ number\ of\ segments(pa, pb)) - diff$
  if $sim > 1$ **then**
    $sim \leftarrow 1$
    return sim
  end if
end if

gives the $similarity$ (sim) value equal 1 - all nodes between graphs were matched. Then the minimal similarity threshold (T_{sim}) is checked. The test was passed, the children nodes (id=3, id=4) are checked. The similarity result with the first node (id=3) does not passed the minimal similarity threshold test - the computed sim was equal 0.5 which is lower than T_{sim} value (0.8). These resulted in abandoning this tree path (consequently its child- id=5 is not tested). The comparison with the second id=2 child (id=4) returned $similarity$ equal to one, which is higher than T_{sim} value. Therefore, its children are tested - id=6 and id=7. The $similarity$ with id=6 node is equal 0.75 which is fairly high, but lower than T_{sim} and this path is also abandoned. Next, the id=7 is tested, the similarity is equal to 1, then its child (id=9) is tested. Since id=9 is a data node, other types of tests are performed. Firstly, the $similarity$ with the first element in the first slice is computed. Next the $similarity$ with the last element in the first slice is computed. In this example both values are higher or equal to T_{sim} so the whole slice is returned as a result of the query.

Because each subtree is tested independently, the querying algorithm could be easily paralleled. The querying algorithm without parallelism is shown in Alg. 5.

C. Deleting nodes

Deleting a graph from the database may be performed as follows: firstly if a graph is not the only one element in the slice, it may be removed from the vector without performing additional operations. If after removing a graph the slice does not contain any elements, the data node and its parent should be removed from the tree.

D. Query parallelization possibilities

The proposed database structure allows parallelization of a query process. Since testing each tree node is independent from others, it may be executed in different threads or machine nodes. If a data node stores many slices, they may be also checked independently. Gathering the results may need some synchronization if all results have to be sent at the same time. However, this process could be also implemented as asynchronous, sending partial results to the client when they are obtained.

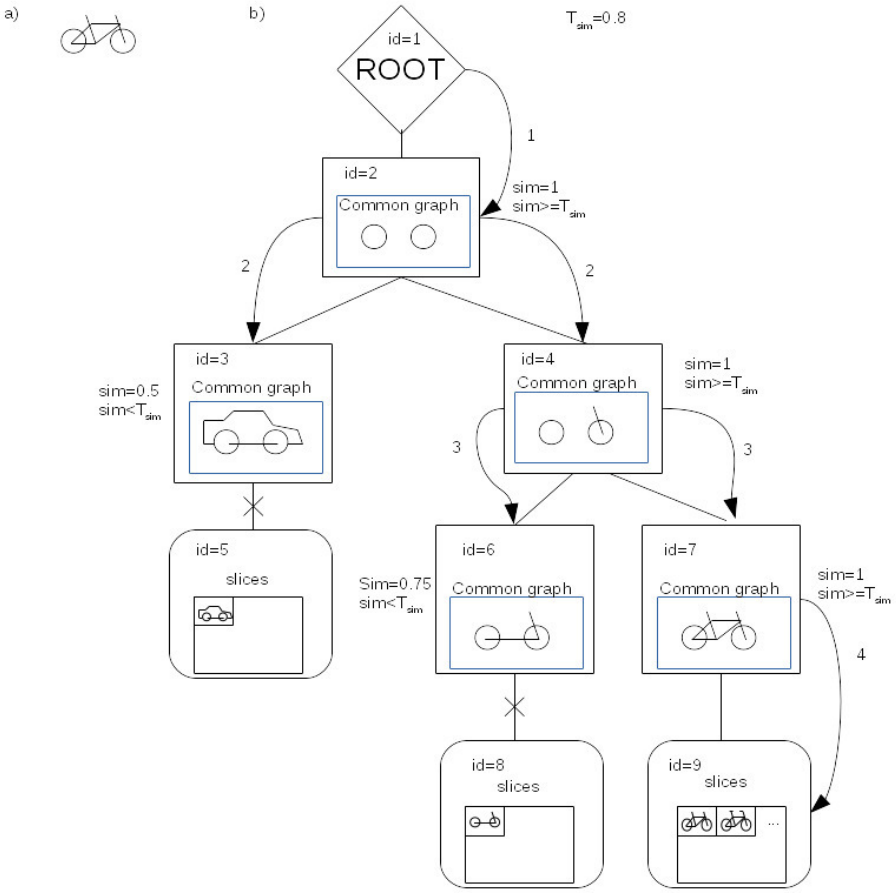

Fig. 7. The example query tree traversal: a) the query object graph, b) the query

Algorithm 5 Querying the database

Ensure: g - the query graph; T_{sim} - minimal similarity of graphs; $stack$ - a stack which is used to store nodes to check;
put $root$ into the $stack$;
while $stack$ is not empty **do**
  $node \leftarrow$ pop element from $stack$
  if $node$ is a data node **then**
    choose all graphs from $node$ slices using Alg. 6;
    continue;
  end if
  if $node$ is not a root **then**
    $sim \leftarrow$ similarity of g and common graph in $node$ [compare graphs using Alg. 3];
    if $sim >= T_{sim}$ **then**
      put all $node$ children to the $stack$;
    end if
  else
    put all $node$ children to the $stack$;
  end if
end while

VI. Experimental results

The proposed approach was initially tested using prototype database structure implementation written in C++ and database of cars, motorbikes, bicycles and scooters containing 111 images. In order to test the precision two coefficients were used:

$$precision = \frac{number\ of\ relevant\ results\ images}{total\ number\ of\ results\ images} \quad (1)$$

$$recall = \frac{number\ of\ relevant\ results\ images}{total\ number\ of\ relevant\ images\ in\ the\ database} \quad (2)$$

The test results for the chosen 6 objects are presented in the Table I. It may be observed that the precision of the results is high for bicycles, motorbikes and cars objects, but for scooter it is much lower. This was caused by the high similarity of scooter graphs to bicycles and motorbikes objects. However the recall values are much lower than precision. This is caused by usage of real life images which contained different variations of objects. As a feature research direction we would to increase this coefficient values.

Additionally some initial tests for comparisons between linear and tree database structure were performed in order to

Algorithm 6 Querying the slice in data node

Ensure: g - the query graph; $slices[1..n][1..m]$ - the n slices which stores vectors of m graphs; T_{sim} - minimal similarity of graphs;

```
for each slice in slices do
    L ← slice[1];
    R ← slice[m];
    while l <= r do
        sim_L ← similarity of g and first graph in slice[L]
        [compare graphs using Alg. 3];
        sim_R ← similarity of g and last graph in slice[R]
        [compare graphs using Alg. 3];
        if sim_L >= T_sim and sim_R >= T_sim then
            add all graphs from slice between L and R indexes
            into the result set;
            break while loop;
        else
            if sim_L >= T_sim then
                L ← L + 1;
            end if
            if sim_R >= T_sim then
                R ← R − 1;
            end if
        end if
    end while
end for
```

TABLE I
THE PRECISION AND RECALL RESULTS FOR CHOSEN TEST OBJECTS

object	Query by Shape	
	precision	recall
bicycle	0.93	0.37
bicycle (a sketch)	1.0	0.60
scooter	0.67	1.0
motorbike	0.86	0.40
car (Fiat 500)	0.89	0.33
car (Mercedes Benz)	0.79	0.73

observe how efficient is proposed structure. The results are presented in the Table II. The tests for two different number of elements were performed. It could be seen that for the smaller number of graphs (23) the query time is similar for both structures. When the number of graphs was increased (to 68) the tree structure returned results about two times faster than linear structure, which was expected. As our future research we would like to perform more similar tests with much higher number of graphs.

VII. Conclusion and future works

This paper presents a new Content Based Image Retrieval database structure. The main idea of the proposed approach is based on object representation proposed in [2]. Each object can be represented as a set of predefined shapes: a line segment, a polygon, a polyline, an arc, a polyarc and an arc-sided polygon. All shapes are connected into a graph, in order to store the mutual relations between them. The object representation allows users to use as a query images or simple sketches which does not need drawing skills. The proposed database structure is based on a tree with two types of nodes - common nodes which are used to organize the data and data nodes which stores similar graphs. This structure allow faster retrieval of results, because during the first query steps almost all not similar graphs are omitted. Moreover the query could be parallelized very easily in order to increase the performance. The proposed database structure is also more universal than our first approach presented in [4] because it is designed in order to allow different implementations suited for specific applications (e.g using SD2DS for servers or simple containers for mobile devices).

TABLE II
THE COMPARISON OF QUERY EXECUTION TIMES FOR LINEAR AND TREE DATA STRUCTURE.

structure	query time in microseconds	
	23 graphs	68 graphs
linear	62	114
tree	58	63

The future research includes testing the database structure with higher number of elements and comparisons with linear structure. Moreover the $recall$ coefficient should be improved. Another direction would be implementing the paralleled queries in order to test their efficiency. Another set of tests should be performed in order to evaluate different lower database level implementations, using e.g. SD2DS data structures or MySQL. Moreover different graphs comparisons algorithms may be tested, e.g. using optimization methods with constrains [23].

References

[1] T. Kasai and K. Takano, "Design of sketch-based image search ui for finger gesture," in *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, July 2016. doi: 10.1109/CISIS.2016.140 pp. 516–521.

[2] S. Deniziak and T. Michno, "Content based image retrieval using query by approximate shape," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2016, pp. 807–816.

[3] S. law Deniziak and T. Michno, "Query by shape for image retrieval from multimedia databases," *Beyond Databases, Architectures and Structures*, p. 377.

[4] S. Deniziak, T. Michno, and A. Krechowicz, "The scalable distributed two-layer content based image retrieval data store," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2015. doi: 10.15439/2015F272 pp. 827–832.

[5] S. Deniziak and T. Michno, "Query-by-shape interface for content based image retrieval," in *2015 8th International Conference on Human System Interaction (HSI)*, June 2015. doi: 10.1109/HSI.2015.7170652. ISSN 2158-2246 pp. 108–114.

[6] C.-Y. Li and C.-T. Hsu, "Image retrieval with relevance feedback based on graph-theoretic region correspondence estimation," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 447–456, April 2008.

[7] H. H. Wang, D. Mohamad, and N. A. Ismail, "Approaches, challenges and future direction of image retrieval," *CoRR*, vol. abs/1006.4568, 2010.

[8] A. Singh, S. Shekhar, and A. Jalal, "Semantic based image retrieval using multi-agent model by searching and filtering replicated web images," in *Information and Communication Technologies (WICT), 2012 World Congress on*, Oct 2012. doi: 10.1109/WICT.2012.6409187 pp. 817–821.

[9] C.-Y. Li and C.-T. Hsu, "Image retrieval with relevance feedback based on graph-theoretic region correspondence estimation," *Multimedia, IEEE Transactions on*, vol. 10, no. 3, pp. 447–456, April 2008. doi: 10.1109/TMM.2008.917421

[10] B. Li, Y. Lu, and J. Shen, "A semantic tree-based approach for sketch-based 3d model retrieval," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016. doi: 10.1109/ICPR.2016.7900240 pp. 3880–3885.

[11] M. Mocofan, I. Ermalai, M. Bucos, M. Onita, and B. Dragulescu, "Supervised tree content based search algorithm for multimedia image databases," in *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, May 2011. doi: 10.1109/SACI.2011.5873049 pp. 469–472.

[12] H. P. Kriegel, P. Kroger, P. Kunath, and A. Pryakhin, "Effective similarity search in multimedia databases using multiple representations," in *2006 12th International Multi-Media Modelling Conference*, 2006. doi: 10.1109/MMMC.2006.1651355. ISSN 1550-5502 pp. 4 pp.–.

[13] T. K. Shih, "Distributed multimedia databases," T. K. Shih, Ed. Hershey, PA, USA: IGI Global, 2002, ch. Distributed Multimedia Databases, pp. 2–12. ISBN 1-930708-29-7. [Online]. Available: http://dl.acm.org/citation.cfm?id=510695.510697

[14] A. Sluzek, "Machine vision in food recognition: Attempts to enhance CBVIR tools," in *Position Papers of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, Gdańsk, Poland, September 11-14, 2016.*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2016. doi: 10.15439/2016F579. ISBN 978-83-60810-93-4 pp. 57–61. [Online]. Available: https://doi.org/10.15439/2016F579

[15] C. Lalos, A. Doulamis, K. Konstanteli, P. Dellias, and T. Varvarigou, "An innovative content-based indexing technique with linear response suitable for pervasive environments," in *2008 International Workshop on Content-Based Multimedia Indexing*, June 2008. doi: 10.1109/CBMI.2008.4564983. ISSN 1949-3983 pp. 462–469.

[16] A. Sluzek, "On moment-based local operators for detecting image patterns," *Image and Vision Computing*, vol. 23, no. 3, pp. 287 – 298, 2005.

[17] M. Bielecka and M. Skomorowski, *Fuzzy-aided Parsing for Pattern Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 313–318. ISBN 978-3-540-75175-5. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75175-5_39

[18] T. Kato, T. Kurita, N. Otsu, and K. Hirata, "A sketch retrieval method for full color image database-query by visual example," in *11th IAPR International Conference on Pattern Recognition, Vol.I. Conference A: Computer Vision and Applications*, Aug 1992, pp. 530–533.

[19] Y. Zhang, X. Qian, X. Tan, J. Han, and Y. Tang, "Sketch-based image retrieval by salient contour reinforcement," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1604–1615, Aug 2016. doi: 10.1109/TMM.2016.2568138

[20] X. Qian, X. Tan, Y. Zhang, R. Hong, and M. Wang, "Enhancing sketch-based image retrieval by re-ranking and relevance feedback," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 195–208, Jan 2016. doi: 10.1109/TIP.2015.2497145

[21] C. Lalos, A. Doulamis, K. Konstanteli, P. Dellias, and T. Varvarigou, "An innovative content-based indexing technique with linear response suitable for pervasive environments," in *International Workshop on Content-Based Multimedia Indexing*, June 2008, pp. 462–469.

[22] S. Kiranyaz and M. Gabbouj, "Hierarchical cellular tree: An efficient indexing scheme for content-based retrieval on multimedia databases," *Multimedia, IEEE Transactions on*, vol. 9, no. 1, pp. 102–119, Jan 2007.

[23] P. Sitek and J. Wikarek, "A hybrid programming framework for modeling and solving constraint satisfaction and optimization problems," *Scientific Programming*, vol. 2016, 2016. doi: 10.1155/2016/5102616