Graph Object Oriented Database for Semantic Image Retrieval

Eugen Ganea and Marius Brezovan

University of Craiova, Craiova,
Bd. Decebal 107, Romania,
ganea_eugen@software.ucv.ro

Abstract. This paper presents a new method for image retrieval using a graph object oriented database for processing the information extracted from the image through the segmentation process and through the semantic interpretation of this information. The object oriented database schema is structured as a classes hierarchy based on graph data structure. A graph structure is used in all phases of the image processing: image segmentation, image annotation, image indexing and image retrieval. The experiments showed that the retrieval can be conducted with good results and the method has a good time complexity.

Keywords: Graph oriented object, object oriented database, image processing, image retrieval

1 Introduction

Image retrieval systems have been developed using a variety of technologies in various disciplines of computer science. In this paper, we use the concepts of object-oriented programming for object recognition applications. The object model used for storing images, is based on the complex and different structure for each image that does not allow a simple data model using predefined data structures such as those used in relational databases. Relational databases have several limitations in representing an image: from the perspective of data representation model, in the relational database, links between two records are achieved through attributes primary key and foreign key. The records have the same values for foreign keys, primary that are logically related, although they are not physically linked (logical references). In the Object-Oriented Databases ($OODB$), the relations is done by reference to an object identifier (OID) which is the key of association of the records. In addition, the object-oriented model, unlike the relational model, support structure that allows complex objects as sets, lists, trees or other advanced data structures. Also allows to define the methods by which messages are exchanged between objects and to implement the inheritance mechanism which offers classes which have new definitions based on existing definitions. An effective recognition of objects impose new requirements on the database structure, and it must exceed the role of a simple storage medium and provide the possibility of efficient retrievals and management capabilities for image information content; an image can be stored using a set of objects with attributes and characteristics that describe. In identifying an

unknown object, object recognition system queries the database and checking the similarities based on characteristics, between unknown object and each of objects from the database. Interface between database and object recognition system is done by sending these messages, which have the advantage of high flexibility in terms of how messages are processed. In an object-oriented database, each object in the real world can be modeled directly as an instance of a class; each instance has an OID and is associated with a simple or complex object. The OID stored in the database is not changed, while other fields associated object can be modified. This identity provides a good support for object sharing updates and simplifies management. Inheritance offered by object-oriented paradigm provides a powerful mechanism for organizing data, it allows to the user to define classes in an incremental way by specializing existing classes. In [1] introduced a model for $OODB$ representation based on a graph type data structure ($GOOD$ - Graph-Oriented Object Database), where the operations on database objects are translated into the transformation of the graph. Based on the approach used for image processing (segmentation and their annotation), database used have a scheme like in the $GOOD$ model. In addition the topological relations between the simple objects present in the image are represented by a graph edges, while the objects are the graph nodes. The structure of paper is organized as follows: in Subsection 1.2, presents the graph model for image representation and the method for construction of hexagonal structure on pixels image; Section 2 describes the process of image annotation based on ontologies; Section 3 presents the graph object oriented database structure; Section 4 describes our experimental results and Section 5 concludes the paper.

1.1 Related Work

In this section we briefly consider some of the related work that is most relevant to our approach. In the image segmentation area, the most graph-based segmentation methods attempt to search a certain structures in the associated edge weighted graph constructed on the image pixels, such as minimum spanning tree [2], or minimum cut [3]. The major concept used in graph-based clustering algorithms is the concept of homogeneity of regions. For color segmentation algorithms, the homogeneity of regions is color-based, and thus the edge weights are based on color distance. For image annotation, an approach based on graph is presented in [4], where the learning model is constructed in a simple manner by exploring the relationship among all images in the feature space and among all annotated keywords. The Nearest Spanning Chain method is proposed to construct the similarity graph that can locally adapt to the complicated data distribution. A recent research [5] associate the labels with regions detected in the training set, which poses a major challenge for learning strategy. They use a novel graph based semi-supervised learning approach to image annotation using multiple instances, which extends the conventional semi-supervised learning to multi-instance setting by introducing two level bag generator method. Object oriented database models [6] are based on the object-oriented techniques and their goal is representing by data as a collection of objects that are organized in hierarchy of classes and have complex values associated with them. The graph database models are an alternative to the limitations of traditional database models for capturing the inherent graph structure of data appearing in applications such as geographic database systems, where the interconnectivity of data is an

important aspect. The first object oriented database model based on a graph structure $O2$ was introduced by [7]. An explicit model named $GraphDB$ is presented in [8] and allows a simple modeling of graphs in an object oriented environment. The model permits an explicit representation of graphs by defining object classes whose instances can be viewed as nodes, edges and explicitly stored paths of a graph. The [9] describes the use of the $OODB$ in content-based medical images retrieval and the proposed approach accelerates image retrieval processing by distributing the workload of the image processing methods in the storing time. In this paper we use, as the core of the proposed management system, the $HyperGraphDB$ [10], which is a database based on hypergraph structure and was development on $BerkleyDB$ [11].

1.2 The Image Graph Model

The construction of the initial graph is based on a new utilization of pixels from the image that are integrated into a network type graph. We used a hexagonal network structure on the image pixels for representation of the graph $G = (V, E)$ and we considered too edges of graph joining the pseudo-gravity centers of the hexagons belongs to hexagonal network as presented in Fig. 1. As you can see, is achieved a triangulation of the image, which is in fact a decomposition of the image in a collection of triangles whose edges form the set V of nodes of the graph G. The condition for achieving a triangulation is satisfied, namely collection of triangles is mutually exclusive (no overlapping triangles) and fully exhaustive (all triangles meeting covers the original image). If it considered the edges which join the gravity pseudo-centers of the hexagons, it obtain a Delaunay triangulation [12]; the grid-graph is a Delaunay graph and based on planarity graph condition ($no\_edge \leq (3 * no\_vertex - 6)$) we demonstrated that the time complexity of segmentation algorithm is $O(nlogn)$. The algorithms for segmentation and the demonstration for complexity are presented in [18]. In the hexagonal structure, for each hexagon h in this structure there exist 6-hexagons that are neighbors in a 6-connected sense and the determination of indexes for 6-hexagons neighbors having as input the index of current hexagon is very simple. The main advantage when using hexagons instead of pixels as elementary piece of information is the reduction of the time complexity of the algorithms. The list of hexagons is stored such as a vector of integers $1 \ldots N$, where N, the number of hexagons, is determined based on the formula:

$$N = \frac{H-1}{2} \times \left(\frac{W - (W \mod 4)}{4} + \frac{W - (W \mod 4) - 4}{4}\right) \tag{1}$$

where H represents the height of the image and W represents the width of the image.

Each hexagon from the set of hexagons has associated two important attributes representing its dominant color and its pseudo-gravity center. For determining these attributes we use eight pixels: the six pixels of the hexagon frontier, and two interior pixels of the hexagon. The dominant color of a hexagon is the main vector color of all the eight colors of its associated pixels. We split the pixels of image into two sets, a set of pixels which represent the vertices of hexagons and a set of complementary pixels; the two lists will be used as inputs for the segmentation algorithm. The mapping of pixels network on the hexagons network is immediately and it is not time consuming in

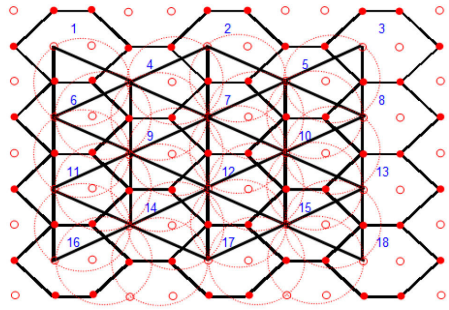

Fig. 1. The hexagonal structure on the image pixels

accordance with the following formula which determines the index for the first vertex of hexagon:

$$fv = 2 \times h + \frac{2 \times (h-1)}{columnNb - 1} \tag{2}$$

where fv represent the index of the first vertex, h the index of the hexagon and $columnNb$ represent the column number of the hexagon network. For representing the output of the image segmentation process we used the Attributed Relational Graph (ARG) [13]. The result of segmentation algorithm is stored as a graph where the nodes represent the regions and the edges represent the neighborhood relations: $G = (V\_r, E\_r)$, where V_r is the set of vertices corresponding regions detected and E_r is the set of edges that describes the neighborhood relations. The spatial relations between regions are divided into 3 categories: distance relations, direction relations and topological relations. For determining these types of relations we choose for each region the following relevant geometric features: the pseudo-center of gravity; the distance between two neighboring regions; the length of common boundary of two regions and the angle which is formed by two regions.

2 Image Annotation Based on Ontologies

In the image annotation process we use two types of ontologies: visual ontology (or object ontology) which refer to an intermediate level which connecting lower level features to high level concept, and domain ontologies [14] which refers to image content annotation. The management of ontologies, used for annotate of images, has two hierarchical levels that are closely associated. On the one hand, the low-level image contains specific properties as color, texture, shape, and the second level, which contains sematic of image that can be perceived human user. An ontology management system should model the low level that supports retrieval and inference level of their content. One scenario of using such a system can be that a user loads all ontologies in a given area, and allows selection of different objects in images and their correlation with the concepts of ontologies. For annotate the simple objects we used learning algorithms based on decision trees - Decision Tree based Semantic Templates algorithm (DT-ST) [15]. DT-ST

induction method for learning image semantics is different from classical algorithms that use semantic templates for continuous values of features of the regions. A ST feature is provided by the representative of a concept, the set of features extracted from the regions of the training images. Built the decision tree to assign high-level concepts, which are attached to leaf nodes of the tree, to lower level features, thus each useful concept of ontology will meet at least one leaf node. In the system developed, each image is divided into a number of areas which can attach semantic meaning, and each extracted region have as member, an instance of class $CFeatureVector$. For each concept we consider a vector with 7 components [$H\ S\ V\ perimeter\ compactness\ eccentricity\ area$], whose normalized values are used to construct the decision tree. The components H,S,V correspond to the HSV color space. The obtained decision tree is translated in a system of rules using Jess inference engine (Java Expert System Shell) [16].

2.1 Ontologies and RDF Format

For specifying the ontologies and the corresponding graph structure of the images segmented and annotated format we used RDF (Resource Description Framework). The RDF is a specification defined $metadata$ processing, providing interoperability between different applications such as an exchange of information, the purpose of understanding the semantics. To use the method of reasoning described in the ontology-based knowledge bases, we used $RDF2Jess$ model, a hybrid model that can be used to fill the gap between RDF and $Jess$. Based on domain knowledge and using Protégé [17] ontology editor, this method turns the RDF format in $Jess$ facts using XSL transformations on XML syntax and additional rules in Jess. For these rules redefined based on RDF semantics $Jess$ inference system is used to implement the reasoning. Predefined rules are used to check consistency and to determine the characteristics of RDF vocabulary. Deducted $Jess$ assertions are helpful for phase domain ontology modeling to assess and refine ontology. According to different levels of expressiveness, $RDF2Jess$ could be extended to new $SWRL2Jess$ where $SWRL$ extends the set of axioms to include Horn rules. The conversion of syntactic and semantic RDF in $Jess$, allows replacement of ontology in RDF format using $Jess$ reasoning engine. Ontology conversion into Jess facts and rules is done in four steps:

- first step is the ontology construction, ontology editor used $Protege$, provides a plug-in RDF ontology to support development. The taxonomy of knowledges into classes, features, restrictions was accepted by experts in the field and by software developers, because this paradigm is very similar to object-oriented modeling (UML). Lately, most ontologies have been formalized using standardized $RDF(S)$ and can be reused to extend the rules, if necessary;
- the second step is to represent the transformation of RDF syntax in $Jess$ syntax using $XSLT$, the output file consists of $Jess$ facts. If support ontology language semantics is specified as $Jess$ rules, matching specific keywords is no longer necessary in the transformation;
- the third step is to combine file $Jess$, including $XSLT$ transformation result, and RDF predefined rules. Moreover, external queries and $Jess$ rules can also be added to the composition similar properties;

– the last step is running the system of rules $Jess$ inference system. The rules defined, is the classification and consistency checking characteristics. The output containing erroneous messages indicating the presence of incorrect syntax in RDF ontology processed.

To implement the RDF, semantics are defined to represent the additional facts and their relations with $RDFS$ primitives such as $rdfs : Class$.

2.2 Using Object-Oriented Graph Grammars for Learning Complex Concepts

The graph grammars were first used for image representation in [19]. There are several areas that have been used in graph grammars, such as recognition of musical notation and biology. In recent research, the type graph grammars have been used to define visual languages [20]. By using graph grammars the syntactic representation and analysis of images are defined the spatial relationships between regions of an image, in Fig. 2 is an example of a graph grammar production rule. Specify model involves determining the

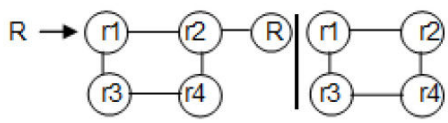

Fig. 2. Example of production rule

syntactic graph grammar appropriate for a particular domain. The inference algorithm is described below and uses the graphs obtained by automatic segmentation and annotation of simple regions as input. Grammar induction system for a type graph, $SubdueGL$ [21], developed within the scope of use, with little change in terms of performance and added variants of generally optional features to the original system. $SubdueGL$ algorithm was developed based on $Subdue$ [22] and uses a breakthrough approach to subgraphs, which focuses on a set of data compression of the graph, contrary to finding the most frequent sub-graphs. Although approaches based on compression and frequency sub-graphs are closely related, they may produce different results as a sub-graph less frequently overall compression can produce a good set of data. Using a growth process graph,$SubdueGL$ generate candidate sub-structures can be used to compress the data set of the graph. Original method uses the value of compression, minimum description length (MDL) to compare the compression of each sub data set of candidate structures. Sub-structure with the highest value for MDL is used to stage data compression crowd. The process is repeated until the set of data is being compressed all - remains a single node or any sub-structure can not be found or when it reached a specified number of iterations. Compression process causes a reduction in hierarchical sub-graphs connected directly corresponding grammar production rules. In Fig. 3 is presented an example of the graph grammar determined by applying the algorithm $SubdueGL$. For a grammar,

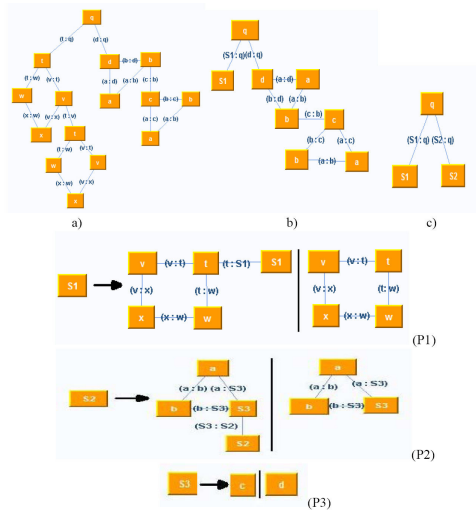

Fig. 3. $SubdueGL$ application example: a) original graph b) graph production recovered after determining P1 c) the final graph obtained after applying productions obtained: P1, P2, P3

so determined, has developed a syntactic analyzer that attaches each production, semantic actions that are used to determine the semantic concepts. For implementing the syntactic analyzer has utilized the CUP Parser Generator for Java - $JavaCup$ [23].

3 Graph Object Oriented Database

The semantic information correspond to concepts of domain ontology on the one hand and to elements of visual ontology on the other hand. The visual concepts determined automatically in the phase of post-processing of segmentation results are stored implicitly in the ARG structure representing relations between regions of an image. Each semantic object will have an attribute that points to the object region interpreted (OID). The OID of semantic object is the same with the identifier of the corresponding synset from $WordNet$ [24]. In this way, we manage the uniqueness of OID attribute values and we provide the link for an annotation with different concepts, but there is a relationship between synonyms. This approach prevents duplication of semantic information in the database, and an overview of how to store links between visual concepts/concepts domain and related regions is given by Fig. 4.

3.1 Indexing Graph Object Oriented Database

Indexing problem is approached using graph theory, the relationship is represented by indexing the index assign classes and forming a directed graph. Other approaches [25] refer to the database schema, thus the optimal time for the selection indexes as high

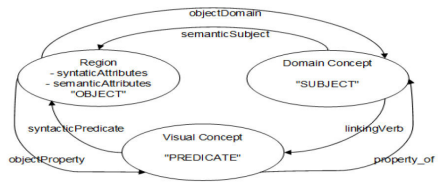

Fig. 4. Triple link: visual concept/semantic concept/Region

complexity. Based on the database scheme, was developed a new approach to the problem of indexing by exploiting graph structure type. This type uses an index nested/inherited, the algorithms required for retrieval and update operations can be compared with other approaches [26]. Type are used as index information, related sites OID objects from the database and corresponding regions in syntactic chains detected in the phase image segmentation. Using facet $index - propagation$ attached to each attribute indexes are grouped according to the characteristics of each object processed region. Grouping (which includes enable/disable attributes through facet $index-propagation$) is achieved in Algorithm 1 based on a function implemented in the states of each class. The function $hyperGraphGroup$ that does clustering indexes, used as support for

Algorithm 1: Grouping the attributes index

Input: The current instance
Output: The index of current instance

```
1  Procedure groupIndexObject (thisObject;indexThis);
2  attributeSet ← atributesOf (thisObject);
3  initialize attributeIndexSet;
4  for attribute ← attributeSet do
5  |   if attribute.pattern-match-facet is reactive then
6  |   |   add attribute to attributeIndexSet;
7  |   end
8  end
9  for attribute ← attributeIndexSet do
10 |   if attribute.index-propagation-facet is non − inherit then
11 |   |   remove attribute to attributeIndexSet;
12 |   end
13 end
14 initialize indexThis;
15 for attribute ← attributeIndexSet do
16 |   indexThis ← hyperGraphGroup(indexThis, attribute);
17 end
```

storing and linking indexes a $hypergraph$ [27], implemented by the $CHypergraph$ class, subject to the previous result that output algorithm ($indexThis$) is an instance of

$CHypergraph$. Choosing $hypergraph$ type structure to represent indexes was made because this type of structure is very good for browsing and retrieving images corresponding graph processed. The function $HyperGraphGroup$ properly algorithm is presented in Algorithm 2.

Algorithm 2: Function hyperGraphGroup

```
   Input: The index of current instance, the current attribute
 1 Function hyperGraphGroup (indexThis, attribute);
 2 initialize currentHEdge;
 3 if isSyntactic (attribute) then
 4 |   currentHEdge ← addToSyntacticHyperEdge (attribute)
 5 end
 6 else
 7 |   currentHEdge ← addToSemanticHyperEdge (attribute)
 8 end
 9 hyperEdgeSet ← hyperEdges (indexThis) U currentHEdge;
10 clear indexThis;
11 for hyperEdge ← hyperEdgeSet do
12 |   if isNonTopological (hyperEdge) then
13 |   |   add hyperEdges to indexThis
14 |   end
15 |   else
16 |   |   add hyperEdge to indexThis
17 |   end
18 end
19 return indexThis
```

For the index management of the $OODB$ we build a system of indexes based on the geometric and the semantic attributes of the shapes. The object-oriented database allows creating an index via a specific field or group of fields. The using of indexes improves the query performance, but in the same time the indexes are stored also in the database and the growing of the size which can lead to a decreasing of the storage performance. As a result of our tests we consider two groups of indexes; the first group (geometric group) is used only for the training images in the off-line phase of system utilization - learning phase, and the second group (semantic group) is used for all other images in the online phase of system utilization - symbolic query phase. First the group of indexes belongs to the attributes extracted after the image segmentation: the perimeter, the gravity center, compactness of shape, eccentricity of shape, the list of gravity centers of hexagons from the contour and the syntactic characteristics of the boundary shape. This approach drives at a good optimization of the retrieval process for linking an image to a synset. In this stage the $OODB$ contains only the information corresponding to the ontology so the space taken by the system of indexes has not influence concerning the storage performance. At the end of this phase the first group of indexes is deleted.

3.2 Retrieval Graph Object Oriented Database

The power of the native query is given by the flexibility of the object-oriented paradigm and by the possibility of using the dynamics queries, which are easily implemented based on the proprieties of the object-oriented languages. In this way the productivity corresponding to the object-oriented programming is not affected through the utilization of the standard SQL query. The query expressions written in symbolic language must be analyzed and converted to an equivalent native query format. In this process the relationships between the concepts of the ontology on one part and between concepts and classes on the other part are used. The translation supposes two stages: in the first step is used the $WordNet$ taxonomy; in the second stage the mapping of concepts on the classes is used. For all the words present in the query expression we search the correspondence with the synsets from the $WordNet$ taxonomy and mark these synsets. In the case when a word has not a synset, we use the synonym relation of the taxonomy to retrieve the synset. If it is not found a synset, the word it is returned to the user as no relevance for semantic query and it is extracted from expression. After this stage for the list of words from initial query we have a list with synsets. In the second step for each returned synset from the list after the first stage we determine the corresponding class and we make an instance for the class through the call of the constructor which receives the name of the synset. All of these instances and classes are used in the process of matching with the objects stored in the $OODB$. After the execution, a native query is obtained in this mode, and we have a list of objects corresponding to the images with semantic content according to the query expression. The name of the file which contains the physique image is formatted based on the unique identified attribute of each attribute. Through the graphical user interface these images are showed to the user and are grouped in clusters according to the input list of synsets.

4 Experiments

A prototype system was designed and implemented in $Java$, and $HyperGraphDB$. We tested our system on $Princeton\ Event$ dataset [28] and on MPEG7 CE Shape-1 Part B dataset [29]. The $MPEG7$ database consists of 70 classes and 20 shapes per class. The retrieval process implies two categories of experiments: a) the retrieval process based on symbolic language queries, and b) the retrieval process based on query-by-example.

4.1 Retrieving with Symbolic Language Query

In this case there were taken into consideration the pseudo-natural queries based on the concepts from ontology. We use the $Princeton\ Event$ dataset which contains 8 sport activities with about 200 images per category: badminton, bocce, croquet, polo, rock-climbing, rowing, sailing and snowboarding. For each category we consider 25 representatives images for the learning phase. In the $OODB$ "sports.hgdb" are stored initially the information extracted by the segmentation from the training images. Using the indexes as the perimeter, the pseudo-gravity center, compactness of shape, eccentricity of shape, the list of gravity centers of hexagons from the contour and the syntactic

characteristics of the boundary shape we allocate and store in the sports.hgdb all images corresponding to the dataset. After the learning and storing phase the OODB is ready to be interrogated. The pseudo-natural query considered is:

red ball one hoop

Using the data from the ontology and the $Wordnet$ information this query is translated in equivalent object oriented native query:

```
CImage imageQuery =
    new CImage ("croquet with red ball and one hoop")
List<CImage> resultsImage =
    db.query (new Predicate<CImage>()
{
    public boolean match(CImage image)
    {
    return imageQuery.getName().
            Equals (image.getName());
    }
});
```

Figure 5 shows the results for this query applied on our "sports.hgdb" databases; we consider only the first 16 retrieval images.

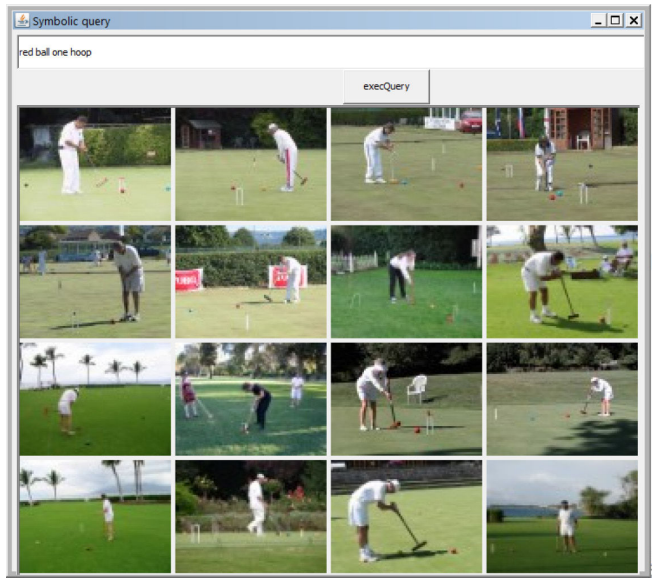

Fig. 5. The results images for the semantic query

4.2 Retrieving with Query-by-Example

In this case there were taken into consideration the query-by-example based on query image; we used images from 16 classes (MPEG7 CE Shape-1 Part B) for evaluate the performance of the shape recognition system based on the retrieval rate.In the $OODB$ "shapes.hgdb" is stored initially the information extracted by segmentation from all the images; the stored data are the perimeter, the gravity center, compactness of shape, eccentricity of shape, the list of gravity centers of hexagons from the contour and the syntactic characteristics of the boundary shape. Fig. 6 shows the results for this query applied on our "shapes.hgdb" databases; we consider only first 14 retrieval images, where the first image represent the query image.

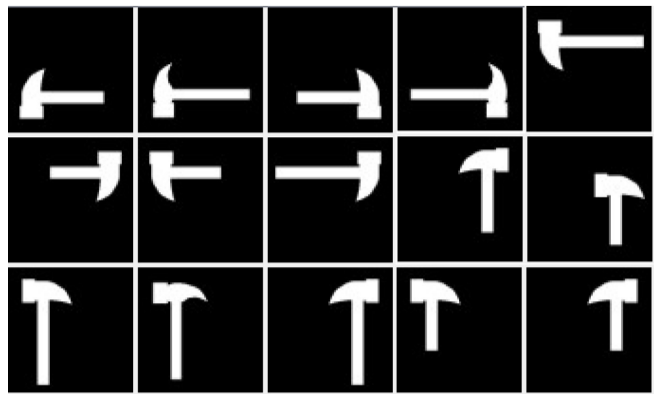

Fig. 6. The results images for the query-by-example

5 Conclusions

In this paper, we propose a method for image processing based on graph structure with the aim of good retrievals. The process have three phases: (I) an image segmentation stage based on graph structure and theory; (II) an adaptive visual feature object-oriented representation of image contents; and (III) a management of the ontologies uses for annotation of the objects from images. Using these tree stages and an object-oriented wrapper for $HyperGraphDB$, the system allows two types of queries: a query-by-example, but especially a query based on symbolic language. The experiments showed that the retrieval process can conducte to good results regardless of the area that the images come from. The future work implies the description and the using of the graph grammar with the goal of searching and retrieving complex images based on the complex query formulated in a symbolic language.

References

1. M. Gyssens, J. Paredaens, J. Van den Bussche, D.V. Gucht, *A graph-oriented object database model*, IEEE Trans. on Knowl. and Data Eng., 6(4), 1994.
2. P.F. Felzenszwalb and W.D. Huttenlocher, *Efficient Graph-Based Image Segmentation*, Intl. Journal of Computer Vision, 59(2), 167–181, 2004.
3. J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.
4. J. Liu, M. Li, W.-Y. Ma, Q. Liu, H .Lu, *An adaptive graph model for automatic image annotation*, Multimedia Information Retrieval, 61–70, 2006.
5. X. Hu and X. Qian, *A Novel Graph-based Image Annotation with Two Level Bag Generators*, Computational Intelligence and Security, International Conference on, 2:71–75, 2009.
6. W. Kim, *Object-Oriented Databases: Definition and Research Directions*, IEEE Transactions on Knowledge and Data Engineering, 2(3):327–341, 1990.
7. C. Lécluse, P. Richard, F. Vélez, *O2, an Object-Oriented Data Model*, Proceedings of the ACM SIGMOD Int. Conf. on Management of Data, pages 424–433, 1988.
8. R. H. Guting, *GraphDB: Modeling and Querying Graphs in Databases*, Proceedings of 20th Int. Conf. on Very Large Data Bases, pages 297-308, 1994
9. C.Jr. Traina, A.J.M. Traina, R.A. Ribeiro, E.Y. Senzako *Content-based Medical Images Retrieval in Object Oriented Database*, Proceedings of 10th IEEE Symposium on Computer-Based Medical System - Part II, 67–72, 1997.
10. *HyperGraphDb*, http://www.kobrix.com/hgdb.jsp (consulted 01/02/2010).
11. A.M. Olson, K. Bostic, M. Seltzer, *Berkeley DB*, Proceedings of the FREENIX Track: USENIX Annual Technical Conference, 1999.
12. B. Delaunay, *Sur la sphére vide*, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 7:793–800, 1934.
13. P. Hong and T. S.Huang, *Spatial pattern discovery by learning a probabilistic parametric relational graphs*, Discrete Applied Mathematics, 139:113-135, 2004.
14. C. Town, D. Sinclair, *Language-based querying of image collections on the basis of an extensible ontology*, Image Vision Comput., 22(3): 251–267, 2004.
15. Y. Liu, D. Zhang, G. Lu, A. Tan, *Integrating Semantic Templates with Decision Tree for Image Semantic Learning*, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, ISSN: 0302-9743, 4352:185-195, 2007.
16. E. Friedman-Hill, *Jess in Action : Java Rule-Based Systems*, Manning Publications, 2002.
17. *Protégé project*, http://protege.stanford.edu/ (consulted 17/06/2009).
18. D.D. Burdescu, M. Brezovan, E. Ganea, L. Stanescu, *A New Method for Segmentation of Images Represented in a HSV Color Space*, Advanced Concepts for Intelligent Vision Systems, Bordeaux, France, 2009.
19. J.L. Pfaltz, and A. Rosenfeld *Web grammars*, International Joint Conferences on Artificial Intelligence, 609–620, 1969.
20. J. Kong and K. Zhang *On a spatial graph grammar formalism* Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC'04), 102–104, Washington, DC, USA, 2004.
21. I. Jonyer, L. Holder, D. Cook *Concept Formation Using Graph Grammars*, Proceedings of the KDD Workshop on Multi-Relational Data Mining, 2002.
22. L.B. Holder *Empirical Substructure Discovery*, In Proceedings of the Sixth International Workshop on Machine Learning, 133–136, 1989.
23. *JavaCup*, http://www.cs.princeton.edu/ appel/modern/java/CUP (consulted 17/04/2009).
24. Miller, G.A., *Nouns in WordNet: a Lexical Inheritance System*, International Journal of Lexicography, 4:245–264, 1990.

25. R. Gagliardi and P. Zezula *An indexing model for object-oriented database systems*, Advanced Computer Technology, Reliable Systems and Applications. 5th Annual European Computer Conference Proceedings, 287 – 289, 1991.
26. E. Bertino *A survey of indexing techniques for object-oriented databases*, Proceedings Dagsthul Seminar Query Processing in Object-Oriented, Complex-Objectand Nested Relational Databases, 1993.
27. B. Goertzel *Patterns, Hypergraphs and Embodied General Intelligence*, IJCNN, Neural Networks International Joint Conference on, 451–458, 2006.
28. L.-J. Li and L. Fei-Fei. *What, where and who? Classifying event by scene and object recognition*, IEEE International Conference in Computer Vision (ICCV), 2007.
29. B. Leibe, B. Schiele, *Analyzing Appearance and Contour Based Methods for Object Categorization*, International Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, June 2003.