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

Tatiana JAWORSKA*

MULTI-CRITERIA OBJECT INDEXING AND A GRAPHICAL USER QUERY AS AN ASPECT OF CONTENTS-BASED IMAGE RETRIEVAL SYSTEM

In this article we propose multimedia-oriented indexing used for image matching in the Content-Based Image Retrieval (CBIR) system containing colour house 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 feature vectors for each object and spatial relationships used as a global feature for image retrieval. Secondly, we address the problem of the graphical 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.

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 before, the CBIR system has yet to reach maturity. Information retrieval is very closely connected with image indexing problem, as well as how to effectively put an image query for the CBIR system. We would like to analyse these two aspects of CBIR in this article.

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

1.1. INDEXING AND QUERY BY IMAGE BACKGROUND

Most of the CBIR systems adopt the following two-step approach to the search of image databases [12]: indexing based on a feature vector and searching by a query image. 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, 8, 9,11], others used a colour correlogram [3] or hierarchical semantics and hierarchical cluster indexes [10].

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, which is currently under construction, is intended to be entirely automatic.

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 [7] the images are retrieved based on the above-mentioned attributes separately or using distance functions between features. Tools in this GUI include: polygon outliner, rectangle outliner, line draw, object translation, flood fill, etc.

2. CBIR CONCEPT OVERVIEW

The purpose of this paper is to present an indexing method which retrieves images based on the individual elements of an image, 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 (version 2009a) with the support of the following toolboxes: Image Processing, Fuzzy, Statistics, Wavelet and Database.

2. The Database, storing information about whole images, their segments (here referred to as image objects), segment attributes and object location. The DBMS has been prepared with Oracle Designer 6i and implemented in Oracle 10g.
3. The indexing module responsible for the image indexing procedure, developed in Matlab.
4. The graphical user's interface (GUI), also applied in Matlab.

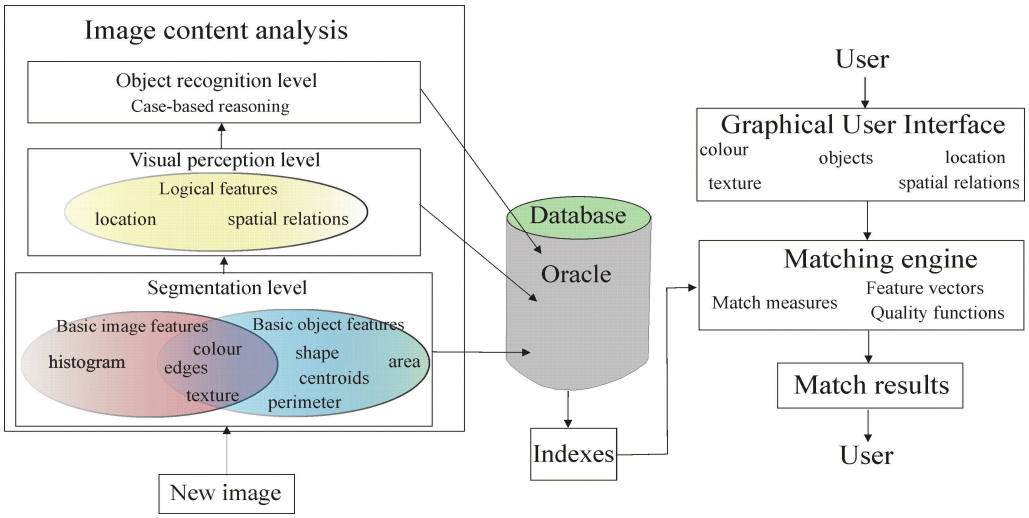

Fig. 1. Block diagram of our content-based image retrieval system.

Our CBIR system is built to support real-estate agencies. In a database of real-estate properties there are images of houses, bungalows and other buildings. To be effective in terms of presentation and choice of houses, the system has to be capable of finding the colour image of a house with some specified architectural elements, for example: windows, roofs, doors, etc. Images are downloaded from the Internet in the JPEG format. 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 [13]. Such an 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 the image [6]. These informative parts, called meaningful regions, possess certain semantic meanings, and, in our system, they correspond to the architectural elements.

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 house of some architectural elements. Moreover, the system presents them some optional houses based on these chosen elements. GUI details are presented in subsection 4.

The next component of the system is the matching engine, which uses indexes based on global and local feature vectors 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 key architectural elements (such as windows, doors, roofs, etc.) must be segmented and extracted from the background at the stage of preprocessing. Colour images are downloaded from the Internet and their preprocessing is unsupervised. An object extraction from the image background must be done in a way permitting 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 objects (referred to as architectural elements of the house) are extracted from each layer. Next, the low-level features are determined for each object, understood as a fragment of the entire image. The segmentation algorithm and object extraction algorithm, as well as texture parameters finding algorithm are presented in detail in an article by Jaworska [4].

3. THE INDEXING SCHEME

3.1. DATA REPRESENTATION FOR OBJECTS

Each selected object is described by some low-level features. These features include: average colour, area, centroid, eccentricity, orientation, texture parameters, moments of

inertia, etc. Average colour is calculated by summing up values of the red, green and blue components for all the pixels belonging to an object, and divided by the number of object pixels:

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

The next feature attributed to objects is texture. Texture parameters are found in the wavelet domain. The algorithm details are also given in [4]. The use of this algorithm results in obtaining two ranges for the horizontal object dimension *h* and two others for the vertical one *v*:

$$T_p = \begin{Bmatrix} \{h\min_{1,2}; h\max_{1,2}\} \\ \{v\min_{1,2}; v\max_{1,2}\} \end{Bmatrix} \quad (2)$$

The other features describing each object include: 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 features 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 features. For an object, we construct a feature vector O containing the above-mentioned features:

$$O = \begin{bmatrix} O(k_{av}) \\ O(T_p) \\ \vdots \\ O(E) \end{bmatrix} = \begin{bmatrix} O(f_1) \\ O(f_2) \\ \vdots \\ O(f_r) \end{bmatrix} \quad (3)$$

3.2. PATTERN LIBRARY

The pattern library contains information about pattern types, shape descriptors, optional object locations and allowable parameter values for an object. We define a model feature vector P_O for each architectural element. We assume weights μ_P

characteristic of a particular type of element which satisfy: $\mu_{P_k}(f_i) \in [0,1]$, where: k - number of pattern. These weights for each pattern component should be assigned in terms of the best distinguishability of patterns.

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

$$d(O,P_k) = \left[\sum_{i=1}^{r} \mu_{P_k}(f_i) \left| O(f_i) - P_k(f_i) \right|^p \right]^{1/p} \tag{4}$$

where: k – pattern number, $1 \le i \le r$, p is the order of the metric. For $p = 1$ and for $p = 2$, it becomes the Manhattan and the Euclidean distance, respectively. The object is assigned to a pattern for which the minimum distance has been obtained and labelled as t_{O_k}.

3.3. THE GLOBAL FEATURE AS A SPATIAL OBJECT LOCATION

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 [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.

In our system spatial object location in an image is used as a global feature. To determine this feature, the mutual position of all objects is checked. These rules are also stored in the pattern library [5]. 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 vector F_g for the global feature:

$$F_g = \begin{bmatrix} (x_{c_1}, y_{c_1}), t_{o_1} \\ (x_{c_2}, y_{c_2}), t_{o_2} \\ \vdots \\ (x_{c_N}, y_{c_N}), t_{o_N} \end{bmatrix}, \tag{5}$$

where $\{x_{c_i}, y_{c_i}\}$ is an object centroid and N – the number of all objects in an image, t_{O_k} – an object label assigned in the process of identification. As you can see in figure 3, we analyse mutual spatial location for particular types of objects.

4. GUI FOR QUERY BY IMAGE

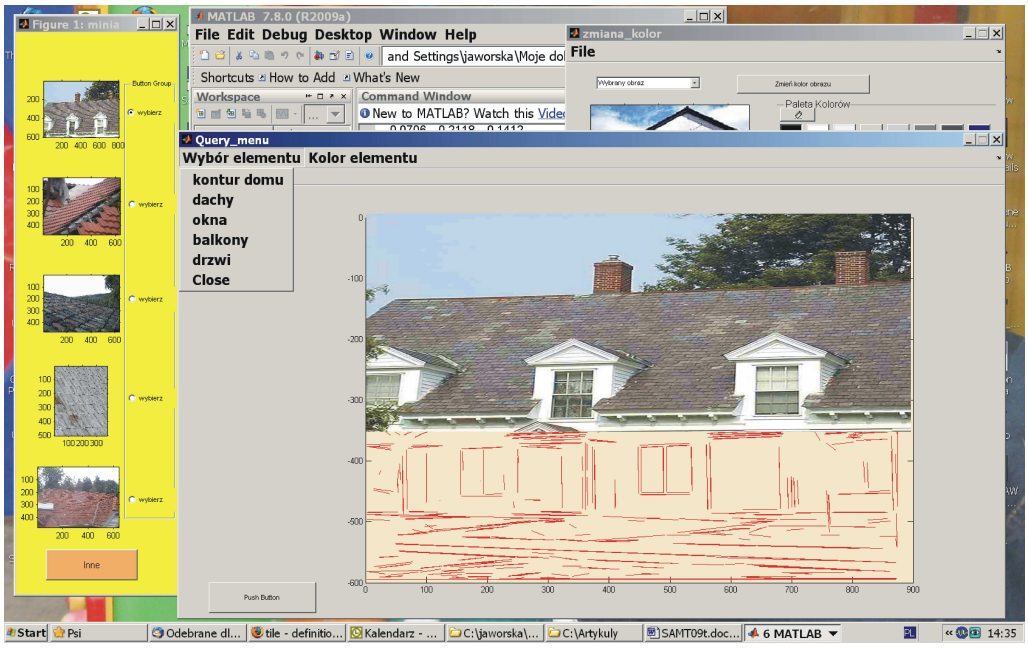

Fig. 2. The user menu used by the system to design a query by image. The left window is used to present architectural elements, for example roofs. For instance, the first roof from the top of the list of miniatures on the left is chosen and located in the house outline.

The Graphical User Interface (GUI) is an immanent element of our system. Drawing on our knowledge 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 of a house. This design is treated as a query by image. Fig. 3 presents the main GUI window entitled “Query_menu”. From the left window the user can choose the house outlines which become visible in an enlarged form in the main window. Next, the user chooses particular architectural elements from subsequent menus and situates them on the appropriate location in the house outline. For each element the user can change its colour. Moreover, there is a window for changing the element texture, if it has one, or adding a texture in a chosen colour for non-textured components.

For more advanced users there are additional options in the query interface which enable them to choose 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. Then, we try to match object $O = \mu_{q_{Oi}}(f_i)$ to objects stored in the DB.

A designed image is sent as a query to DB. 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 $1 \le k \le N$. First, the relevant image $R = \{F_{gR}, O_{R_1}, ..., O_{R_N}\}$ with N objects is 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.

This means that objects are not matched based upon fixed positions in the image. We additionally need a relative location, for example, as you can see in fig. 3, 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 (see tables 1 and 2). For matching images Q and R, whose spatial information is illustrated in tables 1 and 2, 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 $O(t_2)$.

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 1. Spatial information for query image Q from Fig. 3.

	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

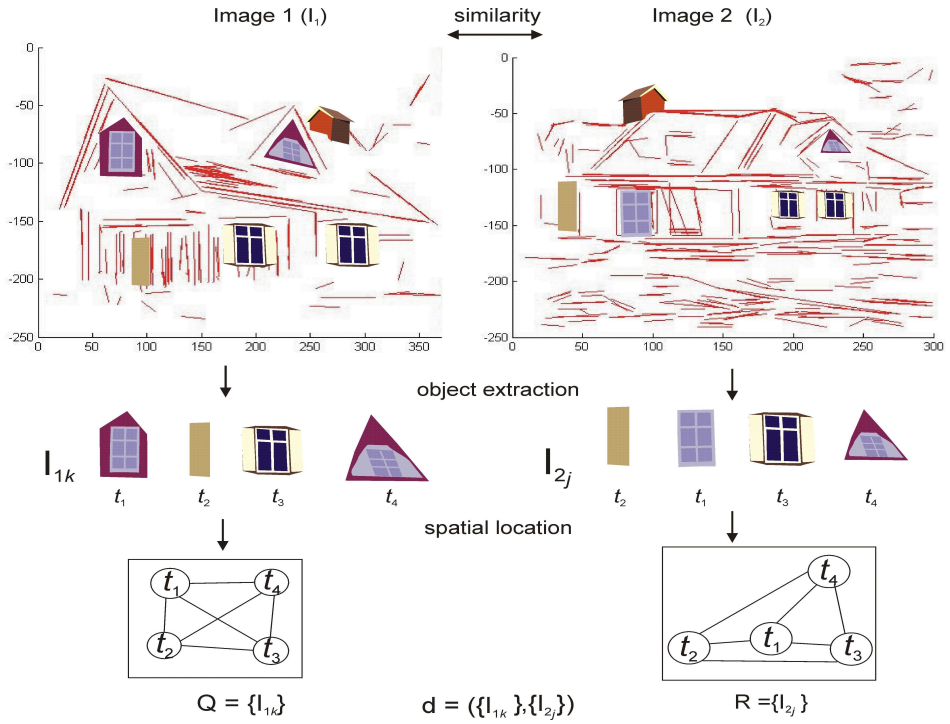

Fig. 3. Model of the spatial object location described as a global vector F_g. For each object t_{Oi} we know its feature vector $O(f_i)$.

Table 2. Spatial information for relevant image R from Fig. 3.

	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

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 $Q = \{O_{q_k}\}$.

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

A perplexing situation is when we attempt to find a picture which is, for instance, a half of another picture. Then the matching is also possible. In this case we are able to match the objects' location table to a fragment of a table for the entire required image.

6. CONTRIBUTIONS AND CONCLUSIONS

In our CBIR system we propose the new GUI specially dedicated to designing a graphical query by the user. So far, the author has not encountered any papers reporting user-designed query by image and, in this respect, the method described above is our original contribution. The construction of the indexing system enables us to retrieve images in an efficient way. Thanks to weights offered to the user, the system can accept user's preferences more flexibly. Having combined two systems: the image processing module and the database, we faced the problem of object identification. For this purpose the object pattern library has been constructed.

To sum up, even though we have experienced a few snags, all our actions lead 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] 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 & Applications, Vol. 9, 2006, 1—20

[2] FLICKNER M., SAWHNEY H., et al., *Query by Image and Video Content: The QBIC System*, IEEE Computer **28**, No. 9, August 1995, 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, 245-268.

[4] JAWORSKA T., *Object extraction as a basic process for content-based image retrieval (CBIR) system*, Opto-Electronics Review, Asso. of Polish Electrical Engineers (SEP), Vol.15, No. 4, Warsaw, 2007, 184-195.

[5] 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, K. Atanassov et al. (eds.), EXIT, Warsaw, 2008, 79-96.

[6] NEWMAN W. M., LAMMING M. G., *Interactive System Design*, Addison-Wesley, Harlow, 1996.

[7] NIBLACK W., FLICKNER M. et al. *The QBIC Project: Querying Images by Content Using Colour, Texture and Shape*, SPIE **1908**, 1993, 173-187.

[8] OGLE V., STONEBRAKER M., *CHABOT: Retrieval from a Relational Database of Images*, IEEE Computer **28**, No. 9, August, 1995, 40-48.

[9] PENTLAND A., PICARD R. and SCLAROFF S., *Photobook: Content-based manipulation of image databases*, International Journal of Computer Vision, Vol. 18 No. 3, 1996, 233-254.

[10] SHI Z., HE Q., SHI Zh., *An index and retrieval framework integrating perceptive features and semantics for multimedia databases*, Multimedia Tools Application, Vol. 4, 2009, 207—231.

[11] SWAIN M., BALLARD D., *Color indexing*, International Journal of Computer Vision, Vol. 7 No. 1, 1991, 11-32.

[12] SWAIN M., STRICKER M., *The capacity of colour histogram indexing*, In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1994, 704-708.

[13] 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, 277—301.