Proceedings of the Federated Conference on Computer Science and Information Systems pp. 807–816
DOI: 10.15439/2016F233
ACSIS, Vol. 8. ISSN 2300-5963

Content Based Image Retrieval using Query by Approximate Shape

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*—In this paper we present a new method for content-based image retrieval. The method is based on querying database by approximate shape representing given object. In this way all images containing the object may be found. Shapes are specified as a set of geometric primitives and attributes. Relations between primitives are represented by a graph. Our graph matching algorithm is used for computing the level of similarity between shapes. The method may be also used for searching transformed as well as partially covered objects. Experimental results showed the efficiency of our approach.**

I. Introduction

SEARCHING for specific images or any graphical objects is one of the most challenging problem in image analysis, called image retrieval. The large image data sets are stored in databases, file systems, Internet resources, and other repositories. Moreover, the size of each individual image is increasing due to the development of new high-resolution sensors. Methods which are used in order to retrieve images may be grouped into the following three categories: Keyword Based Image Retrieval (KBIR), Semantic Based Image Retrieval (SBIR) and Content Based Image Retrieval (CBIR) [1].

The KBIR methods are based on metadata which describes images stored in the database. During the query, the set of keywords is compared with the metadata- textual description for each image, which most often contains also a set of keywords. KBIR methods provide good results and are fast, but their quality depends on the image annotations. In many cases image descriptions are not available, annotations are incomplete or not ambiguous. For example the person who is preparing keywords for images may not have enough knowledge or identify some objects wrongly. There are also some algorithms which tries to prepare keywords automatically without human interaction, but they suffer for the same problems. Moreover the number and precision of descriptions is also very important. In order to overcome problems with proper keywords, the CBIR algorithms were proposed [2]. In this group, a query has a form of a sample image, which is used to retrieve similar images. In SBIR methods queries concern semantic meaning of the image. They are based on textual annotations or automatic image recognition.

In this paper, a new Content Based Image Retrieval method is proposed. The idea is based on image decomposition into primitives with attributes. The detected primitives are transformed into a graph which is used for object matching. The main advantage of our method is that it is not based on strict comparison of objects, but it compares approximate shapes, therefore it may be applied for transformed (e.g. scaled, rotated, tilted) as well as partially covered objects.

The paper is organized as follows: next section describes the review of existing content-based image retrieval methods and the motivation to this work. Our algorithm is presented in the section 3. Finally, experimental results, conclusion and further directions of our research are given.

II. Content-Based Image Retrieval

Two types of CBIR algorithms may be distinguished: low-level approach and high-level one. The low-level algorithms, during image processing, use the whole frame, e.g. computing normalized color histogram [3], a difference moment and entropy [5], spatial domain [4] or MPEG-7 shape and texture descriptors [6]. The results of low-level algorithms are most often precise when searching for the whole image (e.g. a painting), but when querying for a specific object they are insufficient. In this case, the high-level algorithms provides much better results. Most of them are based on regions which are groups of similar pixels. After region extraction, they are transformed into graphs and compared with graphs from the database. During the region extraction different techniques are used, e.g. color-based or fuzzy patterns recognition [7]. This group of algorithms provides very precise results but are problematic for users when they does not have full knowledge about searched objects or does not have sample query image.

Our work focuses on CBIR method based on query by shape. Such methods are useful when a user looks for images containing given class of objects, and the query is specified as an approximate sketch drawn manually or using graphical editor. There are some methods which deal with this problem. In [8] the human drawn image sketches are compared, but their algorithm relied on low-level image sketch and was not oriented on objects. There are a lot of methods for shape extraction from images for image retrieving. They are based of FFT [10] or on extracting some features of the shape [9]. Others are using stroke points with gradient fields which are combined with Poisson HOG [15] or edge based shape vectors [16]. There were also some attempts to use both raster and vectorized image, e.g. using color moment and topogeo

978-83-60810-90-3/$25.00

descriptors [17]. It was shown that shape-based image retrieval methods are very efficient. But existing methods are based on strict shape matching, therefore transformed or partially covered shapes may not be recognized. In [11] a method where query image may be a hand-drawn sketch was proposed. The method compares the query with the whole image using wavelet decompositions.

In our previous work [12], [13], [14] we proposed some high-level methods which are based on predefined shapes, easy to drawn by a human but also to extract from the query image. Our preliminary works showed that our graph-based representation of shapes is very suitable for object matching, even when object are deformed. Hence, we decided to continue our research. The goal of this research was to propose a new CBIR algorithm which will be suitable not only for queries specified by image objects but also for human drawn queries. The queries decompose object shape into smaller parts which are then compared with shapes extracted automatically from images stored in the database. Moreover, because objects are represented by approximate shapes, the matching is defined by the level of similarity.

Our previous researches were focused only on two primitive types - lines and ellipses. During the tests, we found that this set provide good results for human-based, angular objects, but for the other types, the results may be insufficient. In order to improve the precision of results for such objects, the primitive set had to be extended with rounded ones. Moreover, during experiments we also noticed that in some parts of objects there is very important to store the relations between primitives of the same type. For example, some of the lines in the bicycle object are connected with each others and when combined, they create a chassis. In our previous researches that situation was covered by connections between primitives in the graph, but we decided to strengthen the validity of that fact.

III. Query by Approximate Shape

The method is based on two ideas: an object representation and a matching algorithm. Objects are specified using shapes decomposed into primitives which are extracted from the image or are drawn by an user. Shapes are represented with graphs, where nodes correspond to primitives and edges are between nodes associated with adjoining primitives. After building a graph, the matching algorithm, which compares the queried object with an object from the database, is applied. The system overview is shown in the Fig. 1.

A. Object representation

The main idea of our approach is based on approximation of any shape with limited set of simple primitives. Each primitive shape may have attribute defining size, color, orientation etc. The two most basic types of primitives are line segment and arc. Usually lines and arcs compose more complex shapes, therefore we also consider polylines and poly arcs as primitives. When polyline (or poly-arc) creates closed loop, it creates a plane. Since the plane has additional attributes like texture or color of the surface, we also consider polygons and arc-sided polygons as primitives. Hence, 6 different primitives are distinguished (Fig. 2). All shapes are approximated using the above primitives with attributes. It is very important that approximation should be deterministic, i.e. similar objects should be approximated with similar shapes. Query images as well as images from the database are approximated in the same way.

The predefined set of primitives allows approximation of different shapes, even composed of curved segments. For each primitive we also define the following basic attributes, which describes orientation of the primitive:

- for line segment: the size and the angle defining the slope,
- for polyline and polygon: the number of line segments, attributes of the following line segments,
- for arc: the size and the angle,
- for poly-arc and arc-side polygon: the number of segments, attributes of the following arcs.

In order to allow comparisons of shapes with different sizes, all sizes are normalized and have values between 0 and 1.

The approximation of any object is done using the following procedure: first edge detection is performed. Then all segments are approximated with lines and arches, next all polygons and arc-sided polygons are extracted, finally connected lines and arches are converted into polylines and poly-arcs. Since after line segments detection in real life images very often lines are divided into smaller parts, a line merging should be applied. If the distance between endings of two line segments is below the line merging threshold (Fig. 3 a)), their angles should be tested. If for both segments angles are the same, the merging could be performed. Moreover there may be also a situation when an arc was detected as a set of lines (Fig. 3 c)). In order to detect and convert set of segments into an arc, firstly it should contain at least 3 connected segments with endings in very close distance. Next, the angles between lines should be measured. If their values are the same—the set of segments could be converted to an arc (Fig. 3 d)).

The predefined set of primitives covers most geometric shapes, the Fig. 4 shows the example car (b), bicycle (c) and flower (d) objects. There may be noticed that all circles are detected as arches (with 360° angle). Next, all primitives based on arches and line segments have to be detected. Firstly all polygons and polygon arches are extracted, next polylines and poly-arches. When a primitive is detected, all its line segments or arches must not be used as part of other primitives.

After detection of primitives, the graph representation is being built. During this process, the following relations between primitives are stored:

- which primitives are close to each other or which primitives are connected,
- positions of the above primitives (using 8 directions: N, E, W, S, NW, NE, SW, SE - see Fig. 5).

When some primitives are connected to each other, they create a more complex shape (e.g. adjoined triangles, quadrangles and arches). Because information about which primitives belong to which complex shape and how complex shapes are

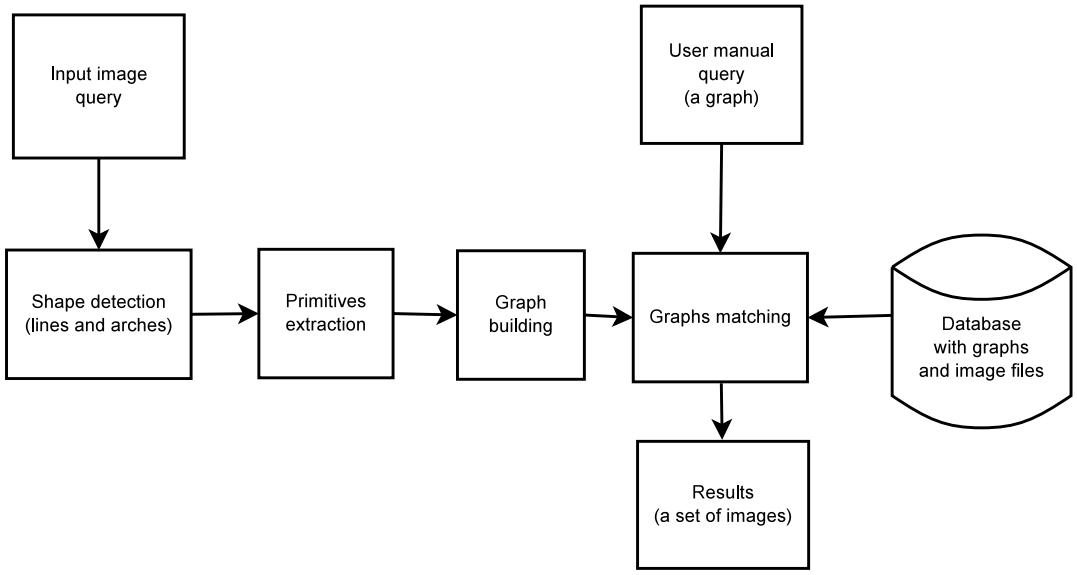


Fig. 1. The system overview.

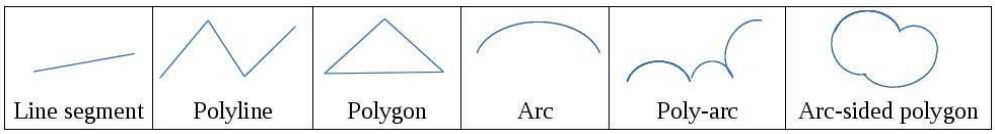


Fig. 2. Predefined primitives

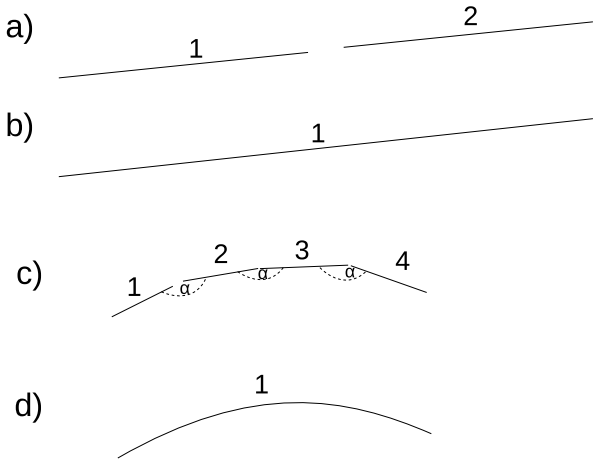


Fig. 3. Lines preprocessing. a) a splitted line into two segments with the same angles and very close distance between endings, b) line after merging, c) lines which may construct an arc d) an arc after lines conversion.

situated to each other may be useful during object matching, it is also stored. Moreover, in real life images, sometimes primitives which should be connected are not connected but are placed in a very close distance, for example because of inaccuracies of line segment detection algorithm. To overcome this problem, the maximal distance between primitives is used when creating connections.

The Fig. 4 shows the conversion of a sample car, bicycle and flower objects into graphs. Firstly, the algorithm detected all base primitives - arches and lines. For the car and bicycle objects, wheels and gear were detected as arches with $360°$, while their chassis as a set of lines. Some parts of car's body were recognized both as arches and lines. For the flower object, stalk and all petals were detected as arches. The second algorithm step, construct more complex primitives on the basis of the previous detection. Firstly, polygon arches and polylines are extracted as a result of the process of line and arches endings examination. For example, if the distance between two lines endings is lower than maximal distance threshold, they are combined as a polyline. The maximal distance threshold was introduced as a result of our previous tests with real life images. Very often we noticed the situation when the lines had to be connected, but as a result of some failures of line detection algorithm there was some space between the endings. In this step, all petals were combined as a one polygon arc, and car's windows and bicycle chassis as polylines. After polylines detection, the algorithm checks if they could construct a polygon (as a result, one of the windows and part of chassis were transformed). The last algorithm step constructs complex shapes and graphs. All detected primitives which are close to each others are combined using maximal distance threshold. All their position relations are stored, as was stated previously (e.g. the stalk arc is slightly on the left and below the petals polygon arc - the SW direction). For

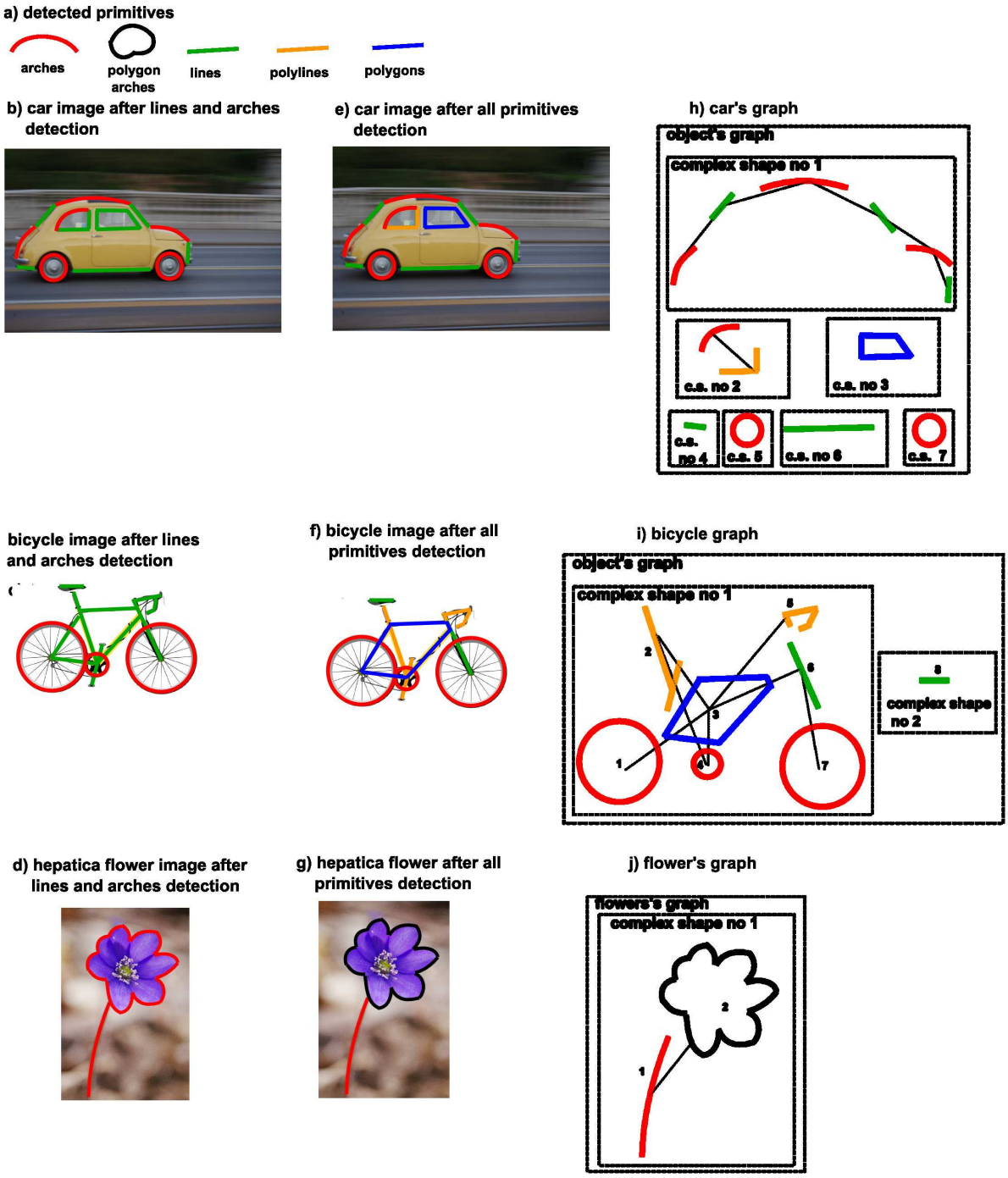


Fig. 4. The examples of primitives detection and graphs used for comparisons.

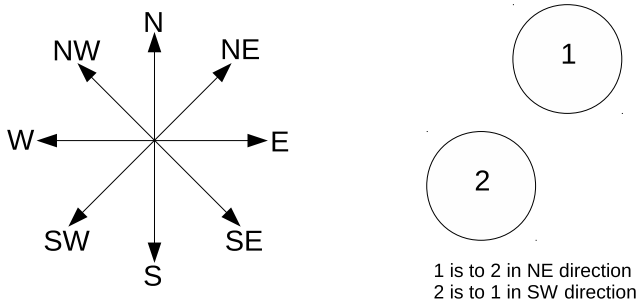


Fig. 5. The directions used in the algorithm and example of usage.

the car, 7 complex shapes were created, because there were unconnected groups of primitives. But in case of the flower, all primitives were connected and created only one complex shape.

B. Matching algorithm

The matching algorithm is partially based on our previous research [12]. All comparisons between objects are performed hierarchically in order to improve the performance, but also to reject objects which are not similar in the first steps. The algorithm is as follows:

Assumptions:

- O - searched object
- M - object from the database with which O is compared
- T_{CS} - threshold which defines the minimal number of complex shapes from O matched with M - values: $< 0, 1 >$, 0 - none, 1 - all
- T_P - threshold which defines the maximal difference between parameters, values: $< 0, 1 >$, 0 means the values are the same
- T_N - threshold which defines the minimal number of connections to other primitives with the same type and mutual position between primitives - values: $< 0, 1 >$, 0 - none, 1 - all
- T_{sim} - threshold which defines the minimal similarity of two objects graphs, values: $< 0, 1 >$, 0 - objects are completely different, 1 - the same objects
- thresholds: T_{CS}, T_P and T_N are used only to improve the performance and to reject not similar complex shapes or primitives in the earliest algorithm stages.

1) Try to match all complex shapes of O to all complex shapes of M, comparing the number and types of primitives and mutual positions.
   a) N_{CSP} = the number of primitives in O's complex shape with the same type as in M's complex shape
   N_{CSO} = the number of primitives in the O's complex shape
   N_{CSM} = the number of primitives in the M's complex shape
   $sim_{CS} = \frac{N_{CSP}}{max(N_{CSM}, N_{CSM})}$
   b) If $sim_{CS} < T_{CS}$ then go to a) and compare O's complex shape with the next M's complex shape.
   c) Add the pair of complex shapes to the list, storing theirs sim_{CS}.
2) For each pair of complex shapes CS_O (O's complex shape) and CS_M (M's complex shape) from the list:
   a) Try to match primitives - compare each primitive P_{CSO} of CS_O with each primitive P_{CSM} of CS_M:
      i) If P_{CSO} and P_{CSM} types are different, check another CS_M's primitive and go to a)
      ii) Compare primitives attributes and compute sim_P coefficient:
      $$c = \sum_{i=1}^{n} |(i^{th}\ attribute\ of\ P_{CSO}) - (i^{th}\ attribute\ of\ P_{CSM})|$$
      $$sim_P = \frac{c}{n}$$
      iii) If $sim_P > T_P$ then check another CS_M's primitive and go to a)
      iv) Compare connections of P_{CSO} and P_{CSM}:
      N_{PCSO} = the number of P_{CSO}'s connection to other primitives with the same type and mutual position as in P_{CSM}
      $$sim_N = \frac{N_{PCSO}}{the\ number\ of\ connections\ in\ P_{CSM}}$$
      v) If $sim_N < T_N$ then check another CS_M's primitive and go to a)
      vi) If $(1 - sim_P) * sim_N$ is greater than previously stored values, store them as a new P_{CSO}'s matching:
      $$P_{sim_{CS}} = (1 - sim_P) * sim_N$$
   b) Modify sim_{cs} value for complex shapes pair CS_O and CS_M:
   $$sim_{CS} = \frac{sum\ of\ P_{sim_{CS}}\ for\ each\ primitive}{number\ of\ primitives\ in\ CS_M}$$
3) For each CS_O choose the CS_M matching using the highest sim_{CS} values. If there are more than one maximum value, use the mutual positions of complex shapes.
4) Compute the graphs similarity coefficient:
   $$sim = \frac{sum\ of\ each\ complex\ shape\ sim_{CS}\ of\ O}{the\ number\ of\ complex\ shapes\ in\ M}$$
5) If $sim < T_{sim}$ then objects are not similar.

IV. Experimental results

The algorithm was firstly evaluated using cars, bicycles and flowers objects. The Fig. 6 and Fig. 7 shows sample images, their graphs and normalized attributes. In the tables, the 'Complex shape' column contains the sequence number (which may be used e.g. as a reference to a specific complex shape). The 'Primitive' column was divided into two sub-columns: 'No' and its 'type'. The 'No' contains the sequence number of a primitive which is unique in the whole image. The

'Attributes' column consists of the 'count' which informs how many sub-primitives constructs the primitive and the 'Values' which is a list of attribute values for each sub-primitive. As mentioned earlier, the attribute is computed as a normalized sub-primitive angle which stores values between 0 and 1. For lines the direction angle is used, for arches the central angle is used.

When comparing the same classes of objects, the sim values should be high. In our tests, for comparison of car 1 with car 2 from Fig. 6 the sim value was equal to 0.75 which means that objects are similar in 75%. Because in both graphs there are two complex shapes containing only one circle (arc with 360°= 1 after normalization), after algorithm execution, there were two candidates of matching for complex shapes no 5 and 7 (in car's 1 graph). In order to choose the best matching, the algorithm checked the mutual positions with other complex shapes. Comparisons with different classes should result with lower sim values. For example when car 1 was compared with a bicycle from Fig. 7 a) and b), the sim value was equal 0.14. For completely different classes, like hepatica flower (Fig. 7 c) and d)), the sim reached 0 value. The hierarchical comparisons enabled faster rejections of not similar complex shapes, which resulted in much lower number of detailed comparisons.

In order to evaluate the algorithm with greater number of images, the prototype application was developed using C++ and OpenCV image processing library. For line detection the Line Segment Detector (LSD) algorithm was used, which is known for providing precise line segment detection results [19]. All detected segments were tested if they can construct one bigger segment and an arc. Moreover, Circular Hough Transform was used to detect circles and improve number of correctly extracted primitives. The images were preprocessed using i.e. morphological operations. After primitives detection, all of them were grouped into complex shapes using minimal distance criterion. During our future research, the primitives extraction algorithm will be refined. As a database, 105 real life images of cars, motorbikes, bicycles and scooters were used. Some images contained background with other objects. In order to evaluate the performance of our algorithm the *precision* and *recall* coefficients were used, defined as following:

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

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

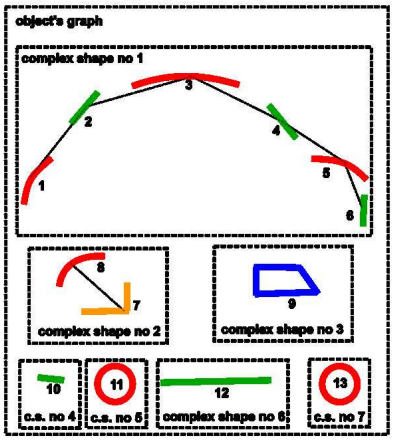
Moreover, the algorithm was also compared with simple region-based method. The precision and recall results for example car, bicycle, scooter and motorbike query images (Fig. 8) are presented in the Table I. It can be seen that our algorithm provided much higher $precision$ values in comparison to the region-based method. For the region-based algorithm, bicycle objects are problematic due to the small uniform color areas which leads to smaller precision values. Contrary, car objects very often contain many big uniform areas and the results are much more precise. Our algorithm is not sensitive to such problems and for both object classes provided high number of correct results. The Query by Shape algorithm provided the best results for the bicycle sketch image, because it contained only the most important primitives for each bicycle. The worst results were obtained for the scooter object because for some bicycle and motorbike objects it obtained enough sim value to qualify them as a correct results.

One of our aim is to provide the algorithm which is able to perform both types of queries: manual (a sketch manually drawn by an user which is then converted into a graph) and automated (graphs automatically detected from input query images) using the same database without any modifications. Because of that fact, we performed also experiments using manually drawn objects and the same database (with the same automatically generated graphs) as in previous tests. For example the bicycle object (Fig. 9) provided results with 0.77 precision.

Since nowadays more and more people are using web searching engines to find specific images, we performed also another tests in order to compare our algorithm with Google Images Search engine. Due to the fact that our Query by Shape system was run on an average performance laptop and Google Images is using huge data centers to process queries, the results cannot be compared directly, but they show differences between approaches. As the query images, we used the same files as for our algorithm (Fig. 8). The results for almost all images were very precise, because the Google Images Search algorithm tries to firstly find the same images and then if none are available it tries to find similar ones. The Google algorithm during query processing uses not only the image data, but also some textual annotations which were chosen to best describe the image. The most precise results were obtained for Mercedes Benz (Fig. 12) and motorbike queries, which was caused by very precise recognized textual descriptions - "mercedes benz s class 1998" and "yamaha 125". However, Google Image Search did not provide the best results for all queries, e.g. the sketch of a bicycle and the scooter. The bicycle sketch was recognized as a "cyclist", but also as a simple, black and white sketch which resulted in only schematic images in the result set (e.g. our Query by Shape provided all types of bicycle images). For the scooter image, the results were much worse. The keywords assigned to the query were "dk raven" and as a result none scooter images were returned but only one type of a specific kind of a bicycle. The tests showed that Google Images Search engine provides very good results for images which are known for it, e.g. previously indexed with proper keywords. For objects which are new, the results may be moderate or even completely incorrect like for the scooter image. Moreover, the Google Image Search engine tries to find the exact object images (for example the same model, color and year of a car). Our algorithm, Query by Shape, tries to find images of objects with the same class (e.g. a car or a bicycle), allowing different

a) car 1 image after all primitives detection

b) car's 1 graph after primitives



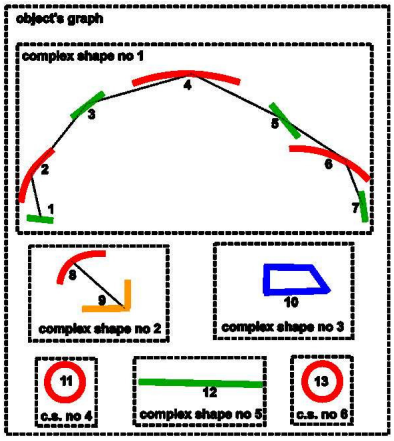
c) car's 1 attributes

Complex shape	Primitive		Attributes	
	No	Type	Count	Values
1	1	arc	-	0,16
	2	line	-	0,14
	3	arc	-	0,22
	4	line	-	0,86
	5	arc	-	0,17
	6	line	-	0,24

2	7	polyline	2	0; 0,25
	8	arc	1	0,31
3	9	polygon	4	0; 0,01; 0,24; 0,85
4	10	line	-	0,8
5	11	Arc	-	1
6	12	Line	-	0,01
7	13	Arc	-	1

d) car 2 image after all primitives detection

e) car's 2 graph after primitives



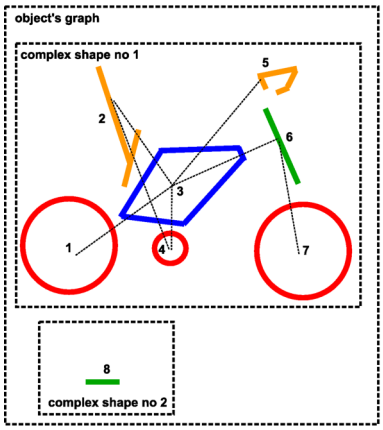
f) car's 2 attributes

Complex shape	Primitive		Attributes	
	No	Type	Count	Values
1	1	Line	-	0,8
	2	arc	-	0,17
	3	line	-	0,11
	4	arc	-	0,22
	5	line	-	0,86
	6	arc	-	0,16
	7	line	-	0,28

2	8	polyline	2	0; 0,25
	9	arc	1	0,31
3	10	polygon	4	0; 0,01; 0,24; 0,84
4	11	Arc	-	1
5	12	Line	-	0
6	13	Arc	-	1

Fig. 6. The example attributes of two compared cars.

a) bicycle graph after primitives extraction

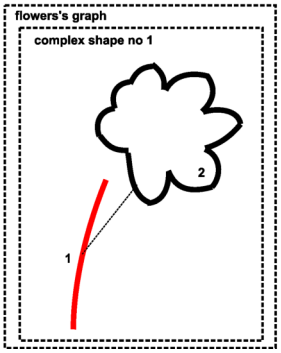


b) bicycle's attributes

Bicycle:

Complex shape	Primitive		Attributes	
	No	Type	Count	Values
1	1	Arc	-	1
	2	polyline	2	0,21; 0,8
	3	polygon	5	0; 0,16; 0,83; 0,87; 0,98
	4	Arc	-	1
	5	polyline	4	0,03; 0,06; 0,68; 0,83;
	6	line	-	0,81
	7	Arc	-	1
2	8	line	-	0

c) flower's graph after primitives



d) flower's attributes

Hepatica flower:

Complex shape	Primitive		Attributes	
	No	Type	Count	Values
1	1	Arc	-	0.08
	2	Polygon arc	2	0.35; 0.36; 0.21; 0.23; 0.17; 0.13; 0.13; 0.24; 0.19; 0.17; 0.16; 0.15

Fig. 7. The example attributes of bicycle and flower objects.

TABLE I
THE PRECISION AND RECALL RESULTS FOR CHOSEN TEST OBJECTS

object	Query by Shape		Region-based	
	precision	recall	precision	recall
car (Fiat 500)	0.89	0.33	0.53	0.75
car (Mercedes Benz)	0.79	0.73	0.51	0.5
bicycle	0.93	0.37	0.23	0.42
bicycle (a sketch)	1.0	0.60	0.28	0.47
motorbike	0.86	0.40	0.75	0.4
scooter	0.67	1.0	0.21	1.0

colors, orientations and other differences in attributes.

V. CONCLUSION AND FUTURE WORKS

In this paper the new CBIR algorithm, using query by approximate shape, was presented. The idea of the method is based on decomposition of shapes into smaller segments - primitives, which are described by theirs attributes. Based on detected primitives, a graph representation of the shape is built, then it is compared with graphs stored in the database. The algorithm is suitable for queries using input image as well as for human-drawn queries. In comparison with our previous research a complete set of primitives was defined, a new graph constructing procedure was used, the matching

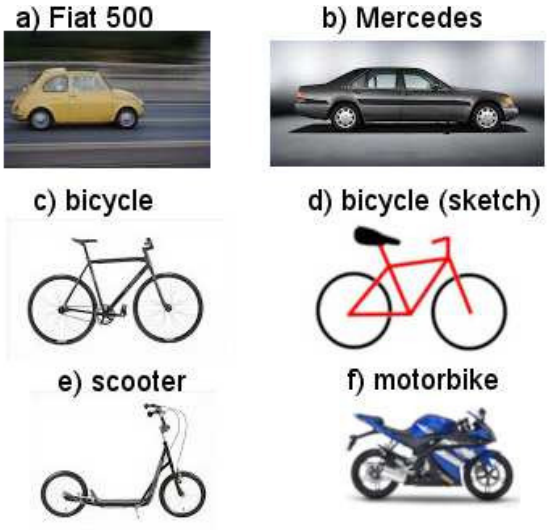


Fig. 8. Example image objects used for tests.

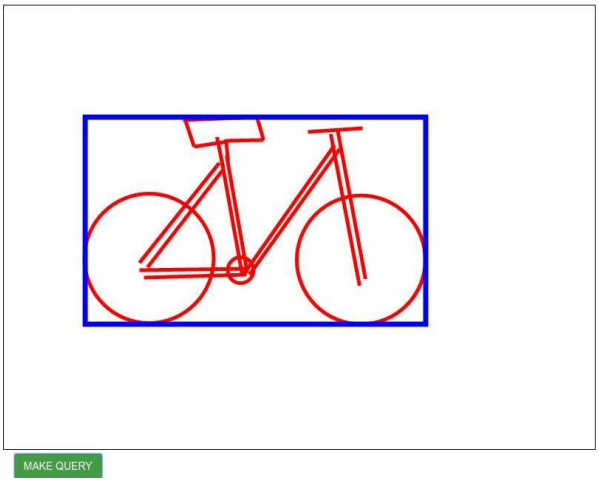


Fig. 9. A bicycle drawn as a manual query. The blue rectangle indicates the detected object during graph constructing stage, the red lines shows lines and arches drawn by an user.

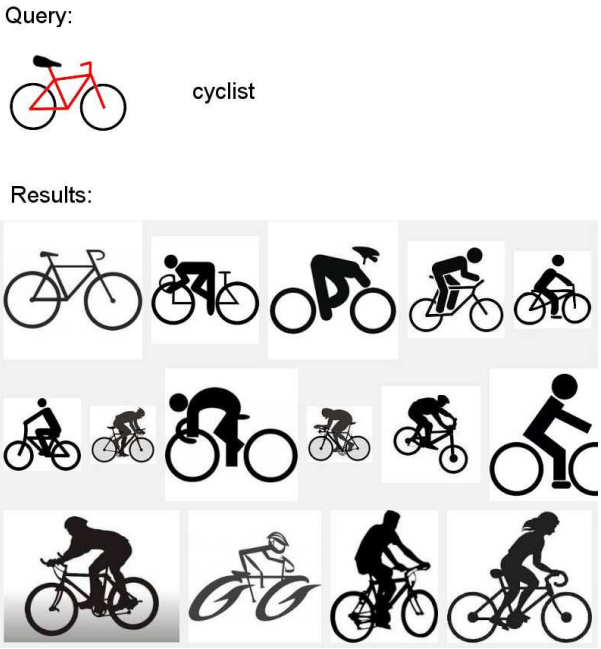


Fig. 10. Bicycle sketch search example using google images.

algorithm was improved and adopted to complex primitives. The experimental results, especially in conjunction with our previous works, are very promising. The main advantage of our approach is that it may be applied to transformed or partially covered objects.

In the future research we will evaluate our approach using greater number of object classes from available image databases, and we also compare the efficiency of our method with more existing state of the art CBIR approaches. Moreover, some modification should be added in order to add ability to achieve better matching e.g. to detect mirrored objects. More advanced graph matching should be also performed, e.g.

Fig. 11. Scooter search example using google images.

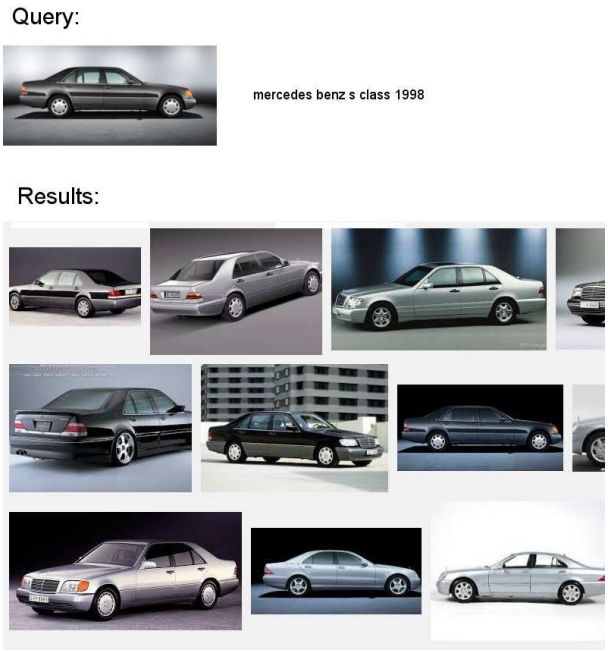


Fig. 12. Mercedes search example using google images.

solving an optimization problem with constraints [18]. Another direction of future research is an efficient storing of objects graphs in the database. Some initial works were performed in [14], but more advanced research should be done.

ACKNOWLEDGMENT

The research used equipment funded by the European Union in the Innovative Economy Programme, MOLAB - Kielce University of Technology.

REFERENCES

[1] H. H. Wang, D. Mohamad, and N. A. Ismail, "Approaches, challenges and future direction of image retrieval" Journal of Computing, vol.2, No.6, 2010, pp. 193-199

[2] R. Datta, D. Joshi, J. Li, J.Z. Wang "Image Retrieval: Ideas, Influences, and Trends of the New Age." ACM Computing Surveys, 40, 2, 2008, 5:1–5:60, doi: 10.1145/1348246.1348248

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

[4] T. K. Shih, "Distributed multimedia databases" T. K. Shih, Ed. Hershey, PA, USA: IGI Global, ch. Distributed Multimedia Databases, 2002, pp. 2–12

[5] H.-P. Kriegel, P. Kroger, P. Kunath, and A. Pryakhin, "Effective similarity search in multimedia databases using multiple representations" in 12th International Multi-Media Modelling Conference Proceedings, 2006, pp. 389–392, doi: 10.1109/MMMC.2006.1651355

[6] 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, 2008, pp. 462–469, doi: 10.1109/CBMI.2008.4564983

[7] M. Bielecka and M. Skomorowski, "Fuzzy-aided parsing for pattern recognition" in Computer Recognition Systems 2, ser. Advances in Soft Computing, M. Kurzynski, E. Puchala, M. Wozniak, and A. Zolnierek, Eds. Springer Berlin Heidelberg, vol. 45, 2007, pp. 313–318, doi: 10.1007/978-3-540-75175-5_39

[8] 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, 1992, pp. 530–533, doi: 10.1109/ICPR.1992.201616

[9] J. F. Nunes, P. M. Moreira and J. M. R. S. Tavares, "Shape based image retrieval and classification", 5th Iberian Conference on Information Systems and Technologies (CISTI), 2010

[10] D. Zhang, G. Lu "Shape-based image retrieval using generic Fourier descriptor" Signal Processing: Image Communication. 17, 10, 2002, pp. 825–848

[11] C. E. Jacobs, A. Finkelstein, D.H. Salesin "Fast Multiresolution Image Querying" Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, 1995, pp. 277–286

[12] S. Deniziak, T. Michno "Query by Shape for Image Retrieval from Multimedia Databases." Communications in Computer and Information Science, Springer, 521, 2015, pp. 377–386, doi: 10.1007/978-3-319-18422-7_33

[13] S. Deniziak, T. Michno "Query-by-Shape Interface for Content Based Image Retrieval" 8th IEEE International Conference on Human System Interactions (HSI), 2015, pp. 108–114, doi: 10.1109/HSI.2015.7170652

[14] S. Deniziak, T. Michno, A. Krechowicz "The Scalable Distributed Two-layer Content Based Image Retrieval Data Store" 8th International Symposium on Multimedia Applications and Processing, Federated Conference on Computer Science and Information Systems (FedCSIS), 2015, pp. 827–832, doi: 10.15439/2015F272

[15] Chao Ma, Xiaokang Yang, Chongyang Zhang, Xiang Ruan, and Ming-Hsuan Yang, "Sketch Retrieval via Local Dense Stroke Features" Image and Vision Computing (IVC), 2016, doi: 10.1016/j.imavis.2015.11.007

[16] R. Krishnamoorthy, , S. Sathiya Devi, "Image retrieval using edge based shape similarity with multiresolution enhanced orthogonal polynomials model" Digital Signal Processing, Volume 23, Issue 2, March 2013, pp. 555–568 , doi: 10.1016/j.dsp.2012.09.018

[17] A. S. Mouratoa, R. Jesus, "Clip art retrieval using a sketch. Tablet application." Conference on Electronics, Telecommunications and Computers – CETC 2013, doi: 10.1016/j.protcy.2014.10.246

[18] P. Sitek, J. Wikarek, "A Hybrid Programming Framework for Modeling and Solving Constraint Satisfaction and Optimization Problems." Scientific Programming, vol. 2016, Article ID 5102616, 13 pages, 2016. doi:10.1155/2016/5102616

[19] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, "LSD: a Line Segment Detector", Image Processing On Line, 2 (2012), pp. 35–55, doi: 10.5201/ipol.2012.gjmr-lsd