MERGING IMAGE FEATURES BY SELF ORGANIZING MAPS IN COATS OF ARMS RETRIEVAL

Christian Breiteneder
University of Vienna, Institute for Applied Computer Science and Information Systems
Liebiggasse 4/3-4, A-1010 Vienna, Austria
eMail: Breiteneder@ifs.univie.ac.at

Dieter Merkl
Vienna University of Technology, Institute for Information and Software Engineering
Resselgasse 3 / 188, A-1040 Vienna, Austria
eMail: merkl@ifs.tuwien.ac.at

Horst Eidenberger
Ministry of Science and Transport, Austrian Libraries Network
Garnisongasse 7/21, A-1090 Vienna, Austria
eMail: hme@bibvb.ac.at

1. Introduction

The development of an image retrieval system for a new application domain requires the selection, adoption and integration of existing solutions and the development of new ones. The designer is confronted with issues as what image features are best suited, how they should be extracted, compared and combined, and what techniques should be employed for proper searching.

In general, images - even synthetic ones - contain a huge number of different features: many general purpose feature extraction functions exist for color, texture and shape [4][6][12] and content-based image retrieval (CBIR) systems like QBIC [5] or Virage [1] use them simultaneously to enhance the performance. Usually, an independent query for each feature is initiated and the results are composed by the linear combination of their weighted distance values. This process is usually called *merging* [16]. Merging causes two major problems:

1. Some feature classes are not linearly-related and therefore merging would not be a suitable combination algorithm. They propose a multi-layer neural network for this purpose.

2. Usually the weights have to be provided by the user. In some systems the weights are fixed to certain values but most systems ask the user for his or her preference. The authors of [15] argue, that the "specification of weights imposes a big burdon on the user, as it requires the user to have a comprehensive knowledge of the low level feature representations used in the retrieval system, which is normally not the case."

When using multiple features and distance functions for retrieval the results of all partial queries have to be joined. This is often done through weighted merging of the distance values. The authors introduce a method how the weights, that are used in this process can be set automatically.

To solve the latter problem we implemented a model for the automatic estimation of weights by self organizing maps (SOM). Up to now, SOMs - although a very popular tool in many research areas - have hardly been used in image retrieval applications, as the authors of [14] have noticed. They use tree structured SOMs to index and search image databases. In [6] the creation of an Iconic Index through self organizing maps is suggested. We think that SOMs can be helpful in many

areas of content-based image retrieval and plan to use them for the automatic generation of query models (see section 2).

In many publications related to content-based image retrieval the term "feature" is used in different ways: First, to describe objects in an image. In this paper we we will use the term *object* (*sub-image*) for this purpose. Second, to denote a function extracting properties of an image. Here we will use *feature extraction function*. Third, to describe a common property of an image or sub-image. We will use *feature* in this sense. In query processing we use *search image* for the example image for which the user wants to obtain similar images from the database. An image which is compared to the search image is called a *candidate image* in our terminology.

The paper is organized in 6 sections: In section 2 query models (introduced in [2]) are explained, section 3 discusses how merging is performed and how the weighting algorithm works. In section 4 the test environment and the test results are presented. Section 5 gives a short outlook on future work. Finally, section 6 summarizes and rounds up the paper in a short conclusion.

2. Query models

We define similarity in our CBIR system by so-called *query models*. A query model is a list of tuples of the form: feature extraction function, distance function, threshold and weight. The threshold is the maximum allowed distance between an image in the database and the query image. The size of the result set is determined by the thresholds of all elements of a query model and not - as common in other retrival approches (e.g. [5]) - by an absolute number. In other words, every element in a query model eliminates some images that cannot be part of the result set any more (e.g., due to minimal distance values for other features). Therefore the threshold is an important part of the similarity definition. Table 1 gives an example of a query model. In the first step the color histogram of every image is compared to the color histogram of the search image. For images with a distance value lower than 0.5 it is determined whether they are symmetrical to their x-axis. Images satisfying also the second condition will be returned as the query result.

No.	Feature extraction function	Distance function	Max. threshold value
1	Color histogram	Euclidean distance	0.5
2	X-axis symmetry	Given? yes / no	0

The idea behind query models is the filtering of an image set through a set of consecutive sieves (Figure 1). Each element of the query model reduces the initial image set until only those images are left that are most similar to the search image.

One of the advantages of employing query models is the possibility to optimize the performance by using those features first that have a faster to compute feature function and eliminate a larger group of images. For example, for our coats of arms database we usually apply a simple feature counting the number of significant colors together with a color histogram. This fast to compute feature reduces the image set at least to half of its original size and improves the overall performance considerably.

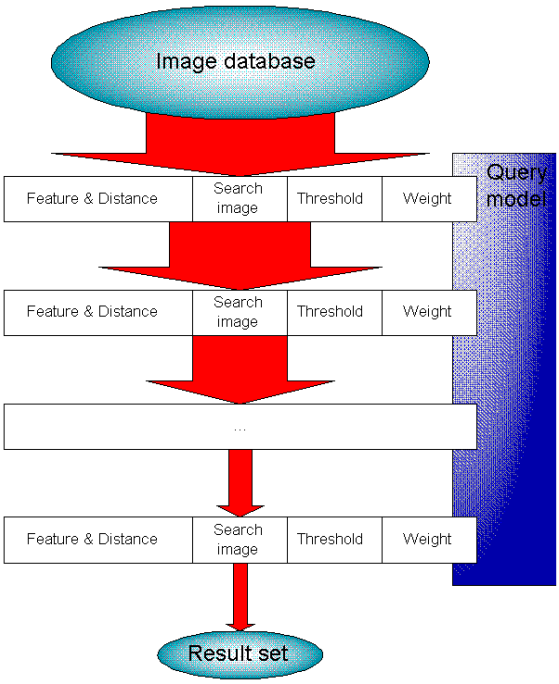

Figure 1: Image filtering

From a different point of view, query models can be seen as representations of clusters within the image database (Figure 2). A query model defines the properties of the corresponding image cluster. We shall see in section 4 that images of coats of arms are indeed grouped in such clusters. The next section shows how clusters can be exploited in an automated weighting algorithm.

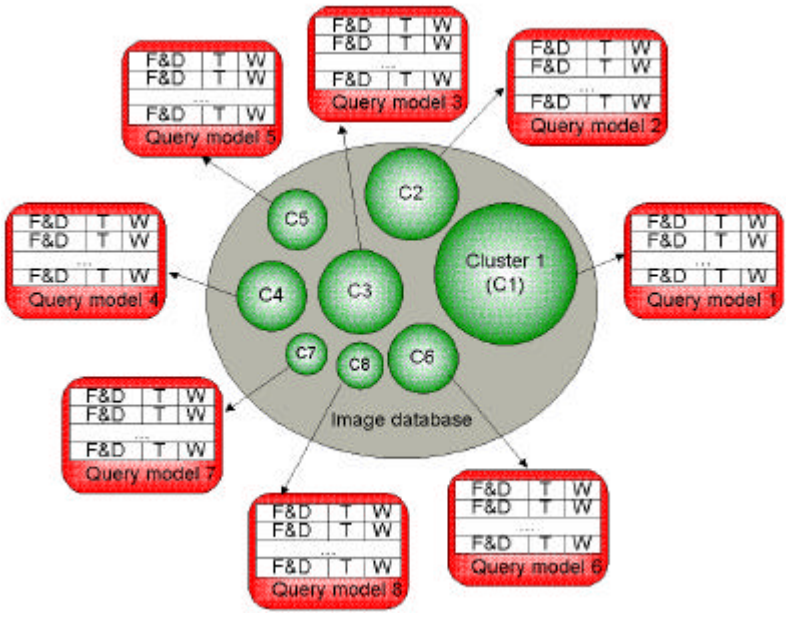

Figure 2: Clusters and query models

3. Merging of features through self organizing maps

This section describes how merging is performed and how suitable weights can be computed. In addition, we present some of the feature extraction functions. The results of feature extraction functions are composed into feature vectors that are used in the SOM ([13]) calculation.

Usually, (e.g., in [5]), when using multiple features for an image query, the result set is ordered by the weighted sum of the distance values (position value). The position value for each database object is defined by equation 1:

$$Position\ value_{Object} = \sum_{i=1}^{F} w_i d_i$$

In this formula F is the number of features, w_i the weight for feature i and d_i the distance value between the query object and the database object for feature i. This evaluation method assumes that all distance functions are standardized on the same range (in our case the interval [0,1]). Actually, a distance function is a measure for *dis-similarity*, that's why the distance between a search image and a candidate image should be small for important features but may be greater for less important ones. The weights should help to order the result set. The most similar image should be next to the query image and less similar ones should be placed at a greater distance. Therefore important features should have higher weights than less important ones to "punish" a greater distance for an important feature by a greater value for the product of distance and weight.

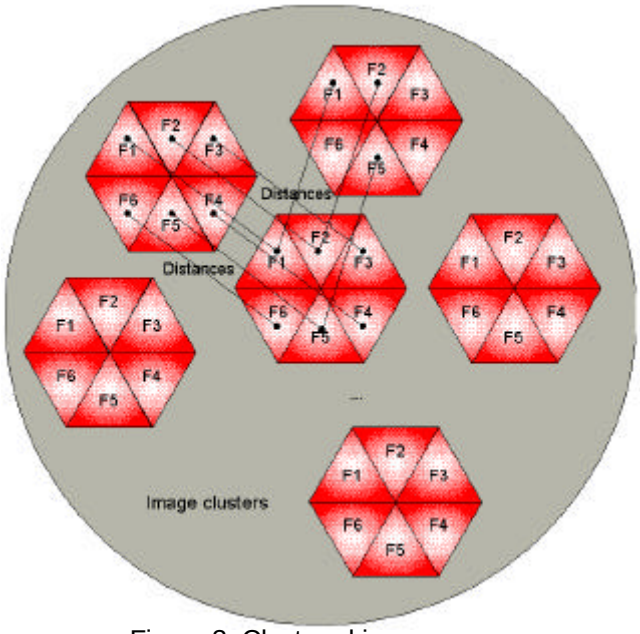

Figure 3: Clustered image space

If we would not use thresholds to limit the range of possible distance values, it could be possible that images appear in the result set that are similar in most aspects of the defined query model but not similar in some of them. Then it would be the task of the weighting process to order images at the end of the result set. This could hardly be achieved by the linear weighting method described above. It follows that using thresholds shifts the importance of the weighting algorithm from an essential part of a retrieval system to a more or less cosmetic operation.

Our idea is to cluster the image database and to use the contribution of each feature for the cluster structure for the selection of suitable weights. Clustering was performed by a self organizing map. The weight of a feature is calculated as the sum of distances (of the feature) between the cluster that contains the search image and all neighboring clusters. Figure 3 shows a clustered image space where six features (F1 - F6) are used. The weighting algorithm consists of the following steps:

1. Clustering of image database
   For each image in the coats of arms database [3] all features are calculated. Then the various feature vectors are exported, merged into a single vector (representing one image) and normalized. The normalized vectors of all images are fed into the map calculation algorithm in SOM-PAK ([13]) which produces a map with hexagonal layout. This means that each cluster has (max.) six neighbors. A cluster is represented by a feature vector pointing to its center.

2. Calculation of weights
   First, the cluster to which the search image belongs is identified and the weights for all features calculated (distance between search image and neighboring clusters). We experimented with two different approaches depicted in Figure 4: the distance between search image cluster and neighboring clusters (1) and the distance between the search image itself and neighboring clusters (2). We found in our tests that the first method is clearly better than the second one. Therefore the results in section 5 are computed by method 1. For the distance calculation itself the Euclidean distance was used.

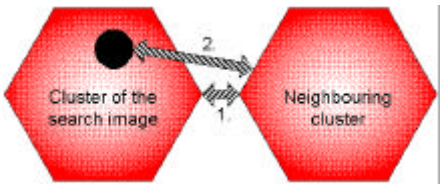

Figure 4: Weight calculation methods

3. Application of weights
   In the test environment weights are used as proportions; for example, for a query model with two features the pair of weights (2, 1) equals the pair (4, 2). Two different approaches were tested for the application of weights: the distance of a feature to all neighbors (1) and the reciprocal value of the sum of distances of all other features in the query model (2). Again, tests have shown that the first method is at least as good as the (more complicated) second one. We therefore used only the first method for the tests discussed in section 4.

We would like to close this section with some remarks on the features used to gain the vectors for the SOM calculation process. Besides the ones discussed in [2] and [3] (color histogram, symmetry features, segmentation, etc.) we used combined features as the one in Figure 5. This feature divides a shield into its elements and calculates a color histogram for each sub-region. The distance function analyzes whether two images have the same layout and returns 1 otherwise. In case of the same layout for each region the Euclidean distance between the two histograms is calculated and the average value returned.

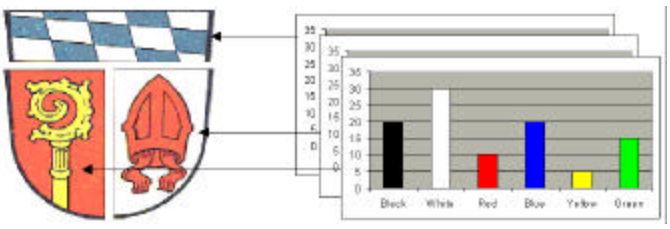

Figure 5: Combined segmentation and color histogram feature

Another feature uses the various symmetry features to find all symmetries in a given image (see Figure 6). The distance function for this feature examines whether two images share all or at least some symmetries. A third feature uses our object feature ([3], [18]) to determine the complexity of

an image. The complexity is defined by the number of objects in an image, the number of edges and mean and variance of the length of the edges and the angle between them.

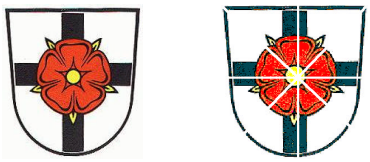

Figure 6: Symmetry description feature

4. Test environment and results

The test environment uses the IBM QBIC system [5] [8] as a kernel. Among its advantages are an easy to use C++-API and input filters for many image formats. The kernel was extended by a practical web-interface (a perl CGI-script), a search engine for query models (similar to QbQBE), the possibility to define thresholds and to limit the result size accordingly and some C-libraries for vectorization, object evaluation, etc. Features were programmed as C++-classes. Figure 7 depicts the test environment:

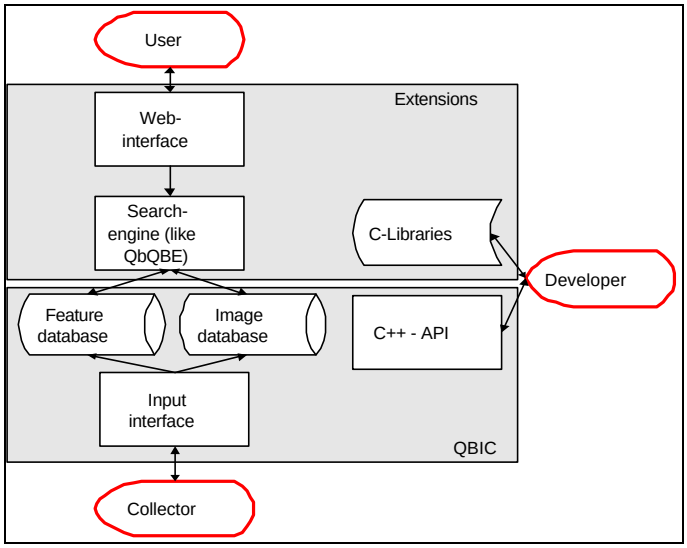

Figure 7: Test environment

The web interface consists of a query section for the definition of query models and two picture sections (to show the search results in weighted and not weighted ordering). The interface was derived from our standard GUI for database queries [3] and optimized for the evaluation of weighting algorithms. A typical screenshot of the interface is given in Figure 9. The coats of arms of our test database (444 pictures) were taken from a heraldry server in the netherlands [9]. Most of them are German civic arms and show the shield only. For each image 16 features resulting in a feature vector with 58 elements were calculated. All feature values were normalized to the interval [0,10]. The SOM was built by the tools in SOM-PAK (see [13]) with the following parameters:

Parameter	Value
Grid type	hexagonal
Map dimensions	8 x 6 bins
Neighbourhood kernel	bubble

For this small map it did not seem necessary to use the "gaussian" neighborhood kernel (see [13]). During the test session 50 tests with 550000 learning steps per test where performed. The average

quantization error (see [13]) for the best solution was 7.466932. Each bin of the chosen map contains between 2 and 30 images. A typical bin is depicted in Figure 8. The cluster consists of images with similar color histograms, no field division and average complexity.

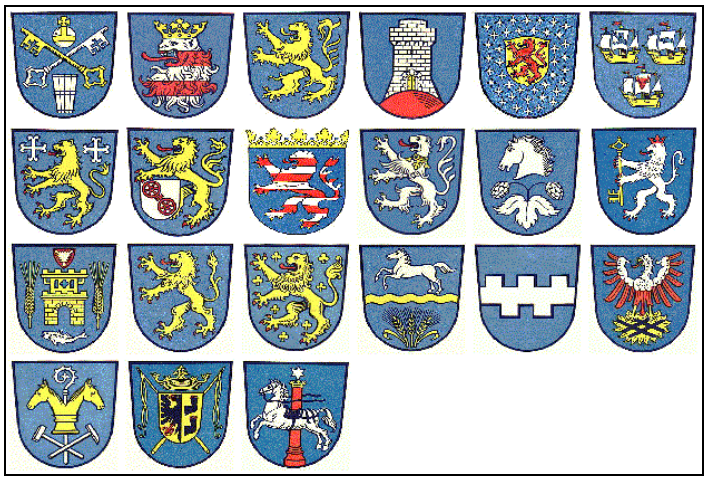

Figure 8: Typical image cluster

Verification was performed by the following steps: First, for each test query the five best images out of the first 12 result images were chosen. Second, for each of these images and the two weighting methods the distance from the actual position to the ideal position was calculated (error sum). The two weighting methods were: *constant weights*: - all weights are equal (1) and *SOM weights*: - all weights are derived from a SOM using the algorithm described in the previous section (2). Finally, the performance is calculated. The performance of a weighting method is defined as the ratio of actual error sum and the maximum possible error sum (in our case: 45).

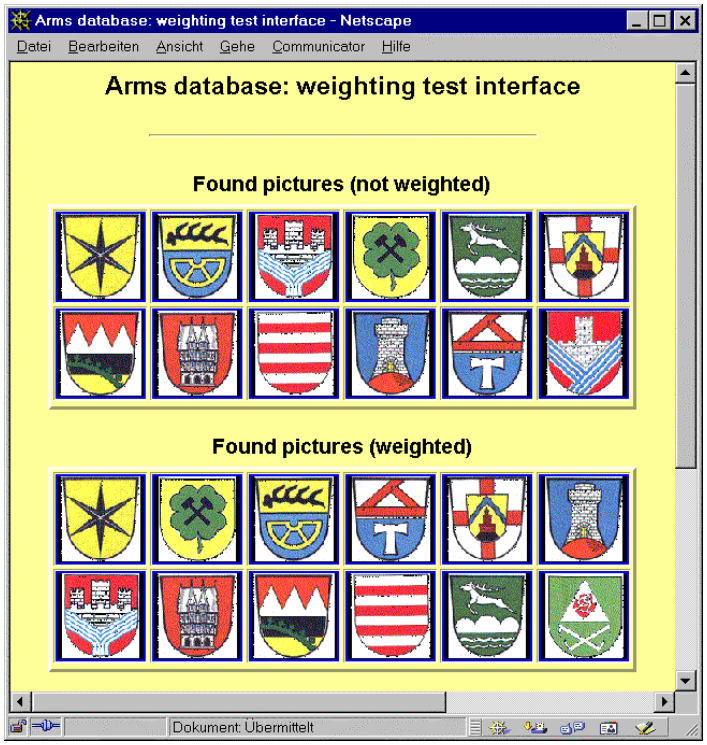

Figure 9: Screenshot of the weighting GUI

Figure 9 shows a typical screenshot of the weighting interface (developed specifically for the verification process). The first image is the search image. The query model consisted of two features: symmetry over the y-axis and similar low complexity. The latter was judged to be more important by the weighting algorithm and images with lower complexity were ordered ahead of more similar ones. It should be noticed that in this example the query model has not the purpose to describe similarity as perceived by humans.

Figure 10 shows the performance results for both methods: The performance of the SOM weights method is about 92% resulting in an improvement over constant weights of more than 8%. It is important to keep in mind that these values can only be compared by the verification algorithm we are employing. If we looked at the first twenty instead of the first twelve images the performance would probably be lower. However, the performance of the constant weights method in this case would be lower too and the improvement still significant.

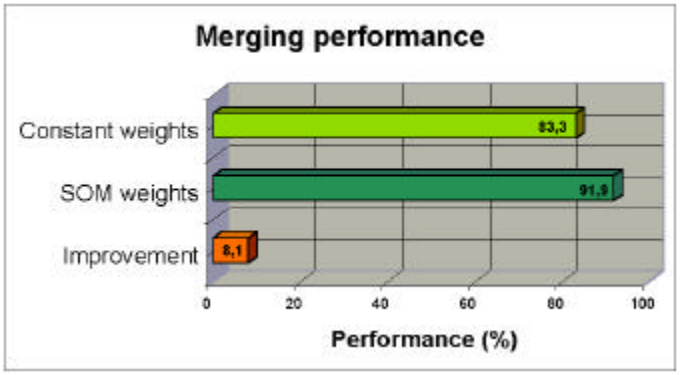

Figure 10: Weighting performance

5. Present and future research

When using thresholds it is not necessary to calculate every feature for every image in the database. In order to minimize the run-time of retrieval it is desirable to use features first that have the fastest distance functions and produce the smallest result sets. We have developed an algorithm to optimize run-time of a retrieval process in this sense. This algorithm consists of two parts:

1. a model predicting the likely result set size for each feature. Here the position of a feature in the query model is taken into account.
2. an optimization algorithm using the estimated size of the result set and the (known) performance of a distance function and calculating the optimal ordering.

Evaluation has shown that ordering performed by this algorithm is correct in more than 80% of all cases.

One of the main goals of the project is the automatic generation of query models. For this purpose we have implemented several algorithms for feature selection and threshold definition. These algorithms use the self organizing map described above and expert knowledge to derive the most suitable query model for a search image. It will be one of the next steps to tune and evaluate these algorithms and compare the results to those gained by a domain expert.

6. Conclusion

This paper shows how merging by linear combination of weighted distance values can be performed. A weighting algorithm is presented for the automatic computation of suitable weights for image features. These features are arranged in query models. The algorithm bases on a self organizing map which describes the "natural" clusters within an image database. We showed that using the contribution of a feature to the cluster structure as weight improves the ordering of query results. The implementation uses IBM QBIC system as kernel and runs under LINUX.

Bibliography

[1] Bach, J., Fuller, C., Gupta, A., Hampapur, A., Horowitz, B., Humphrey, R., Jain, R., Shu, C., "The Virage image search engine: An open framework for image management", Proc. SPIE Storage and Retrieval for Image and Video Databases

[2] Breiteneder, C., Eidenberger, H., A Retrieval System for Coats of Arms, Proc. of ISIMADE'99, 1999

[3] Breiteneder, C., Eidenberger, H., Content-based Image Retrieval of Coats of Arms, in: Proc. of the 1999 International Workshop on Multimedia Signal Processing, 1999

[4] Faloutsos, C., Indexing of Multimedia Data, in: Apers, P. M. G., Blanken, H. M. and Houtsma, M. A. W. (Eds.), Multimedia Databases in Perspective, Springer, Berlin, 1997, p. 219-246.

[5] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P., “Query by Image and Video Content: The QBIC System”, IEEE Computer, 1995

[6] Furht, B., Smoliar, S. W. and Zhang, H., Video and Image Processing in Multimedia Systems, 2nd print, Kluwer, Boston, 1996

[7] Goble, C., “Image Database Prototypes”, Advances in Databases: 13th British National Conference on Databases, Springer, Berlin, 1995, p. 365-375

[8] IBM QBIC – Homepage: http://wwwqbic.almaden.ibm.com/

[9] International Civic Arms: http://www.bng.nl/ngw/int/dld/germany.htm

[10] Introduction to heraldry: http://www.sca.org.au/lochac/scribes/hrld_int.html

[11] Neudecker, O., Coat of Arms Encyclopedia (in German), Bechtermünz, 1991

[12] Kochinka, B., Content-oriented search for multimedia-objects (in German), Technical report of the Technical University of Dresden, 1996

[13] Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., SOM-PAK: The Self-Organizing Map Program Package, Helsinki, 1995

[14] Laaksonen, J., Koskela, M., Oja, E., "Content-Based Image Retrieval using Self-Organizing Maps", 1999

[15] Rui, Y., Huang, T., Ortega, M., Mehrotra, S., Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval, IEEE Transaktions on Circuits and Videoo Technology, 1998, p. 644-655

[16] Sheikholeslami, G., Chang, W., Zhang, A., "Semantic Clustering and Querying on Heterogeneous Features for Visual Data", ACM Multimedia, 1998

[17] Soffer, C., Retrieval by Content in Symbolic-Image Databases, Technical Report, University of Maryland, 1995

[18] Steinböck, E., Extraction of essential structures from grey-scaled bitmaps as a preliminary stage for object-recognition (in German), master thesis Techncal University of Vienna, 1988