PERFORMANCE-OPTIMIZED FEATURE ORDERING IN CONTENT-BASED IMAGE RETRIEVAL

Horst Eidenberger
Austrian Libraries Network, Ministry of Science and Transport
Garnisongasse 7/21, A-1090 Vienna, AUSTRIA
Tel: +43 1 4035158 14; fax: +43 1 4035158 30
e-mail: hme@bibvb.ac.at

Christian Breiteneder
Institute for Software Technology, Interactive Systems Group, Vienna University of Technology,
Favoritenstrasse 9-11/188, A-1040 Vienna, AUSTRIA
Tel: +43 1 58801.18851; fax: +43 1 58801.18899
e-mail: breiteneder@ifs.tuwien.ac.at

ABSTRACT

We present a method to improve the performance of content-based image retrieval (CBIR) systems. The idea is based on the concept of query models [1], which generalizes the notion of similarity in multi-feature queries. In a query model features are organized in layers. Each succeeding layer has to investigate only a subset of the image set the preceding layer had to examine. For the purpose of performance acceleration we group features into two types: features for quick elimination of rather not similar images and features for the detailed analysis of result set candidates. Performance optimization is based on a model for predicting the number of images to be retrieved and on a model describing relationships between features. Results in our test environment show significant reduction of query execution time.

1 INTRODUCTION

The increasing number of digital libraries and databases with visual content requires powerful solutions for content-based image retrieval (CBIR). In most approaches image features are extracted, stored in a database and compared with the features of a particular search image. The result set of a search should only contain images the features of which show a minimal distance in feature space (nearest neighbor searches). However, similarity computation may be very expensive, since feature spaces in general have a high dimensionality. Research therefore addresses topics such as reduction of feature space dimensionality and multi-dimensional data structures and search methods.

The approach presented in this paper intends to gain better search performance on top of these methods by taking into account that a similarity search usually comprises a set of features with distance functions of varying performance.

The reminder of the paper is organized as follows: Section 2 discusses the query model concept and explains the underlying motivation. Section 3 introduces the components of the performance optimization approach: models for prediction and feature relationships and the optimization process. In section 4 we describe how the approach was implemented in our test environment and discuss the results gained.

2 QUERY MODELS

Similarity in our CBIR system is defined by *query models* [1]. 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 search 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 retrieval approaches - by an absolute number. Every entry in a query model eliminates some images until the result set is computed.

The underlying idea of query models was the development of a set of robust features each addressing a specific purpose and the employment of a feature subset for a specific similarity search. However, using query models has an additional advantage that was not originally thought of and is the topic of this paper: Query models enable the reduction of retrieval time. Features having the fastest distance functions and / or reduce the number of images in the result set most should be ranked higher in the query model. Therefore it seems to be useful in a CBIR system to have two different types of features at hand:

1. Features for the quick elimination of images that for certain are not members in the result set. These features are used first in every query model. A typical example would be our feature for computing the number of color shades in an image. In our test environment the feature is employed to decide whether the image in question is of natural or synthetic origin [2].
2. Features for the detailed analysis of result set candidates. These features are likely to have more complicated distance functions consuming more time for computation than those of the first type. They are

therefore used later in a query model. An example for this kind of feature would be the object layout of two images [2].

The implementation of a query engine integrating these ideas requires solving the following problems:

1. Prediction of the number of images a feature in the query model passes to the next layer depending on its position in the query model. For this purpose it is necessary to predict the number of images a feature will select when it is the first in a query model. In addition, relationships among features have to be identified in order to predict the number of images, which two consecutive features would both reject.
2. Defining an optimization model, which takes into account the (measured) performance of the feature distance functions and the estimated number of images a feature will not reject to calculate the performance-optimized ordering of features in a query model. For this model a suitable optimization algorithm has to be implemented and integrated into the query engine.

3 FEATURE ORDERING

The concept of feature ordering is based upon three components that produce the information needed for the performance-optimized ordering:

1. *Prediction model*—A model to predict the number of images retrieved that a feature would rate similar if it is the first in the query model. The purpose of this model is to define the starting point for each feature in the ordering process and to select the first feature in a query model.
2. *Relationship model*—A model to describe the relationships among features by a similarity value. By this model and the information from the prediction model it becomes possible to predict the result set size of each feature of a query model. The prediction is independent from the position of the feature in the query model and the number and type of other features.
3. *Optimization algorithm*—for the ordering of features in a query model by the performance of their distance functions. In addition, the information from the prediction and the relationship models are employed.

Figure 1 shows the data flow in the ordering process. The prediction model uses a three-dimensional array to store the distribution of the number of images in the result set per threshold value for each feature and each class of images. Figure 2 depicts an example of this structure for one feature. The class of an image is detected by comparing its feature vector to a feature-specific orientation point. For this purpose the distance function of a feature class is used. For example, a feature that simply counts the number of colors in an image and stores this value in a one-dimensional feature vector could have the orientation point op = (1). If the distance function for this feature returns the absolute difference between two vectors then the class of an image with three colors (feature vector $v_1 = (3)$) would be 2 and would be 5 for an image with six colors ($v_2 = (6)$).

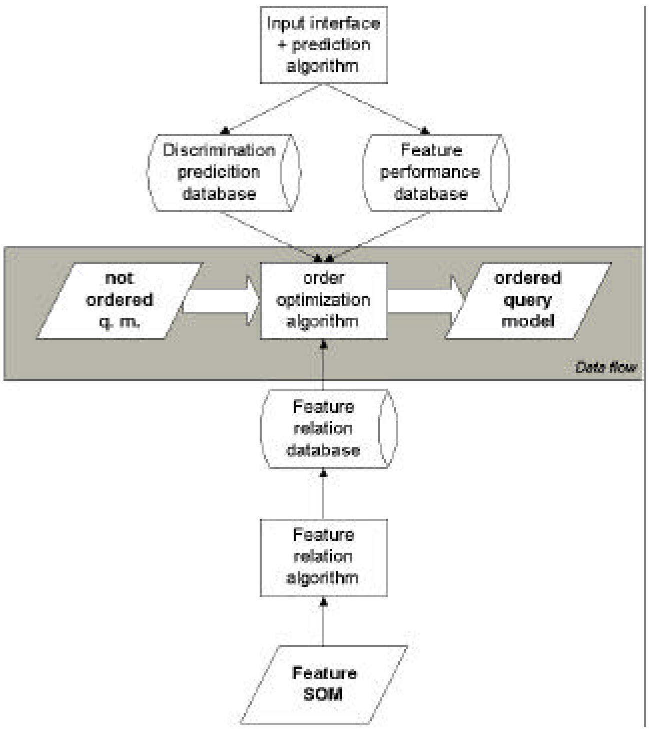
Input interface + prediction algorithm
Discrimination predicition database
Feature performance database
not ordered q. m.
order optimization algorithm
ordered query model
Data flow
Feature relation database
Feature relation algorithm
Feature SOM

Figure 1: Ordering process

The order-distribution array is updated whenever an image is inserted or deleted in the database. On every insert or delete operation the following actions have to be carried out for each feature: calculation of the image class as the distance to the reference object, calculation of the distance to every other image in the database and update of the order distribution. In other words, in a system with n features every update of the content base equals n user queries with only one feature. This is acceptable since this computation is not time-critical and can be performed in the background when computational resources are available.

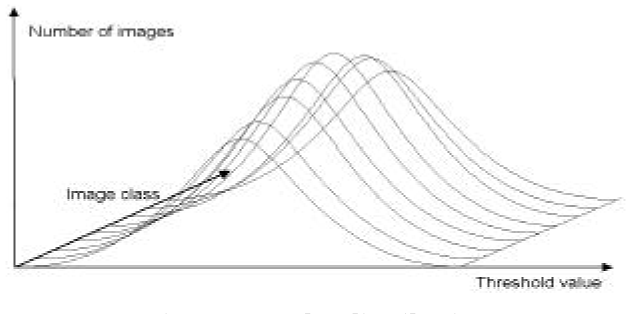

Figure 2: Order distribution

The prediction method was tested for fifteen different features with one- to six-dimensional feature vectors. The average performance was more than eighty percent and most features had a performance of more than ninety percent.

The basis for the relationship model is a Self-Organizing Map (SOM; [4]) that clusters the elements of feature vectors into groups. The fifteen features used sum up to 58 vector elements; we used 444 example images to train a hexagonal map with eight columns and six rows. Using this map we calculate the similarity value for two features with the following algorithm:

1. For each element of the longer feature vector: find the corresponding element of the second feature vector with the smallest distance. The distance of two vector elements is defined as the Euclidean Distance of the two SOM-clusters the elements belong to.
2. These distance values are added to a global distance value.
3. The relationship of two features is the standardized, limited global distance of two features.

This similarity value is a measure for the similarity of the elements of the feature vectors. To be usable in our model, we feed these values together with the results from the prediction model into a linear function. The function calculates the size of the proportion of the basic image set which both features would eliminate. The parameters of this function are determined by heuristics. The size of the result set of the second and all succeeding features is computed by subtracting the aggregated output of this linear function from the result set size of the first feature.

The optimization algorithm uses the results of the prediction and the relationship model together with the measured performance values for each feature to find the optimal ordering for a query model. The goal function is described in equation (1):

$$\min : Time = \sum_{i=0}^{Features} p_i A_i \qquad (1)$$

where p_i is the performance of feature i and A_i the size of the image set examined by feature i. A_0 is the size of the image database. This means the goal is the minimization of the query execution time. The values for A_i are derived from equation (2):

$$A_i = A_{i-1} - R_{i-1} = \ldots = A_0 - \sum_{j=0}^{i-1} R_j \qquad (2)$$

R_j is the predicted number of images, which the feature j will cut off A_j. These values are estimated using the output of the prediction and relationship model. One might argue that it is always suitable to use the fastest feature first in order to minimize the product of p_0 and A_0. However, it is not guaranteed that this feature will eliminate as many images as some other feature and therefore keep A_1 small. Each feature - and especially the first - has an impact on all succeeding features. Currently, we are not using an optimized algorithm to solve the optimization model but try out every possible order to find the optimal solution. The performance values are collected during the normal database operation and measured in milliseconds (ms).

For the implementation of the ordering algorithm we used a database of 444 images of coats of arms. The 15 features mentioned above include general-purpose features (color histogram, number of colors and color shades, shape features, etc.) as well as domain specific features (e. g., segmentation of arms, complexity, seal print analysis, etc.) The retrieval environment uses IBM‘s QBIC system (Version 2; [3]) as a kernel, SOM-PAK [4] for feature vector element clustering and runs on a Linux workstation.

4 PERFORMANCE EVALUATION

The prediction model has an accuracy of at least 80% (actually returned images by prediction) for each feature. For many (those with feature vectors of only one or two elements) it is more than 90% - 95%. For testing the performance of our approach we compared the following methods:

1. An unordered query model supplied by the user and probably ordered correctly by chance. This method is called the *Probability method* below (*Prob.)*.
2. Heuristics where the query models are ordered by the performance values of the features. This method does not take into account the number of images a feature eliminates. It is called the *Performance method (Perf).*
3. Our optimization algorithm without the relationship model. This method supposes all features being independent from each other. In other words, the set of images two features would both eliminate would be always empty. This method is called *OrdA*.
4. The optimization algorithm with the relationship model (*OrdARel*).

The performance of a model is defined as the number of cases, in which the suggested ordering was correct divided by the total number of tests. Figure 3 shows the accumulated performance of these four variations of query models with two to ten layers. For each graph 1000 tests were performed with generated but likely query models.

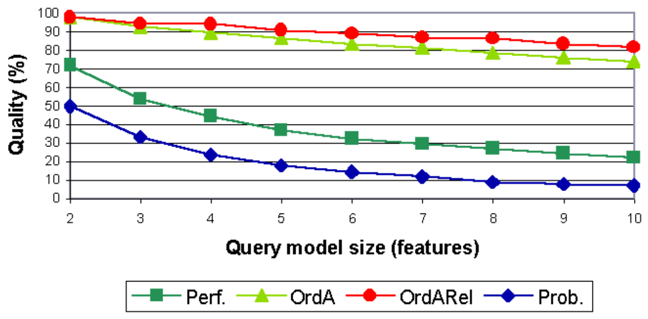

Figure 3: Ordering performance

OrdA and OrdARel produce an almost linear performance graph whereas the other methods show an exponential decrease. Therefore the quality of these algorithms must be rated higher.

The average performance of OrdA for query models with at most 10 layers is about 73%. This is more than 50% better than the performance heuristics, which again is 15% better than the Prob. method. When using the relation model together with the prediction model to estimate the R_j the performance improves by further 8% to 81%. This means that our algorithm produces the correct order in more than 8 of 10 cases, which is about as good as the basic performance of the prediction model.

Figure 4 shows the magnitude of error in the case a model fails. For OrdA and OrdARel the error in query execution time is smaller than 10% in 70% of all cases. Surprisingly, the error - when feature relations are taken into account - is often a little bit higher than for OrdA. The Perf. method is not depicted in this figure because it appears almost linear.

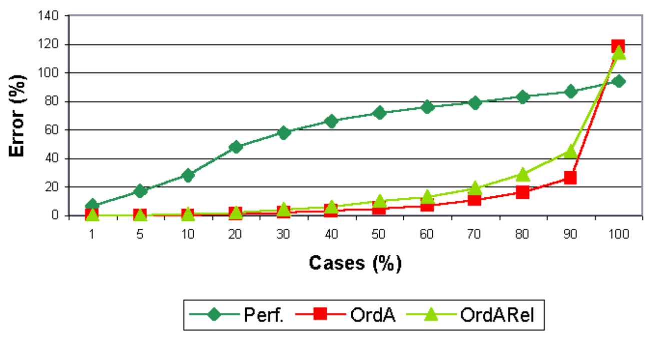

Figure 4: Ordering errors

Using the OrdARel algorithm for query model ordering reduces the time a query consumes in our test environment from an average of 190.7 ms for the performance heuristics to 64.6 ms for the OrdARel algorithm. This is an improvement of 126.1 ms or 66% in comparison to the performance heuristics. Optimally - if the ordering was always correct - a query would take 58.7 ms which would only be 6.1 ms faster than OrdARel (9.4%). The computation time of the optimization algorithm itself is already included in these values and results are therefore almost optimal.

This research on performance-optimized feature ordering was performed within the context of a content-based image retrieval system for coats of arms. Our test database consists of only 444 images and the query engine uses 15 features. A more thorough investigation and evaluation of the improvement our approach provides would definitely require tests with considerably larger image databases. This will be one of the tasks we plan for the future.

However, we are very optimistic that our approach would result in similar improvements—even for a considerably larger database: First, 15 features—divided into two logical groups—should be sufficient for judging the performance of query models with a maximum of ten layers. Queries in most current retrieval systems and most of the queries we issue manually do not consist of such a high number of features. Second, if we had more images in the database, we would observe two major consequences:

1. The elements of the order distribution would have a higher variance and therefore the output of the prediction model would allow for easier distinction of different features.
2. The clustering of feature vectors elements would be more precise if more example images were considered. However, since coats of arms are images of a very specific nature we consider it unlikely that a larger database would result in a very different clustering. The error of the relationship model introduced by a small test base should therefore be rather small.

5 CONCLUSION

We presented an approach to improve the performance in content-based retrieval systems. The idea is based on the concept of a query model, an ordered list of feature extraction functions associated with distance function, threshold and weight. The approach further employs two types of features: one for quick elimination and one for detailed investigation of features and two models for the prediction of the size of the result set and for the specification of feature relationships. An optimization algorithm uses these inputs to produce a performance-optimal ordering of features in a query model.

All algorithms are implemented in a C/C++ library in our test environment on a Linux workstation. The environment consists of QBIC, SOM-PAK, 15 feature classes and our coats of arms test database. Results for performance improvement are quite promising: the average time consumed by a query is reduced by 66% and close to the optimum.

References

[1] Breiteneder, C., Eidenberger, H., "A Retrieval System for Coats of Arms", International Symposium on Intelligent Multimedia and Distance Education, Baden-Baden, 1999.

[2] Breiteneder, C., Eidenberger, H., "Content-based Image Retrieval of Coats of Arms", Proc. of the 1999 International Workshop on Multimedia Signal Processing, Helsingör, 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] Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., SOM-PAK: The Self-organizing Map Program Package, Helsinki, 1995.