A Search-Engine Concept Based on Multi-feature Vectors and Spatial Relationship

Tatiana Jaworska

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

Abstract. At present a great deal of research is being done in different aspects of Content-Based Image Retrieval System (CBIR). Unfortunately, these aspects are mostly analysed separately. We propose how to put together vectors of features for segmented objects and a spatial relationship of the objects. To achieve this goal we have constructed a search engine taking into account multi-set data mining and object spatial relationship. Additionally, we have constructed a graphical user interface (GUI) to enable the user to build a query by image. The efficiency of our system will be evaluated in the near future. In this paper we present the search engine for our CBIR.

Keywords: CBIR, spatial relationship, search engine, GUI, feature vector, multi-set.

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 have been made before, a major problem in this area has been 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 considered separately, and image retrieval based on high-level semantic concepts that perceive an image as a complex whole. This problem becomes especially challenging when image databases are exceptionally large.

Images and graphical data are complex in terms of visual and semantic contents. Depending on the application, images are modelled and indexed using their

- visual properties (or a set of relevant visual features),
- semantic properties,
- spatial or temporal relationships of graphical objects.

Over the last decade a number of concepts of the CBIR [1], [2], [3], [4], have been used. In Wikipedia we can also find a list of CBIR engines, used either for commercial or academic research purposes [5].

H. Christiansen et al. (Eds.): FQAS 2011, LNAI 7022, pp. 137–148, 2011.

Proposals can be found for the relational [6], object-oriented [7], [8] and object-relational database models [9]. Nevertheless, programmers have limited tools when they need to develop graphical applications dealing with imperfect pictorial data. Within the scope of semantic properties, as well as graphical object properties the first successful attempt was made by Candan and Li [10] who constructed the Semantic and Cognition-based Image Retrieval (SEMCOG) query processor to search for images by predicting their semantic and spatial imperfection. This new approach has been very important because earlier, and even present-day, queries to the database are put as query-by-example images.

Hence, in order to give the user the opportunity to compose their own image, consisting of separate graphical objects as a query, we have had to create our own system. An image created in GUI has its own unique object location in the image space. Thus, many researchers Chang [11,12], Chang and Wu, [13, 14], Zhou at al., [15] highlighted the importance of perceiving spatial relationships existing among the components of an image for efficient representation and retrieval of images in the CBIR.

We have dealt successfully with numerous problems involved in the CBIR system, with one final issue that still requires our attention. Ultimately, we have managed to form a new paradigm in comparing images with the search engine.

In this paper we present a concept of a search engine which takes into account object feature vectors, together with different spatial location of segmented objects in the image. In order to improve the comparison of two images, we need to label these objects in a semantic way.

1.1 CBIR Concept Overview

In general, our system consists of four main blocks (fig. 1):

1. The image preprocessing block (responsible for image segmentation), applied in Matlab, cf. [16];
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, cf. [17];
3. The search engine responsible for the searching procedure and retrieval process based on the feature vector for objects and spatial relationship of these objects in the image, applied in Matlab;
4. The graphical user's interface (GUI), also applied in Matlab.

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, layouts 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. 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. More advanced systems enable users to choose as a query not only whole images but also individual objects. The user can also draw some patterns, consisting of simple shapes, colours or textures [18]. In the SEMCOG query processor [10], the user could organize an image as a spatial composition of five semantic groups of objects, such as: car, woman, man, house and bicycle. Additionally, the user could choose the colour, size and shape of a graphical object. In order to retrieve a matched image, the system integrated an image query statement with non-image operation statement.

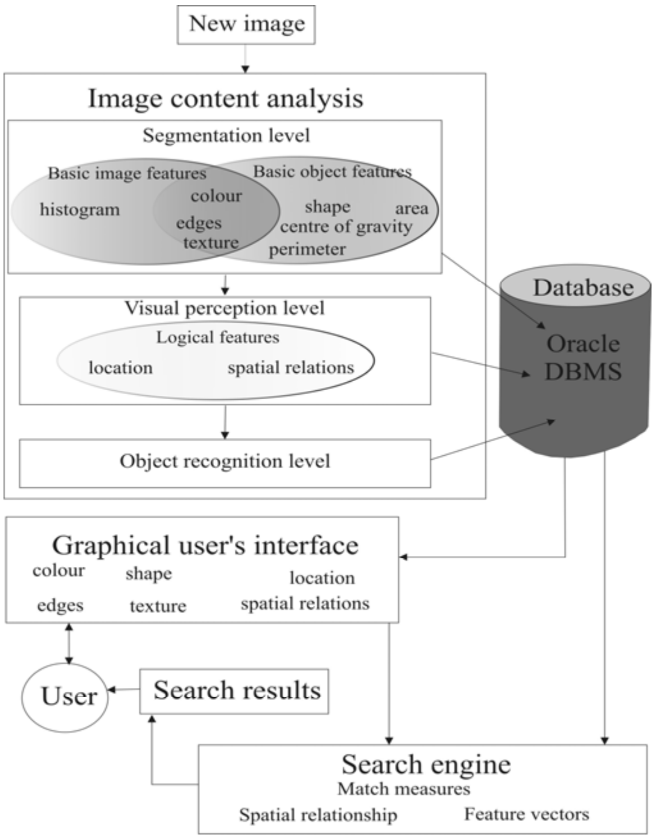

Fig. 1. Block diagram of our content-based image retrieval system

There have been several attempts made by the research community to disperse the demands in the design of efficient, invariant, flexible and intelligent image archival and retrieval systems based on the perception of spatial relationships. Chang [19] proposed a symbolic indexing approach, called the nine directional lower triangular (9DLT) matrix to encode symbolic images. Using the concept of 9DLT matrix, Chang and Wu [20] proposed an exact match of the retrieval scheme, based upon principal component analysis (PCA). Unfortunately, it turned out that the first principal component vectors (PCVs) associated with the image and the same image rotated are not the same. Eventually, an invariant scheme for retrieval of symbolic images based upon the PCA was prepared by Guru and Punitha [21].

2 Graphical Data Representation

In our system, Internet images are downloaded. Firstly, the new image is segmented, creating a collection of objects. Each object, selected according to the algorithm presented in detail in [16], is described by some low-level features. The features 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 $\{bb_1(x,y), ..., bb_s(x,y)\}$ (s – number of vertices), major axis length m_{long}, minor axis length m_{short}, solidity s and Euler number E and Zernike moments $Z_{00},\ldots,Z_{33}$. All features, as well as extracted images of graphical objects, are stored in the DB. Let F be a set of features where:

$$F = \{k_{av}, T_p, A, A_c, \ldots, E\} \tag{1}$$

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:

$$F_O = \begin{bmatrix} O(k_{av}) \\ O(T_p) \\ O(A) \\ \vdots \\ O(Z_{33}) \end{bmatrix} = \begin{bmatrix} O(f_1) \\ O(f_2) \\ O(f_3) \\ \vdots \\ O(f_r) \end{bmatrix}. \tag{2}$$

The average colour is an average of each red, green and blue component which is summed up for all the pixels belonging to an object, and divided by the number of object pixels $k_{av}=\{r_{av}, g_{av}, b_{av}\}$. 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 [16]. The use of this algorithm results in obtaining two ranges for the horizontal object dimension h and two others for the vertical one v:

$$T_p = \begin{Bmatrix} h_{\min_{1,2}}; h_{\max_{1,2}} \\ v_{\min_{1,2}}; v_{\max_{1,2}} \end{Bmatrix}. \tag{3}$$

Additional features of the low level for objects are shape descriptors. They are also included in the above mentioned feature vector. We apply the two most important shape descriptors such as moments of inertia:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y), \qquad p,q = 0,1,2 \tag{4}$$

and Zernike moments [22]. Zernike moments are a set of complex polynomials $\{V_{pq}(x,y)\}$ which form a complete orthogonal set over the unit disk of $x^2 + y^2 \le 1$. Hence, the definition of 2D Zernike moments with p^{th} order with repetition q for intensity function $f(x,y)$ of the image is described as:

$$Z_{pq} = \frac{p+1}{\pi} \iint_{x^2+y^2 \le 1} V_{pq}^{*}(x,y) f(x,y)\, dxdy \tag{5}$$

where: $$V_{pq}^{*}(x,y) = V_{p,-q}(x,y)\ . \tag{6}$$

For our purpose, the first 10 Zernike moments are enough, it means we calculate moments from Z_{00} to Z_{33}. Fig. 2 presents a module applied to find similarities between separate segmented elements.

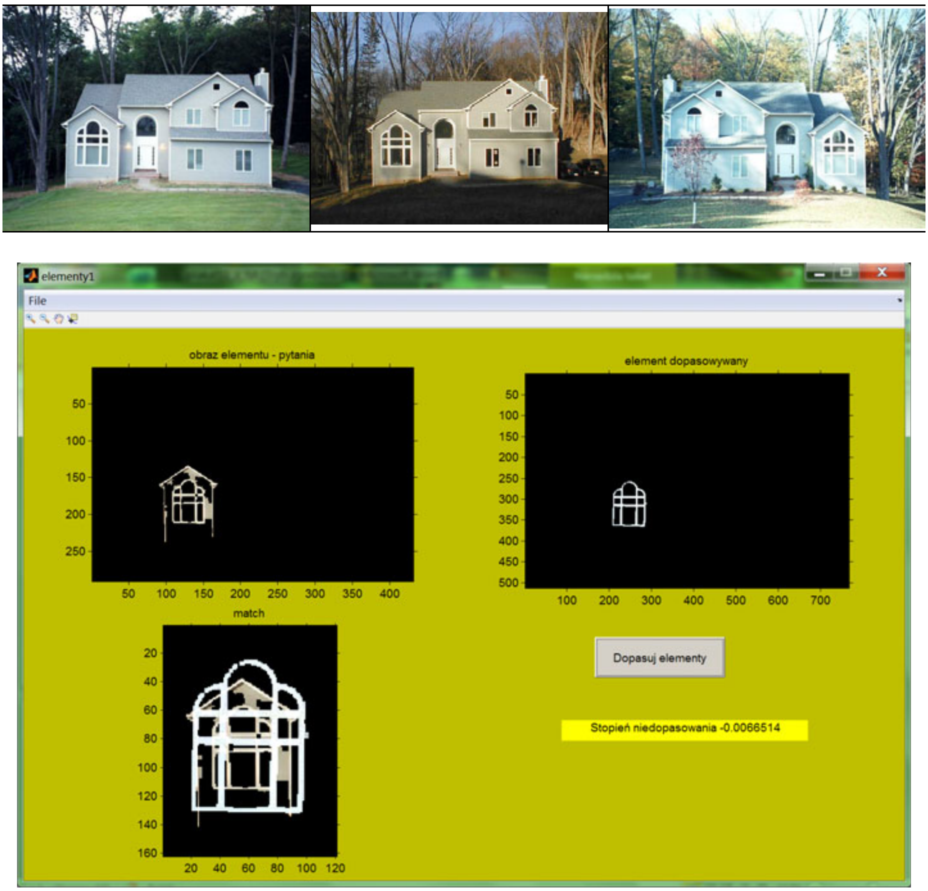

Fig. 2. An example of two matched objects based on the minimal Euclidean distance for the first 10 Zernike moments. These two elements were matched from the objects segmented from the images presented above.

Characteristic features of Zernike moments are:

1. The above-defined Zernike moments are only invariant to rotation.
2. The translation invariance is achieved by the location of the original image centroid in the centre of the coordinates.
3. The scale invariance is obtained by normalizing Z_{00} by the total number of image pixels.

3 Spatial Relationship of Graphical Objects

The feature vector F_o (cf. (2)) is further used for object classification. Therefore, we have to classify objects first in order to assign them to a particular class and second in order to compare objects coming from the same class [23].

In our system 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 [23]. Thirdly, object location reduces the differences between high-level semantic concepts perceived by humans and low-level features interpreted by computers.

For the comparison of the spatial features of two images an image I_i is interpreted as a set of n objects composing it:

$$I_i = \{o_{i1}, o_{i2}, \ldots, o_{in}\} \tag{7}$$

Each object o_{ij} is characterized by a unique identifier and a set of features discussed earlier. This set of features includes a centroid $C_{ij} = (x_{ij}, y_{ij})$ and a label L_{ij} indicating the class of an object o_{ij} (such as window, door, etc.), identified in the process described in [23]. For convenience, we number the classes of the objects and thus L_k's are just numbers.

Formally, let I be an image consisting of n objects and k be a number of different classes of these objects, $k \leq N$, because usually there are some objects of the same type in the image, for example, there can be four windows in a house.

Let us assume that there are, in total, M classes of the objects recognized in the database, denoted as labels $L_1, L_2, \ldots, L_M$. Then, by the signature of an image I_i (**7**) we mean the following vector:

$$\text{Signature}(I_i) = [\text{nobc}_{i1}, \text{nobc}_{i2}, \ldots, \text{nobc}_{iM}] \tag{8}$$

where: nobc_{ik} denotes the number of objects of class L_k present in the representation of an image I_i, i.e. such objects o_{ij}.

Additionally, for an image I_i we consider a representation of spatial relationships of the image objects. The object's o_{ij} mutual spatial relationship is calculated based on the algorithm below. Now, we consider one image; let C_p and C_q be two object centroids with $L_p < L_q$, located at the maximum distance from each other in the image, i.e.,

$$\text{dist}\,(C_p, C_q) = \max\,\{\text{dist}\,(C_i, C_j)\ \forall i,j \in \{1,2,\ldots,k\} \text{ and } L_i \neq L_j\} \tag{9}$$

where: dist(•) is the Euclidean distance between two centroids (see fig. 3). The line joining the most distant centroids is the line of reference and its direction from centroid C_p to C_q is the direction of reference for computed angles θ_{ij} between other centroids. This way of computing angles makes the method invariant to image rotation.

Hence, we received triples (L_i, L_j, θ_{ij}) where the mutual location of two objects in the image is described in relation to the line of reference (see fig. 3 bottom). Thus, there are $T=m(m-1)/2$ numbers of triples, generated to logically represent the image consisting of m objects. Let S be a set of all triples, then we apply the concept of principal component analysis (PCA) proposed by Chang and Wu [20] and later modified by Guru and Punitha [21] to determine the first principal component vectors (PCVs).

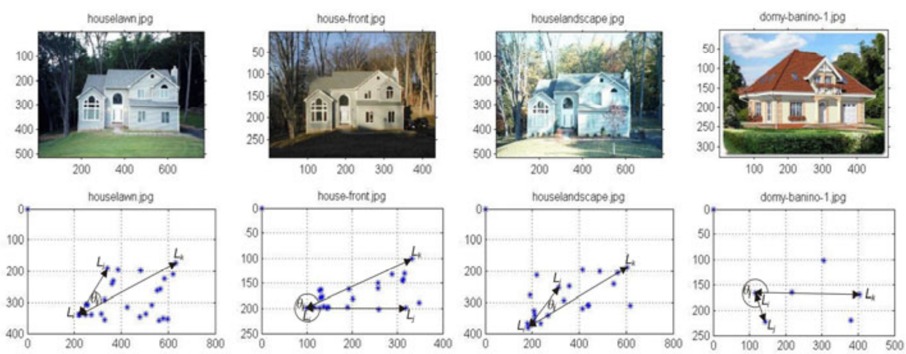

Fig. 3. Determination of angle relative to the reference direction for the construction of matrix S

First, we have to suppose that S is a set of observations for three variables. We construct a matrix of observations $X_{3\times N}$ where each triple is one observation. Next, we count the mean value u of each variable, and we calculate the deviations from the mean to generate matrix $\mathbf{B}=\mathbf{X}-\mathbf{u1}$, where $\mathbf{1}$ - vector of all 1s. In the next step, we compute the covariance matrix $\mathbf{C}_{3\times 3}$ from the outer product of matrix $\mathbf{B}$ by itself as:

$$\mathbf{C} = \mathbb{E}\,[\mathbf{B}\otimes\mathbf{B}] = \mathbb{E}\,[\mathbf{B}\,\mathbf{B}^*] = 1/N\,[\mathbf{B}\,\mathbf{B}^*]. \tag{10}$$

where: $\mathbb{E}$ is the expected value operator, ⊗ is the outer product operator, and * is the conjugate transpose operator. Eventually, we find eigenvectors, which diagonalises the covariance matrix **C**:

$$V^{-1}\,C\,V=D \tag{11}$$

where: D is the diagonal matrix of the eigenvalues of C.

Using the Matlab procedure `V = princomp(X),` we receive three component vectors (PCVs). For further analysis we use the first of them, which is the "spatial component" of the representation of an image I_i, and is denoted PCV_i.

For example, we use centroid coordinates from our CBIR to find angle θ_{ij} (see fig. 3 bottom). Thus, we construct set S of our observations, where N is combinations of the centroid numbers. For example, $N_{I_1} = C_2^{26} = 325$ and $N_{I_2} = C_2^{21} = 210$, respectively. The obtained results are shown in table 1.

Table 1. Representative principal component vectors for the images shown in fig. 3

Image name	First component	Second component	Third component
House-front	-0,001786	-0,003713	0,999992
Domy-banino-1	0,000206	0,003988	0,999992
Houselawn I_1	0,000388	0,001869	0,999998
Houselandscape I_2	0,004109	0,001557	0,999990

4 Construction of Search Engine

4.1 Graphical User Interface

Graphical User Interface (GUI) is a crucial element of our system as the area of human-computer interaction [24]. Hence, 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”. In the left window the user can choose the image outlines which become visible in an enlarged form in the main window.

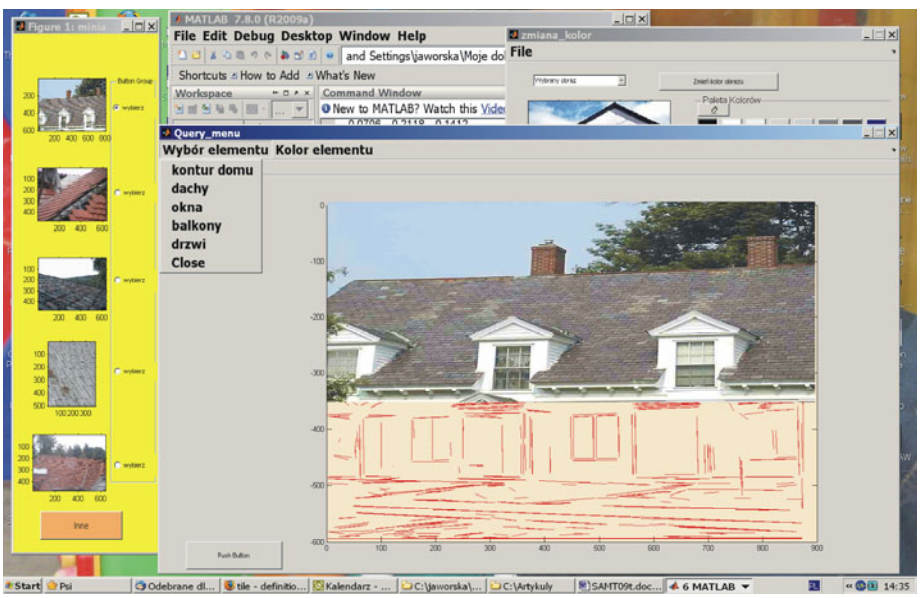

Fig. 4. The user menu applied by the system to design a query by image. The left window is used to present graphical elements, for example house roofs. It is easy to notice that the first roof at the top of the list of miniatures on the left is chosen and located in the house outline.

Next, the user chooses particular graphical elements from subsequent menus and places them on the appropriate location in the chosen outline. These elements can be scaled in a limited range. In most query-by-example systems, the features for retrieval and their importance are estimated by the system. Even in systems where such information can be provided by the user, users cannot always communicate unambiguously what they are looking for. In our system, these constraints are overcome by the user's selection of specific features (for example, the colour and texture of an object) from numerous menus. After the designing process, the image is sent as a query to the DB; it means that we have feature vectors F_{qi} (where i =1,..., N) for all objects used to form either query image I_q and PCV$_q$.

4.2 Similarities

So far, we have described how images are represented in our system. Now, we will describe how the similarity between two images is determined and used to answer a query. Let a query be an image I_q, such as $I_q = \{o_{q1}, o_{q2},\ldots, o_{qn}\}$ (cf. (**7**)). An image in the database will be denoted as I_b, $I_b = \{o_{b1}, o_{b2},\ldots, o_{bm}\}$. In order to answer the query, represented by I_q, we compare it with each image I_b in the database in the following way.

First of all, we determine a first similarity measure sim$_{\text{sgn}}$ between I_q and I_b computing the Hamming distance $d_H(x,y) \in \mathbb{F}_{10}^{(M)}$ between the vectors of their signatures (**8**), i.e.:

$$\text{sim}_{\text{sgn}}(I_q, I_b) = d_H(\text{nobc}_q, \text{nobc}_b) \tag{12}$$

If the similarity (**12**) is smaller than a threshold (a parameter of the query), then image I_b is rejected, i.e., not considered further in the process of answering query I_q. Otherwise, we proceed to the next step and we find the spatial similarity sim$_{\text{PCV}}$ of images I_q and I_b computing the Euclidean distance between their PCVs as:

$$\text{sim}_{\text{PCV}}(I_q, I_b) = 1 - \sqrt{\sum_{i=1}^{3} (PCV_{bi} - PCV_{qi})^2} \tag{13}$$

If the similarity (**13**) is smaller than the threshold (a parameter of the query), then image I_b is rejected, i.e., not considered further in the process of answering query I_q. Otherwise, we proceed to the final step, namely, we compare the similarity of the objects representing both images I_q and I_b. For each object o_{qi} present in the representation of the query I_q, we find the most similar object o_{bj} of the same class, i.e., $L_{qi} = L_{bj}$. If there is no object o_{bj} of the class L_{qi}, then sim$_{\text{ob}}$ (o_{qi}, o_b) is equal to 0. Otherwise, similarity sim$_{\text{ob}}$ (o_{qi}, o_b) between objects of the same class is computed as follows:

$$\text{sim}_{\text{ob}}(o_{qi}, o_{bj}) = 1 - \sqrt{\sum_{l} (Fo_{qil} - Fo_{bjl})^2} \tag{14}$$

where l indexes the set of features F_O used to represent an object, as described in (2). When we find highly similar objects (for instance, sim$_{\text{ob}}$ > 0.9), we eliminate these two objects from the following process of comparison [25]. The process is realized according to the algorithm presented below:

```
Algorithm: Pair matching algorithm with elimination
k=0;
i=1;
j=1;
for j=j:Lqi %number of objects in a particular class
 for i=i:Lbj %number of objects in a particular class
  if sim(i,j)>.9
       match(i,j)=sim(i,j);
       row(i)=i;
       col(j)=j;
        j=j+1;
        i=i+1;
   end;
    end;
end;
while k==0
   [k,R]=min(row);
   [k,C]=min(col);
  match(R,C)=sim(R,C);
  row(R)=R;
  col(C)=C;
end;
```

Thus, we obtain the vector of similarities between the query I_q and an image I_b.

$$\mathrm{sim}(I_q, I_b) = \begin{bmatrix} \mathrm{sim}_{\mathrm{ob}}(o_{q1}, o_{b1}) \\ \vdots \\ \mathrm{sim}_{\mathrm{ob}}(o_{qn}, o_{bn}) \end{bmatrix} \tag{15}$$

where n is the number of objects present in the representation of I_q.

In order to compare images I_b with the query I_q, we compute the sum of $\mathrm{sim}_{\mathrm{ob}}(o_{qi}, o_{bi})$ and then use the natural order of the numbers. Thus, the image I_b is listed as the first in the answer to the query I_q, for which the sum of similarities is the highest.

5 Conclusion

The construction of a CBIR system requires combining different functional systems, linked together and cooperating with each other. For this purpose, object classification and identification procedures have been established and the GUI prototype has been constructed.

We have prepared a model of image similarity as a three-step procedure. This is, of course, a preliminary model of a three-step procedure to answer a query. There are many other possible ways to compute the similarity between the images, e.g. using different metrices. Intensive computational experiments are under way in order to come up with some conclusions as to the choice of the parameters of the model, including the choice of the above-mentioned metrices. However, the preliminary results we have obtained so far using the simplest configuration are quite promising.

References

1. Deb, S. (ed.): Multimedia Systems and Content-Based Image Retrieval, ch. VII and XI. IDEA Group Publishing, Melbourne (2004)
2. Flickner, M., Sawhney, H., et al.: Query by Image and Video Content: The QBIC System. IEEE Computer 28(9), 23–32 (1995)
3. Niblack, W., Flickner, M., et al.: The QBIC Project: Querying Images by Content Using Colour, Texture and Shape. In: SPIE, vol. 1908, pp. 173–187 (1993)
4. Ogle, V., Stonebraker, M.: CHABOT: Retrieval from a Relational Database of Images. IEEE Computer 28(9), 40–48 (1995)
5. `http://en.wikipedia.org/wiki/List_of_CBIR_engines`
6. Pons, O., Vila, M.A., Kacprzyk, J.: Knowledge management in fuzzy databases. Studies in Fuzziness and Soft Computing, vol. 39. Physica –Verlag, Heidelberg (2000)
7. Lee, J., Kuo, J.-Y., Xue, N.-L.: A note on current approaches to extent fuzzy logic to object oriented modeling. International Journal of Intelligent Systems 16, 807–820 (2001)
8. Berzal, F., Cubero, J.C., Kacprzyk, J., Marin, N., Vila, M.A., Zadrożny, S.: A General Framework for Computing with Words in Object-Oriented Programming. In: Bouchon-Meunier, B. (ed.) International Journal of Uncertainty. Fuzziness and Knowledge-Based Systems., vol. 15(Supplement), pp. 111–131. World Scientific Publishing Company, Singapore (2007)
9. Cubero, J.C., Marin, N., Medina, J.M., Pons, O., Vila, M.A.: Fuzzy Object Management in an Object-Relational Framework. In: Proceedings of the 10th International Conference IPMU, Perugia, Italy, pp. 1775–1782 (2004)
10. Candan, K.S., Li, W.-S.: On Similarity Measures for Multimedia Database Applications. Knowledge and Information Systems 51(3), 30–51 (2001)
11. Chang, S.K., Shi, Q.Y., Yan, C.W.: Iconic indexing by 2D strings. IEEE Trans. Pattern Anal. Machine Intell. 9(5), 413–418 (1987)
12. Chang, S.K., Jungert, E., Li, Y.: Representation and retrieval of symbolic pictures using generalized 2D string. In: SPIE Proc. on Visual Comm. and Image Process. Philadelphia, pp. 1360–1372 (1989)
13. Chang, C.C., Wu, T.C.: Retrieving the most similar symbolic pictures from pictorial databases. Informat. Process. Manage. 28(5), 581–588 (1992)
14. Wu, T.C., Chang, C.C.: Application of geometric hashing to iconic database retrieval. Pattern Recognition Letters 15, 871–876 (1994)
15. Zhou, X.M., Ang, C.H., Ling, T.W.: Image retrieval based on object's orientation spatial relationship. Pattern Recognition Letters 22, 469–477 (2001)
16. Jaworska, T.: Object extraction as a basic process for content-based image retrieval (CBIR) system. In: Opto-Electronics Review, Association of Polish Electrical Engineers (SEP), Warsaw, vol. 15(4), pp. 184–195 (2007)
17. Jaworska, T.: Database as a Crucial Element for CBIR Systems. In: Proceedings of the 2nd International Symposium on Test Automation and Instrumentation, vol. 4, pp. 1983–986. World Publishing Corporation, Beijing (2008)
18. Smith, J.R., Chang, S.-F.: Integrated spatial and feature image query. Multimedia Systems 7, 129–140 (1999)

19. Chang, C.C.: Spatial match retrieval of symbolic pictures. J. Informat. Sci. Eng. 7, 405–422 (1991)
20. Chang, C.C., Wu, T.C.: An exact match retrieval scheme based upon principal component analysis. Pattern Recognition Letters 16, 465–470 (1995)
21. Guru, D.S., Punitha, P.: An invariant scheme for exact match retrieval of symbolic images based upon principal component analysis. Pattern Recognition Letters 25, 73–86 (2004)
22. Teague, M.R.: Image analysis via the general theory of moments. In: JOSA, 8th edn., vol. 70, pp. 920–930 (1980)
23. Jaworska, T.: Graphical Object Classification and Query by Image as an Aspect of Content-Based Image Retrieval System. In: Owsiński, J. (ed.) Studies and Materials of Polish Operational and Systems Research Society, Bydgoszcz, Poland, vol. 32, pp. 269–282 (2010)
24. Newman, W.M., Lamming, M.G.: Interactive System Design. Addison-Wesley, Harlow (1996)
25. Mucha, M., Sankowski, P.: Maximum Matchings via Gaussian Elimination. In: Proceedings of the 45th Annual Symposium on Foundations of Computer Science (FOCS 2004), pp. 248–255 (2004)