Raport Badawczy
Research Report

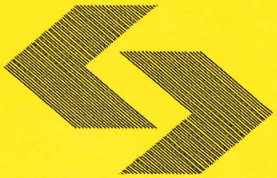
RB/41/2009

Graphical object indexing and query by image as an aspect of Content-Based Image Retrieval System

T. Jaworska

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 3810100

fax: (+48) (22) 3810105

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Janusz Kacprzyk

Warszawa 2009

Graphical object indexing and query by image as an aspect of Content-Based Image Retrieval System

Tatiana Jaworska

Systems Research Institute, Polish Academy of Sciences
01-447, 6 Newelska Street, Warsaw, Poland
Tatiana.Jaworska@ibspan.waw.pl

Abstract. In this article we propose graphical object indexing used for image matching in the Content-Based Image Retrieval (CBIR) system containing colour images. The part devoted to image processing and the inner structure of the database are signalled to the extent which is necessary for the reader to understand how the whole system works. Firstly, we discuss the theoretical construction of indexes for the image objects. The indexes are based on attribute/feature vectors for each object and spatial relationships among objects for image retrieval. Secondly, we address the problem of the graphical query by example. In order to construct the graphical query we implement the user interface (GUI) which has been developed in the light of human-computer interaction. GUI enables the user to design their own image which is further treated as a query for the database. The expected reply is a set of similar images presented to the user by the database.

Keywords: CBIR, image indexing, query by image, human-computer interaction, graphical user interface

1 Introduction

In recent years, the availability of image resources on the WWW has increased tremendously. This has created a demand for effective and flexible techniques for automatic image retrieval. Although attempts to perform the Content-Based Image Retrieval (CBIR) in an efficient way, that is based on shape, colour, texture and spatial relations, have been

made, the CBIR system has yet to reach maturity. A major problem in this area is computer perception. In other words, there remains a considerable gap between image retrieval based on low-level features, such as shape, colour, texture and spatial relations, and image retrieval based on high-level semantic concepts, for example, houses, windows, roofs, flowers, etc. This problem becomes especially challenging when image databases are exceptionally large.

Given the above context it comes as no surprise that fast retrieval in databases has recently been an active research area. The effectiveness of the retrieval process is increased by an index scheme. Information retrieval is also very closely connected with another problem, namely, how to effectively put an image query for the CBIR system. We would like to analyse these two aspects of CBIR in this article.

1.1 Indexing Background

Most of the CBIR systems adopt the following two-step approach to search image databases [1]:

1. (indexing) an attribute/feature vector capturing certain essential properties of the image is computed and stored in a feature base for each image in a database;
2. (searching) the system, given a query image, computes the image feature vector and compares it to the feature vectors in the feature base. As a result images most similar to the query image are returned to the user.

For the classical retrieval system to be successful, the feature vector $f(I)$ for an image I should have the following qualities:

1. $|f(I) - f(I')|$ should be large if and only if I and I' are dissimilar;
2. $f(\cdot)$ should be fast to compute;
3. $f(I)$ should be small in size.

Colour histograms, defined in the above way, were commonly used as feature vectors by some authors [2, 3, 4, 5] some others used a colour correlogram [6].

In 2001 a set of MPEG-7 descriptors was introduced. These descriptors are more complicated as they encompass colour descriptors (colour layout, colour structure, dominant colour and scalable colour), texture descriptors (edge histogram and homogeneous texture) and shape descriptors (contour and region) [7]. The MPEG-7 has been accepted as a standard and is used in some applications. Unfortunately,

it neglects important criteria for the assessment of image similarity, such as spatial information and spatial relationships.

Some authors used hierarchical semantics and hierarchical cluster indexes [8].

However, our system takes into account not only low-level features but object identification in the human sense and mutual location of objects in the image as well. We also highlight the fact that the whole system, whose latest fragments are being currently completed, is intended to be entirely automatic.

1.2 The Background to Querying by Image

A query by image allows users to search through databases to specify the desired images. It is especially useful for databases consisting of very large numbers of images. Sketches, layout or structural descriptions, texture, colour, sample images, and other iconic and graphical information can be applied in this search.

An example query might be: *Find all images with a pattern similar to this one*, where the user has selected a sample query image. More advanced systems enable users to choose as a query not only whole images but also some objects. The user can also draw some patterns consisting of simple shapes, colours or textures. In the QBIC system [3] the images are retrieved based on the above-mentioned attributes separately or using distance functions between features. Tools in this GUI include some basic objects such as: polygon outliner, rectangle outliner, line draw, object translation, flood fill, eraser, etc.

2 CBIR Concept Overview

The purpose of this paper is to present an indexing method which retrieves images based on the two-level indexing system, according to a query by image. The dedicated GUI has been developed to enable the user to put such a graphical query. In general, the system consists of 4 main blocks (Fig. 1):

1. the image preprocessing block (responsible for image segmentation) applied in Matlab;

2. the Oracle Database, storing information about whole images, their segments (here referred to as graphical objects), segment attributes, object location, pattern types and object identification;
3. the indexing module responsible for the two-level image indexing procedure;
4. the graphical user's interface (GUI), also applied in Matlab.

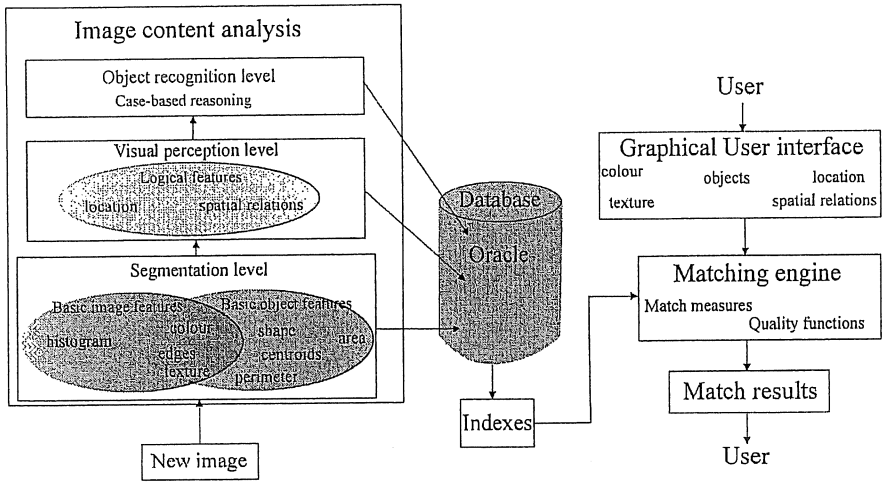


Fig. 1. Block diagram of our content-based image retrieval system.

Our CBIR system consists of photos, such as images of landscapes or houses, downloaded from the Internet in the JPEG format. To be effective in terms of the presentation and choice of images, the system has to be capable of finding the graphical objects that a particular image is composed of. For example, in a colour image of a house, the system can extract some specific architectural elements, such as windows, roofs, doors, etc.

Figure 1 shows the block diagram of our CBIR system. As can be seen, the left part of the diagram illustrates the image content analysis block of our system. In this approach we use a multi-layer description model. The description for a higher layer could be generated from the description of the lower layer, and establishing the image model is synchronized with the procedure for progressive understanding of image contents. These different layers could provide distinct information on the image content, so this model provides access from different levels. Based on the above statements, the multi-layer

description model MDM could be represented by the following formula:

$$MDM = \{LLV, OL, SRR\} \quad (1)$$

where: LLV – lowest layer vector, OL – object location, SRR – spatial relationship rules for object recognition. All these three layers together form a procedure of multi-layer image content analysis [9]. This analysis leads to the extraction of semantically significant objects in an image.

According to a human visual perception theory, during the visual perception and recognition process, human eyes successively fixate on the most informative parts of an image [10]. These informative parts, called meaningful regions, possess certain semantic meanings.

The information obtained from the image content analysis is stored in the database. In the diagram the indexes block is deliberately kept apart as an important element of the system.

The right part of figure 1 is dedicated to users and presents the on-line functionality of the system. Its first element is the GUI block. In comparison to the previous systems, ours has been developed in order to give the user the possibility to design their image which later becomes a query for the system. If users have a vague target image in mind, the program offers them tools for composing their imaginary scenery. Moreover, the system presents them with some optional sceneries, for instance, houses, forest, based on particular chosen elements. GUI details are presented in subsection 4.

The next element of the system is the matching engine, which uses indexes based on the multi-layer description model to search for "the best matching images". The details of index construction and the matching procedure are presented below. Retrieval results are presented by the user's interface.

2.1 Implementation Remarks

Each new image added to the CBIR system, as well as the user's query, must be preprocessed. This process is presented in the image content analysis block as a segmentation level frame (left, Fig. 1). All graphical objects (such as houses, trees, a beach, the sky etc.) must be segmented and extracted from the background at the stage of preprocessing.

Although colour images are downloaded from the Internet, their preprocessing is unsupervised. An object extraction from the image background must be done in a way enabling unsupervised storage of these objects in the DB.

For this purpose, we apply two-stage segmentation, enabling us to accurately extract the desired objects from the image. In the first stage, the image is divided into separate RGB colour components and these components are next divided into layers according to three light levels. In the second stage, individual graphical objects are extracted from each layer. Next, the low-level features are determined for each object, understood as a fragment of the entire image. These features include: colour, area, centroid, eccentricity, orientation, texture parameters, moments of inertia, etc. (see Fig. 2). The segmentation algorithm and object extraction algorithm, as well as texture parameters finding algorithm are presented in detail in an article by Jaworska [11].

3 The Indexing Scheme

3.1 Data Representation for Objects

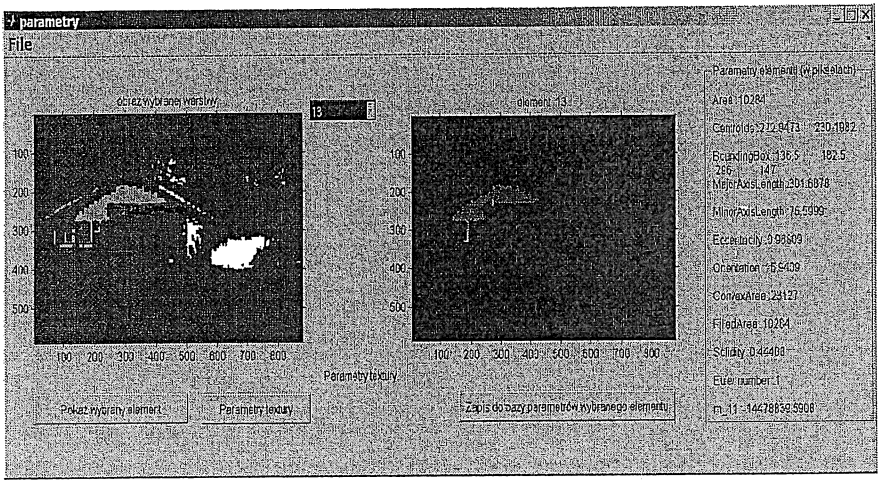


Fig. 2. (Left) One colour layer from which the object was extracted. (Centre) Presentation of a separate object. (Right) Object attributes.

Each object, selected according to the algorithm presented in detail in [11], is described by some low-level features also called attributes. The attributes describing each object include: average colour k_{av}, texture parameters T_p, area A, convex area A_c, filled area A_f, centroid $\{x_c, y_c\}$, eccentricity e, orientation α, moments of inertia m_{11}, bounding box $\{b_1(x,y), ..., b_s(x,y)\}$ (s – number of vertices), major axis length m_{long}, minor axis length m_{short}, solidity s and Euler number E. These attributes are presented in the example window of the interface (Fig. 2) for a selected object. Let F be a set of attributes where

$$F = \{k_{av}, T_p, A, A_c, \ldots, E\}.$$

For ease of notation we will use $F = \{f_1, f_2, \ldots, f_r\}$, where r – number of attributes. For an object, we construct a feature vector O containing the above-mentioned features - attributes:

$$O = \begin{bmatrix} O(k_{av}) \\ O(T_p) \\ O(A) \\ \vdots \\ O(E) \end{bmatrix} = \begin{bmatrix} O(f_1) \\ O(f_2) \\ O(f_3) \\ \vdots \\ O(f_r) \end{bmatrix}. \tag{2}$$

The average colour is a complex feature. It means that values of the red, green and blue components are summed up for all the pixels belonging to an object, and divided by the number of object pixels:

$$k_{av} = \{r_{av}, g_{av}, b_{av}\} = \left\{ \frac{\sum_{m=1}^{n} r_m}{n}, \frac{\sum_{m=1}^{n} g_m}{n}, \frac{\sum_{m=1}^{n} b_m}{n} \right\}. \tag{3}$$

The next complex feature attributed to objects is texture. Texture parameters are found in the wavelet domain (the Haar wavelets are used). The algorithm details are also given in [11]. The use of this algorithm results in obtaining two ranges for the horizontal object dimension h and two others for the vertical one v:

$$T_p = \begin{Bmatrix} h_{\min_{1,2}}; h_{\max_{1,2}} \\ v_{\min_{1,2}}; v_{\max_{1,2}} \end{Bmatrix}. \tag{4}$$

3.2 Pattern Library

The pattern library contains information about pattern types, shape descriptors, object location and allowable parameter values for an object. We define a model feature vector P_k for each graphical element. We assume weights μ_P characteristic of a particular type of element which satisfy:

$$\mu_{P_k}(f_i) \in [0,1] \tag{5}$$

where: $1 \le i \le r$, k – number of patterns. These weights for each pattern component should be assigned in terms of the best distinguishability of patterns.

First, each graphical extracted object is classified into a particular category from the pattern library. For this purpose, in the simplest case, we use an L_m metric, where the distance between vectors O and P_k in an r-dimensional feature space is defined as follows:

$$d(O,P_k) = \left\lceil \sum_{i=1}^{r} \mu_{P_k}(f_i) \left| O(f_i) - P_k(f_i) \right|^m \right\rceil^{1/m} \tag{6}$$

where: k – pattern number, $1 \le i \le r$, m is the order of the metric. For $m = 1$ and for $m = 2$, it becomes the Manhattan and the Euclidean distance, respectively.

In order to improve the retrieval efficiency, object attributes can be described by applying fuzzy sets [12]. Hence, the state of an object is reflected by a set of values that corresponds to the set of fuzzily described attributes. We can divide the types of data according to their complexity. Table 1 shows a description of the different types of attribute values that we consider in our approach. Figure 3 exemplifies different types of attribute values corresponding to a graphical object. As can be seen in this figure, the description of the graphical object's state can be composed of precise or imprecise values, objects and collections.

In the fuzzy set description our weights μ_P correspond to a membership function. Then, for the most important attributes of a graphical object we can assume $\mu_p(f_i) = 1$. For instance, if we compare objects with a similar shape we use the number of vertices s as one of the attributes. First, objects with the same number of vertices s (or s - 1) of bounding boxes are presumed the most similar to each other. If the differences in vertices are greater, the weight decreases down to 0,

$\mu_p(b_i) \geq 0$ in the bounding boxes case, and it means that object shapes are not similar.

Table 1. Different kinds of feature - attribute values [13].

Type	Description
Precise values	Classical basic classes that usually appear in an object-oriented data model
Imprecise values	Different types of labels, according to fuzzily described imprecise value.
Objects	Including references to other complex objects
Collections	Sets of objects. These sets may be fuzzy and/or the elements of the sets may be fuzzy values (even fuzzily described objects)

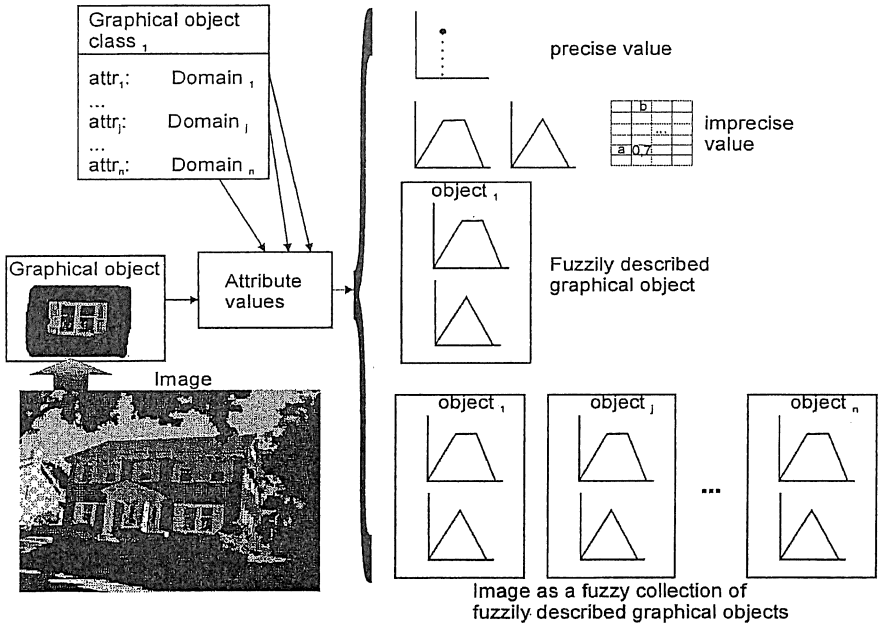


Fig. 3. Different kinds of attribute values for the graphical object description.

Generally, if a membership function $\mu_p(f_i) \rightarrow 0$ for any attribute, this attribute plays a less important role in an object comparison. For a given object, if we find the minimum distance d from (6) or we obtain

the best matching based on a fuzzy set comparison, we can assign this object to a pattern and label it as t_{O_k}. This label is stored in the DB as an additional object parameter. In fact, this assignment of labels which are semantic names for the graphical objects overcomes the gap that has separated low-level image features from high-level semantic concepts and that has so far perplexed the CBIR system creators.

3.3 Spatial Object Location as a Global Feature

Object classification into a particular category is not enough for full image identification. There is also a need to assign a global feature to an image to make indexing more efficient. Chow, Rahman and Wu [14] proposed a tree-structured image representation, where a root node contains the global features, while child nodes contain the local region-based ones. This approach hierarchically integrates more information on image contents to achieve better retrieval accuracy compared with global and region features individually. The next step is an examination of mutual relationships of objects and object position in the whole image. Wang [9] proposed spatial relationships and similarity retrieval using a minimum bounding rectangle and a 2D Bε -string model.

In our system the spatial object location in an image is used as the global feature. Firstly, it is easy to recognize this spatial location visually by the user. Secondly, it supports full identification based on rules for location of graphical elements. Let us assume that we analyse a house image. Then, for instance, an object which is categorized as a window cannot be located over an object which is categorized as a chimney. For this example, rules of location mean that all architectural objects must be inside the bounding box of a house. For an image of a Caribbean beach, an object which is categorized as a palm cannot grow up from the middle of the sea, and so on. For this purpose, the mutual position of all objects is checked. The location rules are also stored in the pattern library [15]. Thirdly, object location reduces the differences between high-level semantic concepts perceived by humans and low-level features interpreted by computers.

In our case spatial information, namely the object's mutual relationships, are presented as a vector F_g for the global feature:

$$F_g = \begin{bmatrix} (x_{c_1}, y_{c_1}), t_{o_1} \\ (x_{c_2}, y_{c_2}), t_{o_2} \\ \vdots \\ (x_{c_N}, y_{c_N}), t_{o_N} \end{bmatrix} \tag{7}$$

where: $\{x_{c_i}, y_{c_i}\}$ is an object centroid and N – number of all objects in the image, t_{O_k} – an object label assigned in the process of identification. As you can see in figure 5, we analyse mutual spatial location for particular types of objects.

4 GUI for Query by Image

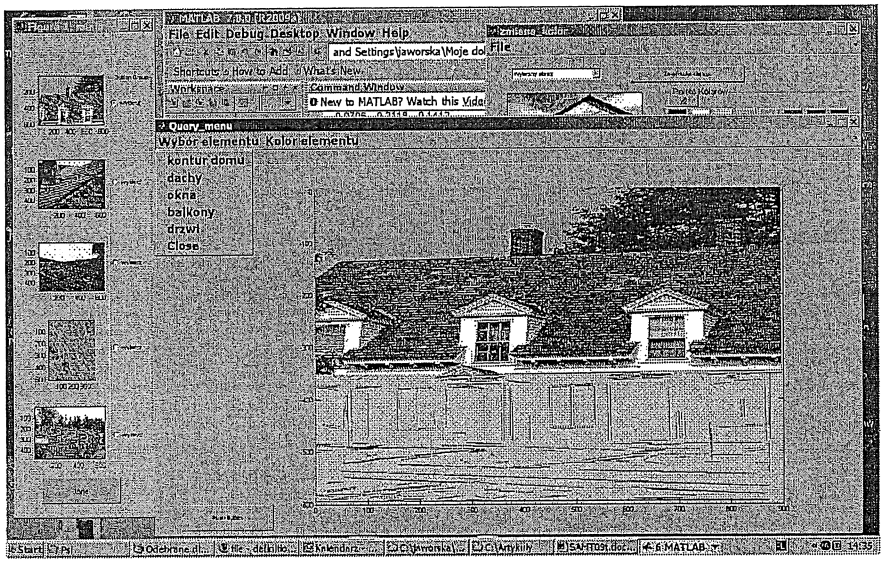


Fig. 4. The user menu applied by the system to design a query by image. The left window is used to present graphical elements, for example house roofs. It is easy to notice that the first roof from the top of the list of miniatures on the left is chosen and located in the house outline.

Graphical User Interface (GUI) is an immanent element of our system. Drawing on the latest findings in the area of the human-computer interaction, we have made an effort to create a useful tool for the user

who is interested in designing their own image. This design is treated as a query by image. Fig. 4 presents the main GUI window entitled "Query_menu". From the left window the user can choose the image outlines which become visible in an enlarged form in the main window.

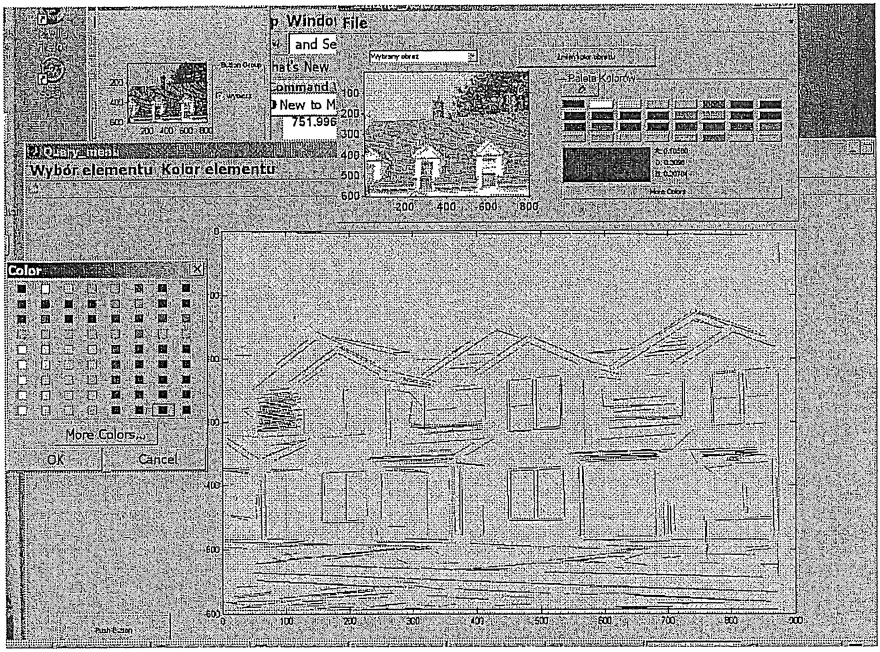


Fig. 5. Menu tools dedicated to changing element colour. When the user selects a graphical element from the window containing miniatures, he can open the "zmiana_koloru" window in order to change a colour of this element. If the basic colour pallet is not enough, the user can open the "more colours" window. Having determined the element colour, the user locates the element in the appropriate position in the image outline.

Next, the user chooses particular graphical elements from subsequent menus and situates them on the appropriate location in the chosen outline. For each element the user can change its colour (see Fig.5). Moreover, there is a window for changing the texture of an element, if it has one, or adding a texture for non-textured components. For a texture the user can also choose its colour.

For more advanced users, there are additional options in the query interface which enable them to select the most interesting feature. These preferences are implemented in the system as weights μ_{qO} which are to be taken into account during the final matching. This fact is especially important if we use fuzzily described object attributes. Then we compare a query object with a feature vector $O_q = \mu_{q_{Oi}}(f_i)$ to objects stored in the DB.

After the designing process, the image is sent as a query to DB and all CBIR retrieval rules are applied to it. The GUI is strictly dedicated to the CBIR system and consists of the most important components only. In further work some additional menus will be added if a need to improve the retrieval process arises.

5 Image Matching Strategy

Image matching is conducted with the aid of object recognition and spatial relationships. Query image $Q = \{F_{gq}, O_{q_1}, ..., O_{q_N}\}$ consists of a global feature vector F_{gq} and object feature vectors for all objects O_{q_k}, where $1 \leq k \leq N$. First, the relevant images $R = \{F_{gR}, O_{R_1}, ..., O_{R_N}\}$ with N objects are searched for in the database. Next, we check if objects have the same pattern t_{R_k}. If the answer is positive, then the global feature vectors F_{gq} and F_{gR} are compared. Their similarities are searched for based on mutual object locations in the images.

This means that objects are not matched based upon fixed positions in the image. For example, as you can see in fig. 6, object $O(t_1)$ is to the left of object $O(t_4)$. This information is collected and stored in tables as a global feature. For matching images Q and R, whose spatial information is illustrated in tables 2 and 3, we compare each table cell. The notation used in the tables is as follows: l - object $O(t_1)$ is to the left of object $O(t_2)$, p - object $O(t_1)$ is to the right of object $O(t_2)$, d - object $O(t_1)$ is below object $O(t_2)$, g - object $O(t_1)$ is above object $O(t_2)$.

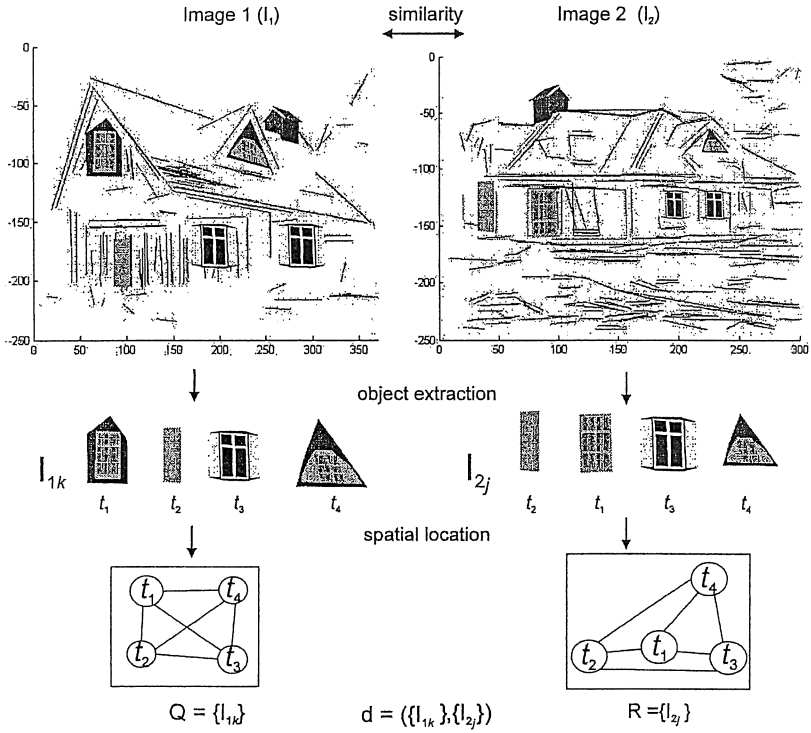


Fig. 6. Model of the spatial object location described as a global vector F_g. For each object t_{O_k} we know its feature vector $O_k(f_i)$.

Table 2. Spatial information for query image Q from Fig. 6.

	t_1	t_2	t_3	t_4
t_1	0	d	l, g	l
t_2	g	0	l	l, d
t_3	p, d	p	0	d
t_4	p	p, g	g, l	0

We assume a strong constraint that the tables are well matched if all cells contain the same information. Only if these tables are well matched, is the relevant image sent as a result of the matching process.

Table 3. Spatial information for relevant image R from Fig. 6.

	t_1	t_2	t_3	t_4
t_1	0	p	l	l, d
t_2	l	0	l	l, d
t_3	p	p	0	l, d
t_4	p, g	p, g	p, g	0

5.1 Discussion

In the case of a lack of relevant images the user can decide if spatial information is the most important for them. If the objects are more important, we can limit the matching procedure only to check the local feature vectors, restricted to $Q = \{O_{q_1}, ..., O_{q_N}\}$.

We can also imagine a situation in which the user's preferences enable us to impose weaker constraints on object matching. In this case, we can only check a global feature vector F_g.

Some confusion can set in when we attempt to find a picture which is, for instance, a half of another picture. However, matching here is also possible; the object location table is matched to a fragment of a table for the entire required image.

The more complex situation is when we analyse images of the same scene taken from different directions. Then, the objects (like trees, buildings, or statues, for example) are nearly the same, but their spatial relationships vary.

The situation can become even more ambiguous (depicted in figure 6) when the user, designing their query image, selects objects belonging to different images. Then, our program, looking for relevant images, will have to use methods of comparisons between fuzzy collections.

6 Conclusions and Further Works

The construction of a CBIR system requires combining two systems: the image processing module for automatic segmentation and the database to store the generated information about images and their

segments. Having built these two elements of the system, we faced the problem of image retrieval. We dealt with it by constructing and describing the object pattern library. Object patterns are used for the optimum object distinction and identification.

The image indexing method employed so far in the image retrieval in our system is still rather rough. Now, we are on the point of applying fuzzily described objects which makes our system more efficient and sophisticated.

In our CBIR system we propose the new GUI specially dedicated to designing a graphical query by the user. Hitherto, the author has not encountered any papers reporting a user-designed graphical query by example and, in this respect, the method described above is our original contribution. The formulation of the indexing system enables us to retrieve images in the preliminary stage. Thanks to the algorithm adopting comparisons between fuzzy collections which is currently under construction, the system will be able to accept user's preferences more flexibly.

Furthermore, the results of this initial study have to be verified with the use of a large number of a different kind of images involving long-term usage of the system in practice.

To sum up, even though we have experienced a few snags, all our actions have led to the creation of a user-friendly system. In the nearest future we hope to apply a more sophisticated semantic analysis so that the user will not experience the roughness of the system.

References

1. Swain M., Stricker M.: The capacity of colour histogram indexing, In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 704-708 (1994)
2. Swain M., Ballard D.: Color indexing, Int. J Com. Vision, Vol. 7 No. 1, pp. 11-32 (1991)
3. Flickner M., Sawhney H., et al.: Query by Image and Video Content: The QBIC System, IEEE Computer **28**, No. 9, August, pp. 23 -32 (1995)
4. Ogle V., Stonebraker M.: CHABOT: Retrieval from a Relational Database of Images, *IEEE Computer* **28**, No. 9, August, pp. 40 -48 (1995)

5. Pentland A., Picard R., Sclaroff S.: Photobook: Content-based manipulation of image databases. International Journal of Computer Vision, Vol. 18, No. 3, 233- 254 (1996)
6. Huang J. et al. Spatial Color Indexing and Applications. International Journal of Computer Vision, Vol 35, No. 3, Kluwer Academic Publishers, the Netherlands, pp. 245- 268 (1999)
7. ISO/MPEG N6828, Overview of MPEG-7, ver. 10, (ed.) Martinez J. M., Palma de Mallorca, October (2004)
8. Shi Z., He Q., Shi Zh.: An index and retrieval framework integrating perceptive features and semantics for multimedia databases. Multimed Tools Appl, Vol. 4, pp. 207—231 (2009)
9. Wang Y. H.: A Spatial Relationship Method Supports Image Indexing and Similarity Retrieval. chap. 12, In: Multimedia Systems and Moment-Based Image Retrieval, (ed.) Deb S., IGP. Melbourne, pp. 277—301. (2004)
10. Newman W. M., Lamming M. G.: Interactive System Design. Addison-Wesley, Harlow (1996)
11. Jaworska T.: Object extraction as a basic process for content-based image retrieval (CBIR) system. In: Opto-Electronics Review, Asso. of Polish Electrical Engineers (SEP), Vol. 15, No. 4, Warsaw, pp. 184-195 (2007)
12. Zadeh L. A., From Computing with Numbers to Computing with Words – from Manipulation of Measurements to Manipulation of Perception. IEEE Transactions on Circuits and Systems. In: Fundamental Theory and Applications, Vol. 45, pp. 105-119 (1999)
13. Kacprzyk J., Berzal F., et al.: A general framework for computing with words in object-oriented programming. In: International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 15, Supplement, Feb., (ed) Bouchon-Meunier B., pp.111-131 (2007)
14. Chow T. W., Rahman M. K., Wu S.: Content-based image retrieval by using tree-structured features and multi-layer self-organized map. Pattern Anal Applic. Vol. 9, pp. 1—20 (2006)
15. Jaworska T.: Analysis of Object Features in Terms of the Dissimilarity of Pattern Recognition. In: Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Applications. Vol. II, Atanassov K. et al. (eds.) EXIT, Warsaw, pp. 79-96 (2008)