TATIANA JAWORSKA
Instytut Badań Systemowych, PAN
Ul. Newelska 6, Warszawa

GRAPHICAL OBJECT CLASSIFICATION AND QUERY BY IMAGE AS AN ASPECT OF CONTENT-BASED IMAGE RETRIEVAL SYSTEM

Abstract

In this article we propose graphical object classification 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 graphical object classification. The indexes are based on feature vectors for each object and spatial relationships among objects for image retrieval. We present some classes and classfied objects. 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 classification, 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 mining of graphical data 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, beaches, 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 [13]:

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 [12, 2, 7, 10] some others used a colour correlogram [3].

In 2001 a set of MPEG-7 descriptors was introduced, and as a standard, is used in some applications. 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) [4]. 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 [11].

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.

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 [6] 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 highlight the two-level image indexing procedure and image classification that further is used for images retrieval, 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 four main blocks (Fig. 1):

1. the image preprocessing block (responsible for image segmentation) applied in Matlab with the support of some spaecially dedicated toolboxes;
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 and image classification;
4. the graphical user's interface (GUI), also applied in Matlab.

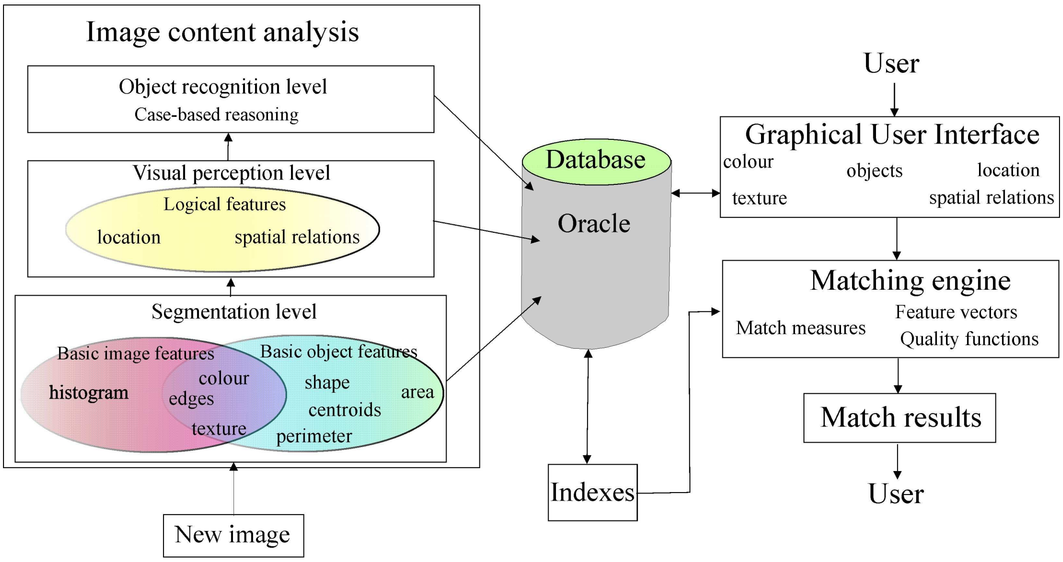

Fig. 1. *Block diagram of our content-based image retrieval system.*

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 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.

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 based on the image matching strategy (see Sec. 5) searches 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. The segmentation algorithm and object extraction algorithm, as well as texture parameters finding algorithm are presented in detail in an article by Jaworska [5].

3. The Indexing Scheme

3.1. Data Representation for Objects

Each object, selected according to the algorithm presented in detail in [5], 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.

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:

$$O = \begin{bmatrix} O(k_{av}) \\ O(T_p) \\ O(A) \\ \vdots \\ O(E) \end{bmatrix} = \begin{bmatrix} O(f_1) \\ O(f_2) \\ O(f_3) \\ \vdots \\ O(f_r) \end{bmatrix}. \qquad \mathbf{(1)}$$

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:

$$k_{av} = \{r_{av}, g_{av}, b_{av}\} = \left\{ \frac{\sum_{m=1}^{n} r_m}{n}, \frac{\sum_{m=1}^{n} g_m}{n}, \frac{\sum_{m=1}^{n} b_m}{n} \right\}. \quad \textbf{(2)}$$

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 [5]. The use of this algorithm results in obtaining two ranges for the horizontal object dimension *h* and two others for the vertical one *v*:

$$T_p = \begin{Bmatrix} h_{\min_{1,2}}; h_{\max_{1,2}} \\ v_{\min_{1,2}}; v_{\max_{1,2}} \end{Bmatrix}. \quad \textbf{(3)}$$

3.2. Pattern Library

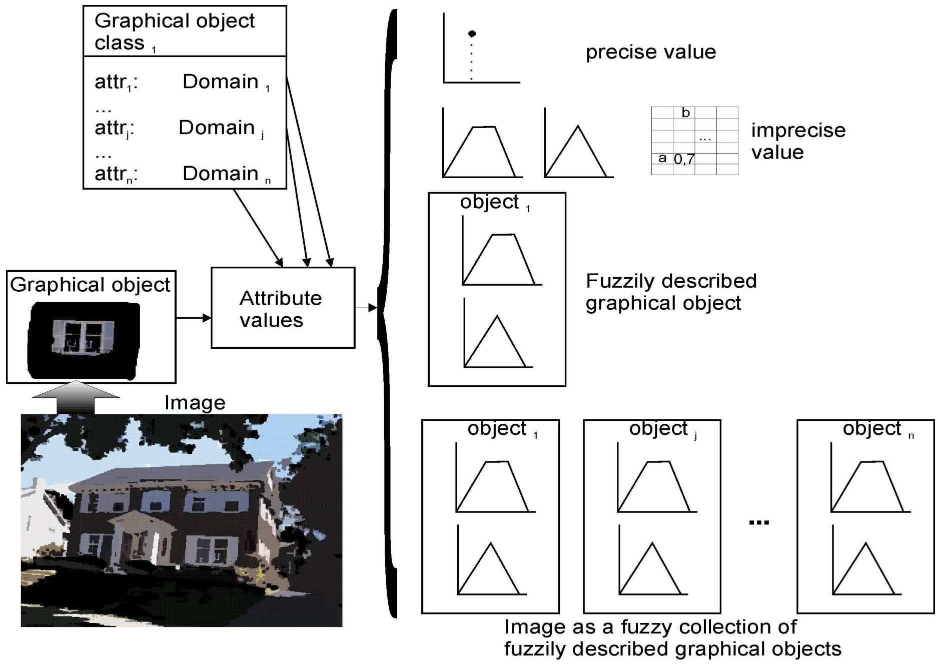

Fig. 2. *Different kinds of attribute values for the graphical object description.*

The pattern library [6] 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:

$$\mu_{P_k}(f_i) \in [0,1] \tag{4}$$

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:

$$d(O,P_k) = \left[\sum_{i=1}^{r} \mu_{P_k}(f_i) \left|O(f_i) - P_k(f_i)\right|^m\right]^{1/m} \tag{5}$$

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 [15]. 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 2 exemplifies different types of attribute values [7] 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, $\mu_p(b_i) \ge 0$ in the bounding boxes case, and it means that object shapes are not similar.

Generally, if a membership function $\mu_p(f_i) \to 0$ 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 eq. 5 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. Object Classification Results

In our task, for exemplification we chose patterns for door, glass pane and window frame models distinguished from other objects for house images. We used the classification tree for eight attribute data of an object. These attributes are: eccentricity, moments of inertia, solidity, minor axis length, major axis length, orientation and average colour RGB components. We had to normalize all data to [0, 1] to be able to compute distances of vectors from the particular class pattern and later compare objects with each other. Objects are assigned to a class according to patterns and a weight for each feature.

Table 1 presents values of attributes for the three above-mentioned classes, with weights for each feature.

Table 1. *Patterns for the door, glass pane and window frame models based on the most distinguishable features.*

Features	Pattern door	Weight μ_P	Pattern pane	Weight μ_P	Pattern frame	Weight μ_P
Eccentricity	0,93	0,1	0,85	0,1	0,57	0,01
Moments of inertia	average	0,01	average	0,01	average	0,01
Solidity	0,8	0,3	0,9	0,19	0,369	0,29
Minor axis length /Major axis length	0,427	0,1	0,5	0,1	0,8	0,2
Orientation	0,99	0,46	0,99	0,3	0,47	0,05
Average colour component R	0,33	0,01	0,15	0,1	0,93	0,05
Average colour component G	0,217	0,01	0,22	0,1	0,92	0,05
Average colour component B	0,33	0,01	0,12	0,1	0,95	0,05

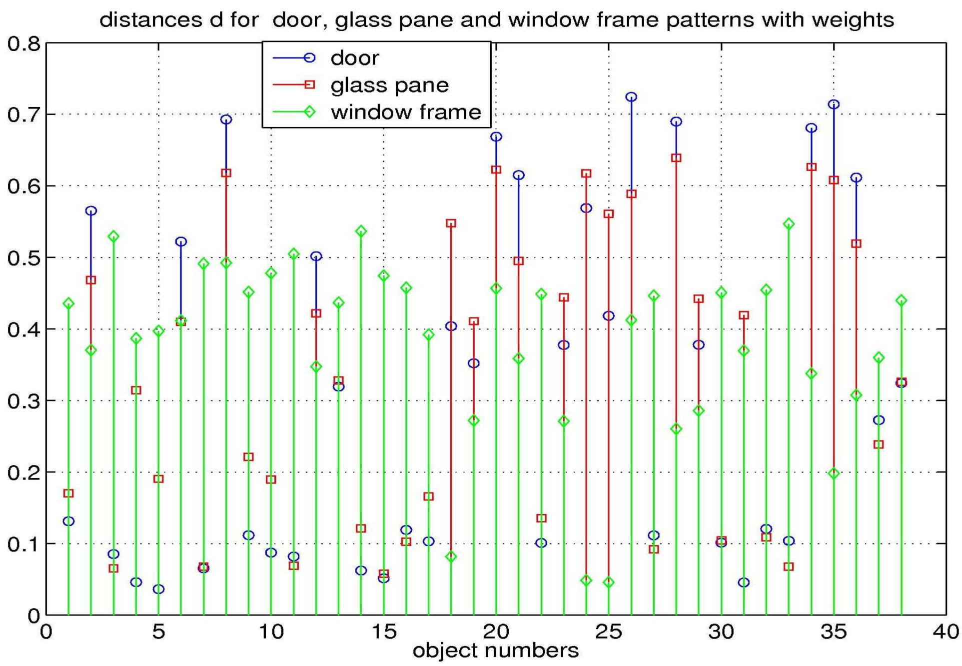

Fig. 3. *Distances* d *for all of 38 graphical objects computed for pattern_door, pattern_pane and pattern_window with corresponding weights. The smallest* ds *assign objects to a particular class.*

For a classification experiment, we used thirty-eight unknown graphical objects from the database, previously extracted from some images. In Fig. 3 there are distances *d* (computed based on eq. 5) with weights μ_{P_k} for each object in its ID order. The figure presents overlapping distances for door, glass pane and window frame patterns (see key). The majority of smallest *d*s corresponds with the object number IDs for pattern_door, pattern_pane and pattern_window. As we can see in Fig. 4, based on the *d* values, we found objects ID = 18, 24, 25, 35 (not having been previously classified) belong to the window frame class, objects ID = 3, 6, 32, 33 belong to the glass pane class and objects ID = 4, 9, 31 belong to the door class.

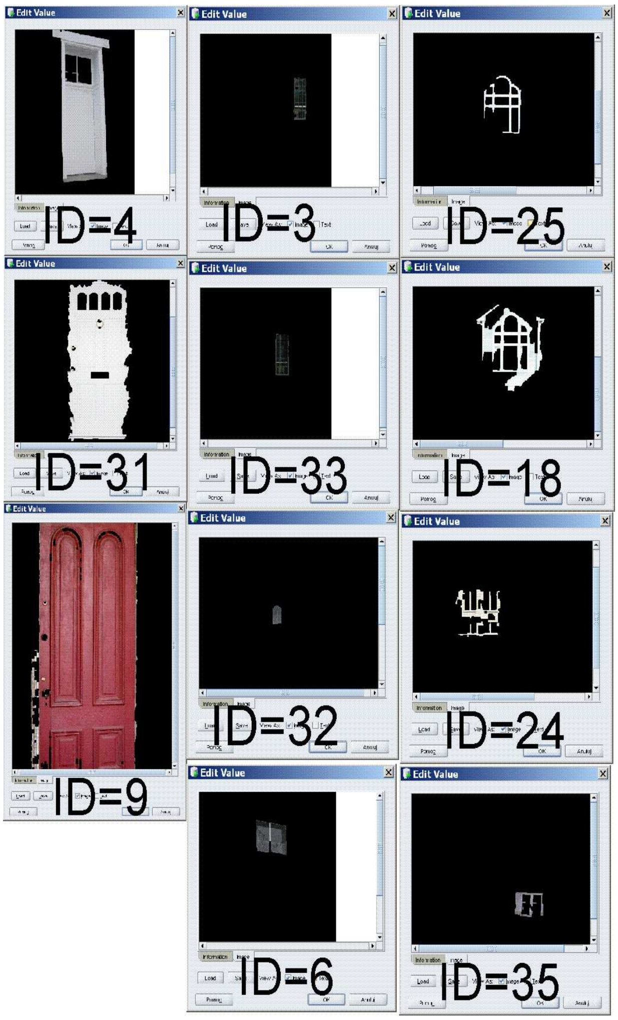

Fig. 4. *Graphical objects found as a result of the classification method and object indexing. Object IDs correspond to object numbers in Fig. 3.*

This fact proves that the classification method, as well as class patterns and correspondent weights, are adequate for our purpose. Fig. 3 and 4 confirm the appropriateness of our decision.

3.4. Spatial Object Location as a Global Feature

Object classification into a particular category is not sufficient 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 [1] 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 [14] 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 for the user to recognize this spatial location visually. 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 in 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 [6]. 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, is presented as a vector F_g for the global feature:

$$F_g = \begin{bmatrix} (x_{c_1}, y_{c_1}), t_{o_1} \\ (x_{c_2}, y_{c_2}), t_{o_2} \\ \vdots \\ (x_{c_N}, y_{c_N}), t_{o_N} \end{bmatrix} \qquad (6)$$

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

Graphical User Interface (GUI) is an intrinsic element of our system. According to a human visual perception theory, during the visual perception and recognition process human eyes fixate successively on the most informative parts of an image [8]. These informative parts, called meaningful regions, possess certain semantic meanings.

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. 5 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.

Next, the user chooses particular graphical elements from subsequent menus and places them on the appropriate location in the chosen outline. For each element the user can change its colour (see Fig. 6). 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.

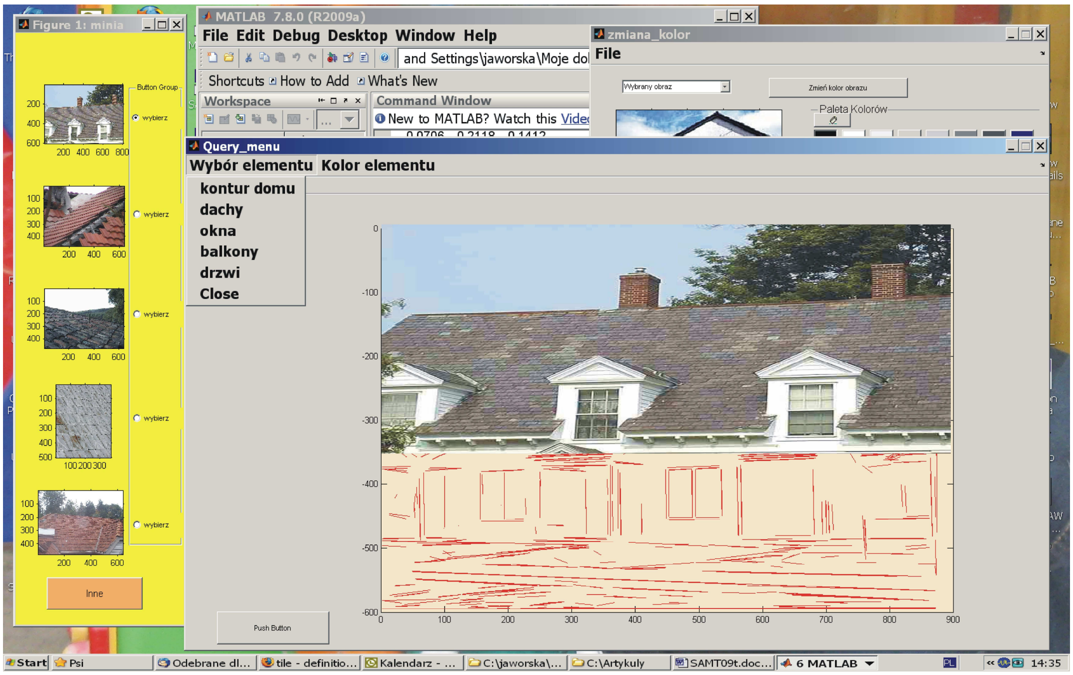

Fig. 5. *The user menu applied by the system to design a query by image. The left window is used to present graphical elements, for example house roofs. It is easy to note that the first roof from the top of the list of miniatures on the left is chosen and located in the house outline.*

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 when 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.

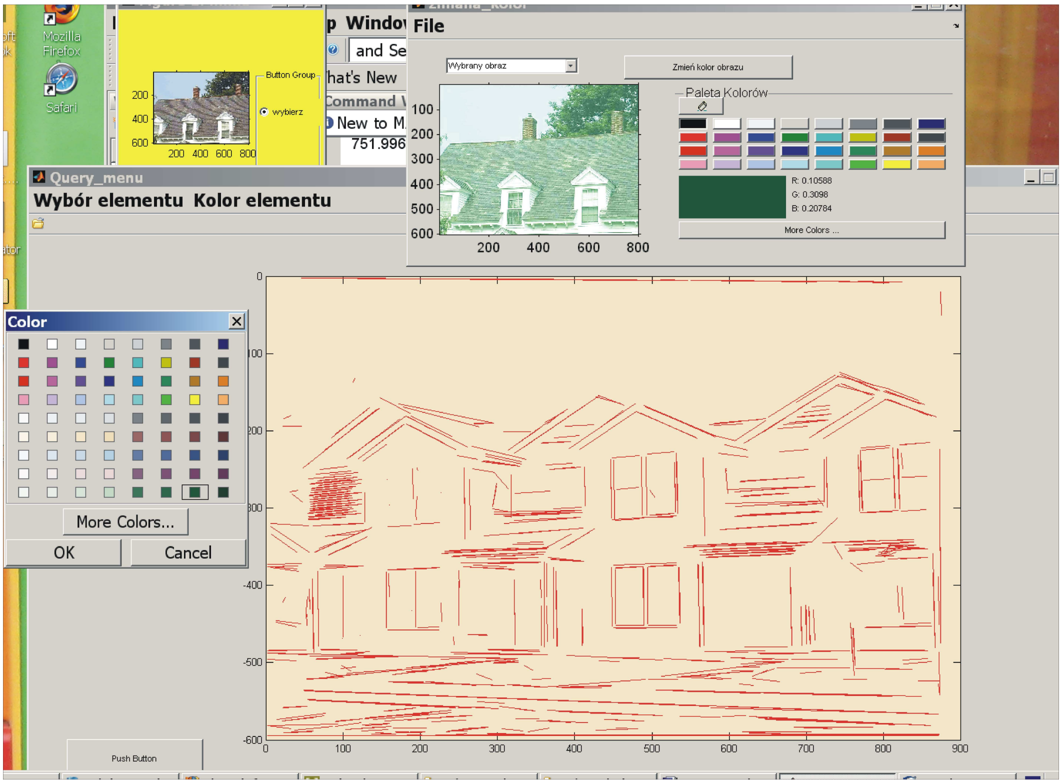

Fig. 6. *Menu tools dedicated to changing element colour. When the user selects a graphical element from the window containing miniatures, he can open the "zmiana_koloru" window in order to change the colour of this element. If the basic colour pallet is found too limited, the user can open the "more colors" window. Having determined the element colour, the user locates the element in the appropriate position in the image outline.*

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 $1 \le k \le N$. 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 label 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. 7, 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 3 and 4, we compare each table cell. The notation used in the tables is as follows: E - object $O(t_1)$ is to the left of object $O(t_2)$, W - object $O(t_1)$ is to the right of object $O(t_2)$, S - object $O(t_1)$ is below object $O(t_2)$, N - object $O(t_1)$ is above object $O(t_2)$.

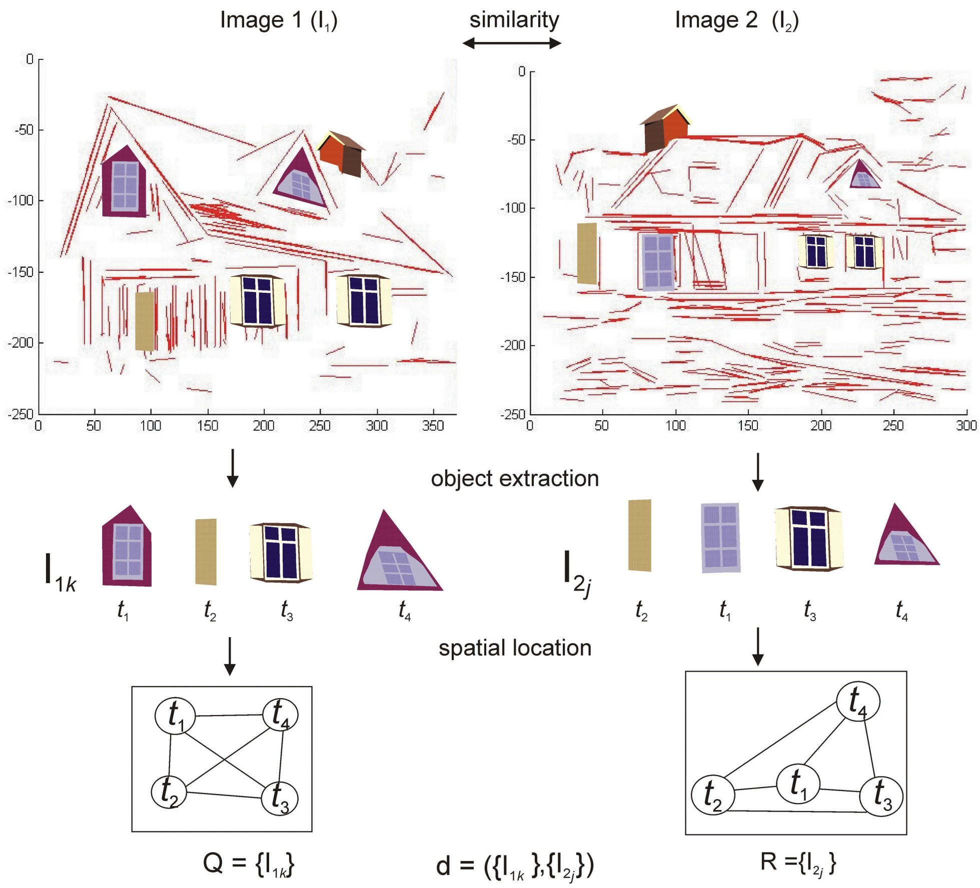

Fig. 7. *Model of the spatial object location described as a global vector* F_g. *For each object* t_{O_k} *we know its feature vector* $O_k(f_i)$.

Table 2 and 3. *Spatial information for query image Q and for relevant image R from Fig. 7.*

Q	*t₁*	*t₂*	*t₃*	*t₄*
t_1	0	S	NE	E
t_2	N	0	E	SE
t_3	SW	W	0	SW
t_4	W	NW	NE	0

R	t_1	t_2	t_3	t_4
t_1	0	W	E	SE
t_2	E	0	E	SE
t_3	W	W	0	SE
t_4	NW	NW	NW	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.

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, thus, 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 7) 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 comparison between fuzzy collections.

6. Conclusions and Further Works

The construction of a CBIR system requires preparing the image processing module for automatic segmentation, as well as the database to store the generated information about images and their segments as its foundation. Based on it, we can build mechanisms for image retrieval. For this purpose, we have proposed indexing and classification methods for graphical objects.

These methods, employed so far in the image retrieval in our system are still rather rough. We are currently 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 different kinds 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.

7. References

[1] Chow T. W., Rahman M. K., Wu S.: *Content-based image retrieval by using tree-structured features and multi-layer self-organized map*. Pattern Analysis and Applications Vol. 9, 2006, pp. 1—20.

[2] Flickner M., Sawhney H., et al.: *Query by Image and Video Content: The QBIC System*, IEEE Computer **28**, No. 9, 1995, pp. 23 –32.

[3] Huang J. et al. *Spatial Color Indexing and Applications*, International Journal of Computer Vision, Vol 35, No. 3, Kluwer Academic Publishers, the Netherlands, 1999, pp. 245—268.

[4] *ISO/MPEG N6828, Overview of MPEG-7*, ver. 10, (ed.) Martinez J. M., Palma de Mallorca, October, 2004.

[5] 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, 2007, pp. 184 -195.

[6] Jaworska T.: *Database as a Crucial Element for CBIR Systems*, In: Proceedings of the 2nd International Symposium on Test Automation and Instrumentation, Vol. 4, World Publishing Corporation, Beijing, China, 2008, pp. 1983-1986.

[7] 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., 2007, pp. 111—131.

[8] Newman W. M., Lamming M. G.: *Interactive System Design*, Addison-Wesley, Harlow, 1996.

[9] Ogle V., Stonebraker M.: *CHABOT: Retrieval from a Relational Database of Images*, *IEEE Computer* **28**, No. 9, 1995, pp. 40 –48.

[10] Pentland A., Picard R., Sclaroff S.: Photobook: *Content-based manipulation of image databases*, International Journal of Computer Vision, Vol. 18, No. 3, 1996, pp. 233—254.

[11] Shi Z., He Q., Shi Zh.: *An index and retrieval framework integrating perceptive features and semantics for multimedia databases*, Multimedia Tools and Applications, Vol. 4, 2009, pp. 207—231.

[12] Swain M., Ballard D.: *Color indexing*, International Journal of Computer Vision, Vol. 7 No. 1, 1991, pp. 11—32.

[13] Swain M., Stricker M.: *The capacity of colour histogram indexing*, In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 704-708.

[14] 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, 2004, pp. 277—301.

[15] 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, 1999, pp. 105- 119.