Content-Based Image Retrieval in Digital Libraries

Christian Breiteneder and Horst Eidenberger
Institute for Software Technology,
Vienna University of Technology, Austria, {cb, hme}@ifs.tuwien.ac.at

Abstract

Today's systems for Content-Based Image Retrieval (CBIR) suffer from several drawbacks: First, user interfaces are much too complicated for average users. Second, the quality of results tends to be low. Finally, querying performance with often long reply times is unsatisfactory.

We have developed a model for CBIR searches intended to overcome these drawbacks where the user has to select only one or more example images to initiate a query. Out of the examples the actual query including feature selection and weighting is generated automatically. The results of the first query may later on be refined by relevance feedback. We discuss the major components necessary for our approach and the results achieved in our test environment.

1. Introduction

Several of the more recent digital library projects include subsystems for Content-Based Image Retrieval (CBIR). Due to the nature of these projects in addressing large repositories, most of these subsystems address general domains. However, digital libraries may also cover information of very specific domains, that require more specific solutions. Systems therefore have to be extensible in order to include future application domains. This kind of extensibility requires that the general architecture and especially the user interface stays the same when new system components, as for example, new types of features are integrated.

However, extensibility of CBIR systems are not the crucial factor in the context of digital libraries. There are problems inherent to CBIR that require new architectures that enable especially the support of casual users. With this respect current CBIR systems suffer from several drawbacks:

- Complicated interfaces—Casual users are overtaxed by the demand for a definite opinion on similarity, the selection of features and especially, by the often necessary provision of weights. Many users would not even try a typical CBIR interface, if they had the opportunity to use it. To improve the acceptance of CBIR systems their interfaces have to be made much simpler.
- Unsatisfactory querying performance—CBIR systems use distance functions to calculate the dis-similarity between a search image and database images. This process is often very slow and reply times in the range of minutes may occur for large databases.
- Low result quality—By using only general features for all types of images and asking the user to choose features leads to low quality retrieval results.

In this paper we outline the major components of a CBIR system that implement various algorithms to overcome or at least reduce these problems. The system was developed for the application domain of coats of arms. The system includes features for coats of arms, models for retrieval, similarity definition and algorithms for query definition, acceleration and refinement. Despite being developed in a rather narrow and specific test environment most of the methods can be used for any other application domain.

Section 2 summarizes related work. Section 3 describes the general concept of our approach for a user friendly CBIR system and our model of similarity. Section 4 addresses the implementation of algorithms, Section 5 and Section 6 present the test environment and the results achieved.

2. Related work

The focus of this section is on CBIR systems, the computer centric approach and relevance feedback. Famous CBIR systems are IBM's QBIC [3], Virage [1], VisualSEEk [12] and Photobook [7]. All these systems ask the user to select feature(s) and sometimes,

suitable weights. They offer Query by Example and (most of them) sketch-based queries.

Almost all CBIR systems follow the so-called computer centric approach (CCA, [8]): querying is done with feature and distance functions where the user has to select features for a query and the weights to determine their relative importance. The CCA has two major drawbacks:

- The semantic gap—This term addresses the difference between high level concepts for CBIR and the low level features that are used for the querying ([9], [10], [11]). The authors of [10] argue that features will never represent similarity to a sufficient degree and CBIR therefore can only make correlations between a search image and result sets visible.
- Subjectivity of human perception—Different persons or the same person in different situations may judge visual content differently. This problem occurs in various situations: different persons may judge features (color, texture, etc.) differently, or if they judge them in the same way they still may perceive them in different ways.

One approach to reduce the problems of the CCA in the field of retrieval system design is to replace the classic one-shot query with an iterative process that utilizes a user's judgement of the query results. By relevance feedback the user may iteratively refine his query and increase the quality of results ([9], [6], [12]). This procedure helps to close the semantic gap as far as possible. In [9] a multi-feature approach is presented with color histograms, Fourier coefficients, Tamura features, etc. Merging of the features is done by weighted linear merging (see Section 4) where the weights are adjusted by the feedback of the user, which images of the query result are relevant and which are not.

3. Click & Refine Model

Our approach, the click & refine model, aims at a more global solution to increase the usability of CBIR systems.

1. The first query is generated automatically. A user selects one image or a group of images for which he wants to find similar ones. The system is able to conclude from the examples a user's idea of similarity and to select suitable features for the querying process.
2. Out of the results of the first query the user can find additional relevant information by iterative refinement.

Since the quality of the first query is very important we developed models and algorithms in order to optimize these results. Figure 1 summarizes the data flow in the Click & Refine model. The main advantage of this model is the user friendliness: the user does not have to deal with features, similarity measures or the number of images to return. All the user has to do is specifying what he wants by clicking on an image and judging the relevance of query results. We think that all other tasks, which are involved in a query, can be successfully automated and that the results of this highly automated querying process are not worse but most times better than the results an average user can achieve with a traditional CBIR query engine.

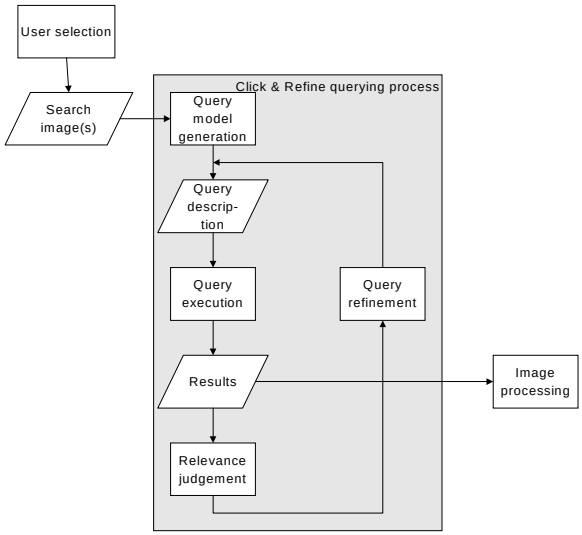

Figure 1 Click & Refine model.

3.1. Query Models

The basic idea in our approach is the combined use of several simple and robust features, that can be general-purpose or application-specific. The selection of features for a certain query should depend on the example image(s) the user chooses. For this application independent approach to define similarity user and query related in CBIR systems we developed a general form called query models. A query model is a set of layers, which are used to define one aspect of a similarity term, the overall idea of similarity is the aggregation of all layers. Each layer consists of the following elements:

- feature extraction function
- distance function

- parameter for result size determination. We use a threshold value for the maximum distance an image is allowed to have in order to be in of the result set.
- weight. We use linear weighted merging to compress the distance values of all layers to a global distance value (see Section 4).

Figure 2 shows the information (image) flow in a query model. Each layer acts as an image filter eliminating images with a greater distance to the example image than the threshold value. Finally, the result set consists only of images meeting all conditions (layers). From a different point of view query models can be seen as representations of clusters of more than average similar images in the database. Methods to make the clustering of a database visible include various forms of cluster analysis and the neuronal networks-based Self-organizing maps. Each cluster has specific properties that are described by query models.

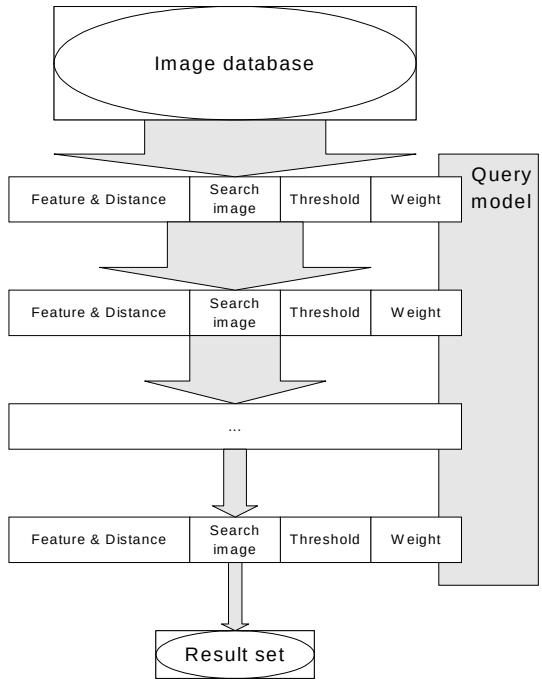

Figure 2 Query model.

Query models can be additionally exploited to reduce the reply time in a querying process. Succeeding layers in a query model need only process images, that were not rejected by any predecessors. We therefore designed and use two different types of features: features with fast distance functions, which eliminate a large quantity of not relevant images by coarse selection and features with complicated distance functions for fine-graded selection. Features for coarse selection are used as the first layers in a model and those for fine selection employed later on.

To perform the first query in the Click & Refine model a query model has to be generated; features have o be selected, threshold values and weights identified and layers ordered. The accomplishment of these tasks is based on the following information:

- The example image(s)—Each image or group of images has specific properties (colors used, symmetry, complexity, etc.). These properties have to be identified and a decision made, if they are significant enough to be further investigated. If yes, proper features have to be selected and added to the query model.
- Cluster information—The clustering structure of an image database can be made explicit by a cluster analysis. From the general clustering pattern and the clusters to which examples belong to conclusions on the features forming the clusters can be drawn.
- Expert knowledge—includes knowledge about generally useful features and relationships between features that help to decide, which features to employ.
- Additional information—generated when images are stored in the database: statistics about the distribution of the image features, the distance to reference objects, etc.; useful when thresholds have to be determined.

After the first query has been executed methods for refinement have to be placed at the users disposal.

4. Implementation

To generate a suitable query model for the first shot, we had - using the information described above - to solve the following problems:

- Selection of suitable features. As shown in Section 6, it is not a good strategy to apply all available features in a query model. To define the best possible query model those features have to be chosen that correspond to the properties of the example image(s). We have developed several methods for feature selection depending on the number of example images (one or several).
- Definition of a suitable threshold value for each feature. The critical issue is setting the thresholds low enough for a good precision value and high enough for a good recall value. We derive thresholds from

cluster information and expert knowledge on image classes and distance derivations (referring to a reference object).

- Selection of fitting weights for a suitable rating of the query model layers. The weights determine the ordering of the images in the result set. Due to the usage of thresholds weights have no impact on the content. Weight definition is therefore an easier task than threshold definition and performed by exploiting cluster information.
- Performance-optimized ordering of the layers to produce results as fast as possible. A good optimization algorithm is very valuable and saves considerable time, but is not easy to implement. For our approach we need to predict the number of results of each layer in a query model and any threshold.

4.1. Query Model Generation

We developed a set of algorithms for the generation of query models from one example image and another set for the generation from two ore more examples. In this paper only the first case is discussed. Features are selected by the evaluation of striking properties of the example image and information on the cluster structure of the image database. Thresholds are derived from a prognosis database, which stores information on the computation of the distance of image classes to reference objects, and from the cluster structure. To find proper features, we developed two different methods:

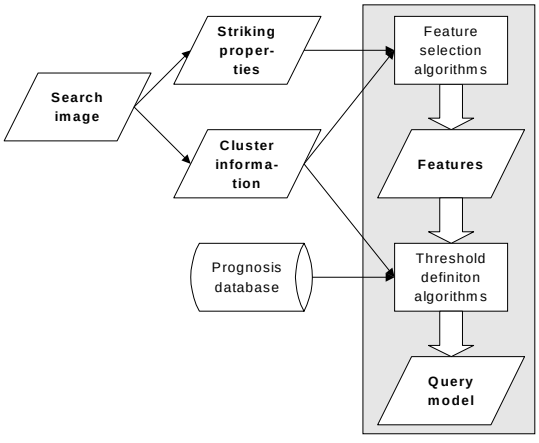

Figure 3 Model generation with one example image.

- Use all features meeting the following condition:

$$w_f > f(\mu_w)$$

where w_f is the weight of feature f, μ_w the mean over all weights and f a suitable linear function. This method is based on feature clustering by self-organizing maps and therefore called the *SOM method*. Since the weight of a feature (or layer) is a measure of its importance it seems only reasonable to employ this information for the feature selection.

- Use all features, that for the given example image satisfy a certain condition ("striking properties"). For example, for a feature counting the number of color shades (described in [2]), the feature vector has only one element, f_0, we used the following condition. In other words, this feature is used in a query model if the number of color shades is less than 4 or greater than 10.

$$f_0 < 4 \vee f_0 > 10 \xrightarrow{?} use \quad feature$$

For each feature a suitable condition has to be defined. We therefore call this method the *condition method*. The condition describes a pattern of feature vector indicating the importance of a specific feature. In an additional step we use the two methods in combination and employ all features selected by one or both of them.

4.2. Threshold Definition

After features are selected, thresholds have to be defined. We have developed two alternative methods for this task:

- Setting thresholds in such a way that all features eliminate an equal portion of the image database. For this purpose the prediction of the number of images a specific feature and threshold combination would eliminate is required. For this purpose we use the prediction model of the performance-optimization algorithm. The method is called *shared method*, since all features participate equally. The major disadvantage of the method is the use of an absolute number describing the size of the result set. From this number thresholds are calculated back for all features. In other words with this method it is not possible to take the full advantage (better quality of results) of thresholds.
- Deriving the thresholds values from features weights: more important features should have lower threshold values to guarantee that returned images have similar

properties. We defined this threshold by the equation

$$t_f = 1 - f\left(\frac{w_f}{\sum_{i=0}^{F} w_i}\right)$$

where w_f is the weight of feature f and F the number of features. This method does not guarantee that the threshold is set high enough for the considered feature to accept any image at all. In addition to the two basic methods, we tested combinations of the two methods.

4.3. Weight Selection

After the selection of features and thresholds for the generated query models each layer of the models has to be weighted. As we have seen above the weights delivered by a weighting function are important for the first two steps of the model generation as well. The weight is a measure for the importance of a model's layers. In the next paragraphs we shall show how linear weighted merging is performed and how suitable weights can be computed.

4.3.1 Weight Selection Principles

Usually, (e.g., in [4]), when multiple features for a query are employed, the result set is ranked by the weighted sum of the distance values (position value). This evaluation method assumes that all distance functions are standardized upon the same range (in our case the interval [0,1]). The major advantage of this method is the simple calculation and application, the major disadvantages are the fact that not all features show a linear relationship and that linear merging therefore is not a suitable method to join such features, and that in most systems weights have to be provided by the user.

A distance function is actually a measure for *dissimilarity*, the distance between a search image and a candidate image should therefore be small for important features but may be greater for less important ones. Weights should help to rank 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.

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 less important operation.

Our idea is to cluster the image database by the global feature vectors (merged sum of all feature vectors) and to use the contribution of each feature to the natural cluster structure for the selection of suitable weights. Clustering was performed with 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. Therefore the weight is a measure for the contribution of a feature for the discrimination between the search images cluster and its neighbors.

4.3.2 Weight Selection Algorithm

The weighting algorithm consists of the following steps:

1. Clustering of image database. For each image in a database all feature vectors are calculated. Then the various feature vectors are exported, merged into a single global vector (representing one image) and normalized. The normalized vectors of all images are fed into the cluster map calculation algorithm for self-organizing maps (SOM-PAK) producing a map with hexagonal layout. As a consequence each cluster has at most six neighbors. A cluster is represented by a feature vector pointing to its center (median). The Euclidean distance is used as similarity measure for clusters.
2. Calculation of weights. The cluster to which the search image belongs to is identified and the weights for all features calculated as the Euclidean distance between search image and neighboring clusters. We experimented with two different approaches: the distance between search image cluster and neighboring clusters and the distance between the search image itself and neighboring clusters. In our tests we found that the first method is clearly better than the second one.
3. Application of weights. In the test environment weights are used as proportions; for example, for a query model with two features the weights (2, 1) equal the tuple (4, 2). Two different approaches were tested for the application of weights: the distance of a feature to all neighbors and the reciprocal value of the sum of distances of all other features in the query

model. Tests have shown that the first method is at least as good as the more complicated second one. Therefore the first method was chosen for the final version of the algorithm.

The final implementation of the weighting algorithm is a function, that for a specific search image and feature delivers a real value to describe the importance of the feature for a query on the search image. The major problem of our algorithm in a vivid system is the employment of information stemming from static clustering methods. Possible approaches to reduce this problem are a recalculation of the cluster map after N inserts or time T (may be computational expensive) or to cluster a set of reference objects instead of the whole database (inaccurate).

5. Test Environment

Our test environment for the Click & Refine model is based on IBM's QBIC system (version 2). Despite its advantages we had to make a lot of changes and extensions to the original system. One was the search engine for query models, which uses the generation algorithms that were implemented as C/C++ libraries. This query engine (NetSrv) produces in addition to a result set a lot of useful statistics for the evaluation of our algorithms. Another major part of the test environment are the various web interfaces for Click & Refine and the algorithm tests. The clustering of feature vectors and elements was done with SOM-PAK, where hexagonal maps were used with a linear neighborhood kernel (see [5]). Figure 4 shows the major components of the test environment: the search and the input interface. The important databases in the system are the image database, the feature database and the prognosis database.

A major part of the test environment consists of the 19 features for the application domain of coats of arms (implemented as C++ classes, derived from the QBIC standard feature class). They fall into three groups: general color features (number of colors, coats of arms color histogram, etc.), sketch-based features (object layout feature, etc.) and application specific features (seal print, arms segmentation, etc.). Out of these 19 features 15 are used in the generation algorithms.

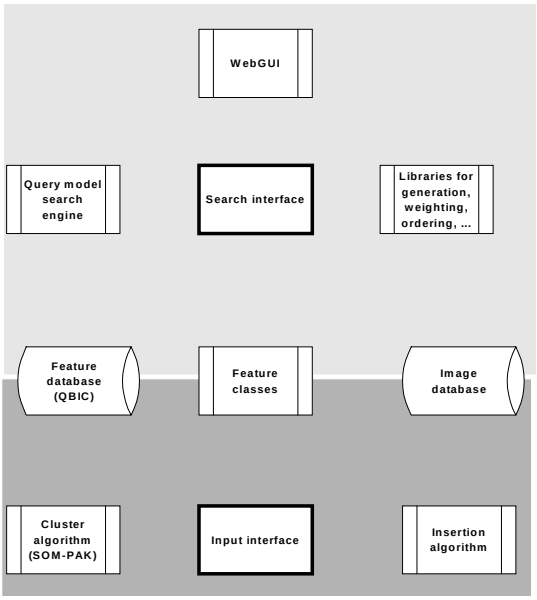

Figure 4 Test environment.

6. Tests and Results

6.1. Feature Ordering Tests

To test the ordering algorithms thousand synthetic query models with 2 to 10 layers and meaningful threshold values were generated, the optimal order (with the knowledge from the database, etc.) calculated and results compared. The performance of the algorithms is defined as the ratio of correct orderings and total orderings. Figure 5 shows the performance of four different algorithms:

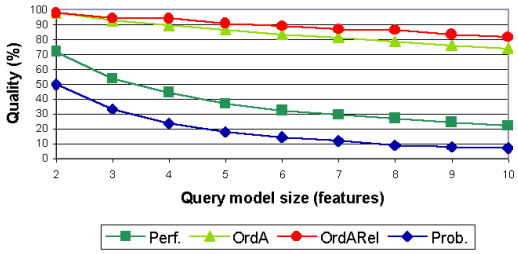

Figure 5 Results for feature ordering algorithms.

- Random ordering (“Prob.” method)—query models are not ordered at all.
- Performance heuristic (“Perf.” method)—query models with faster distance functions are used first.
- Ordering algorithm without relationship model

("OrdA" method)—relationships between features are not considered.

- Ordering algorithm with relationship model ("OrdARel" method).

The graphs show the decreasing cumulated performance of query models with 2 to 10 layers. The overall performance of the Prob. method is only 5%, the Perf. method reaches 20%, OrdA has a performance of about 75% and OrdARel of more than 81%. In other words the ordering algorithm produces the correct order in more than 80% of all cases. Tests have shown that using the OrdARel algorithm for feature ordering reduces the average query time in our test environment from 190.7 ms to 64.6 ms for the OrdARel method. This is an improvement of 126.1 ms (66%). If the ordering were always correct a query would take 58.7 ms which would only be 6.1 ms faster than OrdARel (9.4%).

6.2. Weight Definition Tests

Tests of the weighting algorithm were performed in 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 by the weighting algorithm. Finally, the performance was 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). For the tests we developed a special web interface, which shows the result set of a query twice: with constant and SOM weights.

The constant weights had a performance of 84%. The performance of the weighting algorithm turned out to be 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 with the test algorithm we are employing.

6.3. Query Model Generation Tests

For testing the generation algorithm we use recall and precision with the focus to optimize recall and keeping precision reasonable.

We first tested the quality of the basic methods for feature selection and threshold definition. To test the feature selection methods we made several queries for each feature selection method and various threshold definition methods. Additionally, we compared the performance of our algorithms with the case of using all available features together. Besides the bad query computation performance of the latter method it turned out that the recall of this method is much worse than the ones of our algorithms.

The best recall was produced by the condition method (79%). This is considerably higher than the recall for the second basic method, the SOM method (56%). The reason why these values are rather poor is that they are averaged over all threshold definition methods. We will see later, that the best combination of basic methods produces quite acceptable results. Using all features may result in such a bad recall (25%) because similarity here is defined globally and not according to the specific qualities of the given image class.

Next we investigated the performance of the different threshold definition methods. Again, we tested each method with every available feature selection method and calculated the mean for recall and precision. The recall of the shared method is much higher (74%) than the one of the alternative method (41%). Surprisingly, the combined method improves the recall by 3%.

After testing the basic methods we verified all possible combinations to identify the best algorithm for automatic query model generation. Additionally, we compared the results of the generated models to the results a human expert could achieve. Figure 6 shows the resulting recall and precision values.

The best combination comprised the two best basic methods: feature selection by striking properties and threshold definition by the shared method with additional cluster information. A recall value of 94% and a precision value of 68% were gained.

Image class and query environment experts reached a recall of 83% with a precision of 91%. The decrease in the recall compared with the best generation algorithm can be explained by the way the expert tests where performed: we used the best generated query model and tried to improve the precision by adapting the threshold values without significantly dropping the recall. This test is an application of the Click & Refine model.

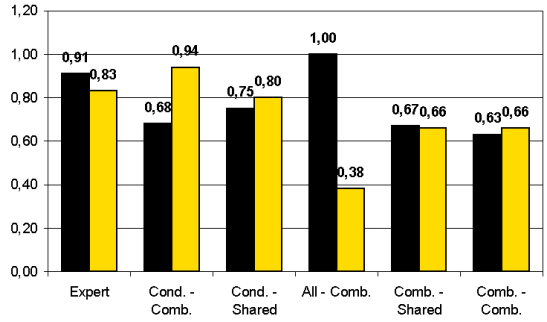

Figure 6 Precision and recall for query model generation.

7. Conclusion

We presented a new, more user-friendly approach to content-based image retrieval that is based on the Click & Refine model. The central concept is the liberation of users from providing parameters as e.g., features and weights. Queries are automatically generated from example images. The central concept of the approach is the query model. Query models allow for

- the subjective definition of similarity. They integrate classic CBIR concepts with our threshold approach to improve the results of image retrieval search engines.
- the reduction of querying time.
- the easy combination of different features with various complexity, purpose or application domain.

The concepts were implemented and tested in various algorithms for

- the automatic generation of query models (from example images, cluster information, expert knowledge, etc.); the generation of query models includes feature selection, threshold definition and weight selection.,
- performance-optimized ordering,
- a query engine for query models and
- features for the application domain in our test environment: coats of arms.

References

[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*, 1996.

[2] Breiteneder, C., Eidenberger, H., "Content-based Image Retrieval of Coats of Arms", *Proc. of the 1999 International Workshop on Multimedia Signal Processing*, Helsingör, pp. 91-96, 1999.

[3] 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.

[4] IBM QBIC homepage; http://wwwqbic.almaden.ibm.com/

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

[6] Nastar, C., Mitschke, M., Meilhac, C., "Efficient Query Refinement for Image Retrieval", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.

[7] Pentland, A., Picard, R. W., Sclaroff, S., "Photobook: Content-Based Manipulation of Image Databases", *SPIE Storage and Retrieval Image and Video Databases II*, 1994.

[8] Rui, Y., Huang, T., Chang, S., "Image Retrieval: Past, Present and Future", *International Symposium on Multimedia Information Processing*, Taiwan, 1997.

[9] Rui, Y., Huang, T., Ortega, M., Mehrotra, S., "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval", *IEEE Transactions on Circuits and Systems for Video Technology*, 1998.

[10] Santini, S., Jain, R., "Beyond Query By Example", *ACM Multimedia*, 1998.

[11] Santini, S., Jain, R., "Integrated Browsing and Querying for Image Databases", *IEEE Multimedia Magazine*, 1999.

[12] Smith, J. R., Chang, S., "VisualSEEk: a fully automated content-based image query system", *ACM Multimedia*, 1996.

[13] Wood, M., Campbell, N., Thomas, B., "Iterative Refinement by Relevance Feedback in Content-Based Digital Image Retrieval", *ACM Multimedia*, 1998.