Efficient access methods for content-based image retrieval with inverted files

Henning Müller, David McG. Squire, Wolfgang Müller and Thierry Pun

Computer Vision Group, University of Geneva

24 Rue du Général Dufour, CH-1211 Genève 4, Switzerland

ABSTRACT

As human factor studies over the last thirty years have shown, response time is a very important factor for the usability of an interactive system, especially on the world wide web. In particular, response times of under one second are often specified as a usability requirement.[1] This paper compares several methods for improving the evaluation time in a content-based image retrieval system (CBIRS) which uses inverted file technology. The use of the inverted file technology facilitates search pruning in a variety of ways, as is shown in this paper. For large databases (> 2000 images) and a high number of possible features (> 80000), efficient and fast access is necessary to allow interactive querying and browsing. Parallel access to the inverted file can reduce the response time. This parallel access is very easy to implement with little communication overhead, and thus scales well. Other search pruning methods, similar to methods used in information retrieval, can also reduce the response time significantly without reducing the performance of the system. The performance of the system is evaluated using precision vs. recall graphs, which are an established evaluation method in information retrieval. A user survey was carried out in order to obtain relevance judgments for the queries reported in this work.

Keywords: inverted file, content-based image retrieval, efficient access, search pruning, speed evaluation

1. INTRODUCTION

The response time of an interactive computer system is great importance to user satisfaction. Despite this, the issue is not frequently explicitly addressed in CBIR: many authors mention its importance, but only a few quote actual times.[2] The classic advice on response times states that 0.1 second is about the limit for the user to feel that the system is reacting instantaneously, 1.0 second is about the limit for the user's flow of thought to stay uninterrupted, and 10 seconds is about the limit for keeping the user's attention focused on the dialogue.[3] A CBIRS should attempt to stay below the 1 second limit for any query, although in a distributed system, such as a web-based CBIRS, network bandwidth can be the limiting factor.

The goal should be a suitable trade-off between response time and result quality. Studies of the behaviour of journalists performing searches with a digital photo archive have indicated that browsing is an essential search strategy, and that response time was a key issue " ... journalists did not seem to have much time to find 'the best' photo. Rather they tended to make acceptable selections.".[4,5]

Relevance feedback has been shown to be extremely useful in text retrieval applications,[6] and it is now being applied in some CBIRSs.[7–9] By augmenting the query with features from relevant retrieved images, a query can be produced which better represents the user's information need. Since human perception of image similarity is both subjective and task-dependent,[10–12] we believe relevance feedback to be an essential component of a CBIRS.

Performance evaluation is a difficult problem in CBIR, largely due to the subjectivity and task-dependence issues mentioned above. For these reasons, we believe that evaluation *must* involve experiments with *several* real users. Examples of such studies exist for medical CBIRSs,[13] but much published work contains little or no quantitative performance evaluation. The CBIR community still lacks a commonly accepted database of images, queries and relevance judgments, such as the TREC databases used in text retrieval.

Henning.Mueller@cui.unige.ch, David.Squire@cui.unige.ch, Wolfgang.Mueller@cui.unige.ch, Thierry.Pun@cui.unige.ch
tel. ++41 22 705 7633, fax. ++41 22 705 7780

The evaluation of retrieval performance has been thoroughly studied in the text retrieval community.[14] One of the most common measures, the *Precision* vs. *Recall* (PR) graph,[14,15] is now becoming more widely used in CBIR.[16,9,17] In this paper, performance results are presented in the form of PR graphs averaged over several users and several queries. It is important to note that different users frequently specify differing sets of images as relevant for a given query image.

In this paper we describe several methods for reducing the response time of a CBIRS. These methods are based either on search pruning or the parallel evaluation of a query. It is shown that the use of an inverted file data structure facilitates both these techniques. The search pruning methods presented here are "lossy", meaning that the final response to the query is not guaranteed to be identical to that of the unpruned search. Results are presented comparing the performance of these time-limited query evaluations with those for which there was no search pruning, showing that fast response can be achieved without degrading system performance.

2. RELATED WORK

2.1. Image retrieval systems

The aim of a CBIRS is to retrieve images from a database based on their similarity to a query image or sketch. In recent years many different image retrieval systems have emerged, focusing on different aspects of image retrieval or different kinds of databases. Since the general object recognition problem remains unsolved, such systems are either restricted to limited domains, such as the detection of cars[18] or buildings,[19] or use some function of low-level image features to attempt to capture a general measure of image similarity.

Such systems use example images, region(s)[20] or predefined blobs[21] as the query. The features extracted are usually represented as a fixed length feature vector, and some distance metric or probabilistic model is applied to find the most "similar" images in the collection. The features usually employed by such systems fall into the classes of colour, texture and shape. Colour histogram intersection[22,23] and texture features based on Gabor filters[24,9] or wavelets[25] are common.

Since search in high dimensional spaces is very computationally expensive, most systems try to find a compact features representation (the "best" features) in an effort to provide acceptable system response time. Techniques such as factor analysis[26] or self-organizing maps[27] have been used to reduce feature space dimensionality. A few systems try to find the best features for each query: Dy *et al.*[28] use the fact that different features need to be used to distinguish images from different classes than to distinguish between subclasses of one class in a database of medical images.

Another method for improving response time is the clustering of images into groups containing similar images.[24,29] Although it can reduce response time, clustering can exclude large numbers of images from the querying/browsing process which is very important. Moreover, since image similarity is not a fixed notion, any fixed boundaries in a system are potentially dangerous.

2.2. Text retrieval systems

Compared to CBIR, text retrieval is a mature field. Originally driven by libraries and the needs of the legal communities in countries with precedent-based legal systems, text retrieval is now an essential component of world wide web search engines such as Yahoo and AltaVista. Text retrieval researchers have been investigating techniques for indexing millions of documents using thousands of features (words) for more than thirty years.[30,14,31] Despite this long experience, most image retrieval researchers do not seek to use this great body of knowledge to improve their systems. Text and images have been seen to be too different to be accessed in the same way.

Inverted files are the most common data structure used in text retrieval. An inverted file contains an entry for every possible feature (word) which consists of a list of the images (documents) which contain that feature, the frequency of occurrence of that feature in the collection, and possibly the frequency of that feature in each image. The text retrieval community has developed techniques for building and searching inverted files very efficiently.[31] The key realization is that in such systems both queries and stored objects are sparse: they have only a small subset of all possible features. Search is thus restricted to the subspace spanned by the query.

Whilst it is true that the low-level features used in most CBIRSs are at a very different semantic level from the words in texts, the same retrieval methods can be used if the features are suitably sparse. Witten *et al.* propose several methods for search pruning in an inverted file-based database.[31] In this work we apply some of these methods, adapted to the requirements of image retrieval.

3. THE *Viper* SYSTEM

The *Viper* system, inspired by text retrieval systems, uses a very large number of simple features.* The present version employs both local and global image colour and spatial frequency features, extracted at several scales, and their frequency statistics in both images and the whole collection. The intention is to make available to the system low-level features which correspond (roughly) to those present in the human vision system.

More than 80000 features are available to the system. Each image has $\mathcal{O}(10^3)$ such features, the mapping from features to images being stored in an inverted file. The use of such a data structure, in conjunction with the feature weighting scheme discussed below, means that the integration of text annotations is completely natural: textual features can be treated in exactly the same way as visual ones.

3.1. Features

3.1.1. Colour features

Viper uses a palette of 166 colours, derived by quantizing *HSV* space into 18 hues, 3 saturations, 3 values and 4 grey levels. Two sets of features are extracted from the quantized image. The first is a colour histogram, with empty bins are discarded. The second represents colour layout. Each block in the image (the first being the image itself) is recursively divided into four equal-sized blocks, at four scales. The occurrence of a block with a given mode color is treated as a binary feature. There are thus 56440 possible colour block features, of which each image has 340.

3.1.2. Texture features

Gabor filters have been applied to texture classification and segmentation, as well as more general vision tasks.[24,32] We employ a bank of real, circularly symmetric Gabors, defined by

$$f_{mn}(x,y) = \frac{1}{2\pi\sigma_m^2} e^{-\frac{x^2+y^2}{2\sigma_m^2}} \cos(2\pi(u_{0_m} x \cos\theta_n + u_{0_m} y \sin\theta_n)), \tag{1}$$

where m indexes filter scales, n their orientations, and u_{0_m} gives the centre frequency. The half peak radial bandwidth is chosen to be one octave, which determines σ_m. The highest centre frequency is chosen as $u_{0_1} = 0.5$, and $u_{0_{m+1}} = u_{0_m}/2$. Three scales are used. The four orientations are: $\theta_0 = 0$, $\theta_{n+1} = \theta_n + \pi/4$. The resultant bank of 12 filters gives good coverage of the frequency domain, and little overlap between filters. The mean energy of each filter is computed for each of the smallest blocks in the image. This is quantized into 10 bands. A feature is stored for each filter with energy greater than the lowest band. Of the 27648 such possible features, each image has at most 3072. Histograms of the mean filter outputs are used to represent global texture characteristics.

3.1.3. Feature weighting and relevance feedback

As discussed above, relevance feedback can produce a query which better represents a user's information need. *Viper* uses relevance feedback, in conjunction with feature weighting schemes inspired by those used in text retrieval.[33] Some modifications were necessary since the image features used can not always be treated in the same way as words in documents. The weighting function can depend upon the *term frequency* tf_j (frequency of feature j in the image) and *collection frequency* cf_j (frequency of the feature j in the entire database) of the feature, as well as its type (block or histogram). The motivation for using term frequency and collection frequency is very simple: features with high *tf* characterize an image well; features with high *cf* do not distinguish that image well from others.[33] We consider a query q containing N images i with relevances $R_i \in [-1, 1]$. The frequency of feature j in the pseudo-image corresponding to q is

$$tf_{qj} = \frac{1}{N} \sum_{i=1}^{N} tf_{ij} \cdot R_i. \tag{2}$$

*Visual Information Processing for Enhanced Retrieval. Web page: `http://cuiwww.unige.ch/~viper/`

Viper weighting functions make use of a base weight

$$wf^0_{kqj} = \begin{cases} tf_{qj} & \text{for block features} \\ \operatorname{sgn}(tf_{qj}) \cdot \min\left\{\operatorname{abs}\left(tf_{qj}\right), tf_{kj}\right\} & \text{for histogram features} \end{cases}. \tag{3}$$

(The second case is a generalized histogram intersection.) Two different logarithmic factors are used, which depend upon cf:

$$lcf_{1j} = \begin{cases} \log(\frac{1}{cf_j}) & \text{for block features} \\ 1 & \text{hist.} \end{cases} \qquad lcf_{2j} = \begin{cases} \log(\frac{1}{cf_j} - 1 + \epsilon) & \text{for block features} \\ 1 & \text{hist.} \end{cases}. \tag{4}$$

ϵ is added to avoid overflows. We investigated a variety of weighting functions in an earlier study.[34] The weighting function used for the experiments reported in the present work is:

$$\text{classical idf:} \quad wf^2 = wf^0_{kqj} \cdot \left(lcf_{1j}\right)^2. \tag{5}$$

$$\tag{6}$$

For each image k, using weighting method l, a score s^l_{kq} is calculated:

$$s^l_{kq} = \sum_j wf^l_{kqj}. \tag{7}$$

Scores for all images containing features present in the query are maintained in a scoreboard.

Unlike the text retrieval case, we have several groups of features of different kinds: local colour, global colour, local texture and global texture. These groups have very different numbers of features and very different distributions of term and collection and frequencies. In this work we investigate the separate evaluation and normalization of each group before the calculation of a final score. This allows the influence of groups with differing numbers of features to be balanced (*e.g.* in any image there are many more features describing local texture than global colour).

3.2. Retrieval performance evaluation

To measure the performance of our system we used a database of 2500 heterogeneous real-world colour images provided by Télévision Suisse Romande. These images are extremely diverse and there are few groups of very visually similar images. Three users searched the entire database manually for images relevant to 14 query images. The queries were nearly all best described at the semantic level, by phrases such as “soccer crowd” or “hospital scene”. Thus it is very hard for a system with only low level features to give good query results. Some example query images can be seen in Figure 1.

(a) soccer crowd (b) child with gun (c) laboratory (d) hospital scene

Figure 1. Example query images

The sets of relevant images chosen by each user differed greatly, both in the number of images selected and also in the actual images. The number of images in the sets varied between three and 36, with an average of 19. Even for the same query image, one user selected four relevant images and another 36.

In order to demonstrate the great importance of relevance feedback, for each user all the relevant images in the first 20 images returned for the initial query were used as positive feedback for a second query. Negative feedback was not used in this study. This automatized relevance feedback is certainly not optimal: a trained user can do much better with using both positive and negative feedback.

PR graphs were then calculated, average over queries and users, in order to show the retrieval performance both before and after relevance feedback.

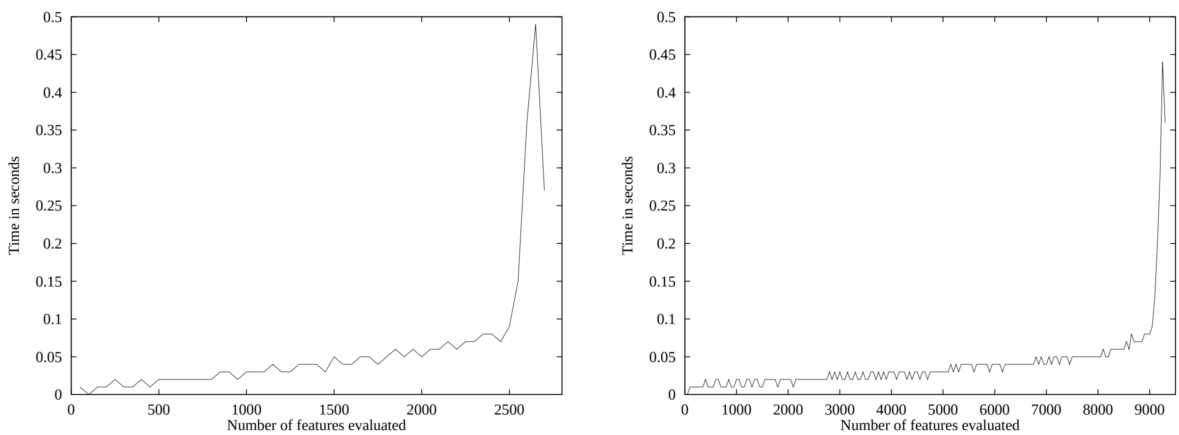
4. RESPONSE TIME EVALUATION

The evaluation times reported in this work are measured on a Sun Ultra 10 with 256 MB of memory and an 8 GB hard disk. The inverted file is read from the disk every time a query is evaluated; the other information is stored in memory. Evaluation times were measured using the C internal function for time measurements, which has a resolution of 0.01 seconds and measures elapsed processor time. As a second time measurement we used the C clock function, which returns the time of day with millisecond resolution. Elapsed time measure in this way varies greatly, since it is influenced by other processes running on the machine and network traffic, which we could not control. For non-parallel queries these times were essentially the same. For parallel query evaluation, they varied much more widely, since inter-processor communication is strongly influenced by the network traffic.

The time taken by *Viper* to evaluate a query depends strongly on the number of features in the query. Individual images have a widely varying numbers of features, usually between 500 and 3000 in our database. The number of features in a query increases with the number of input images, albeit sublinearly. Multiple image queries might have up to 10000 or more features.

4.1. Evaluation times for individual features

The time taken to evaluate a given feature depends strongly on the collection frequency of this feature, since this determines the number of calculations and disk accesses required to update the scoreboard. Figure 2 shows the the evaluation times for groups of 50 features for both a single and a multiple image query. For these queries, the evaluation of 50 features takes between 0.01 and 0.49 seconds. Very similar results are obtained for other queries. Features are evaluated in order sorted by weight (see Equation 5). It is clear that the final features to be evaluated are very costly (*i.e.* those with low weights and thus high collection frequencies).

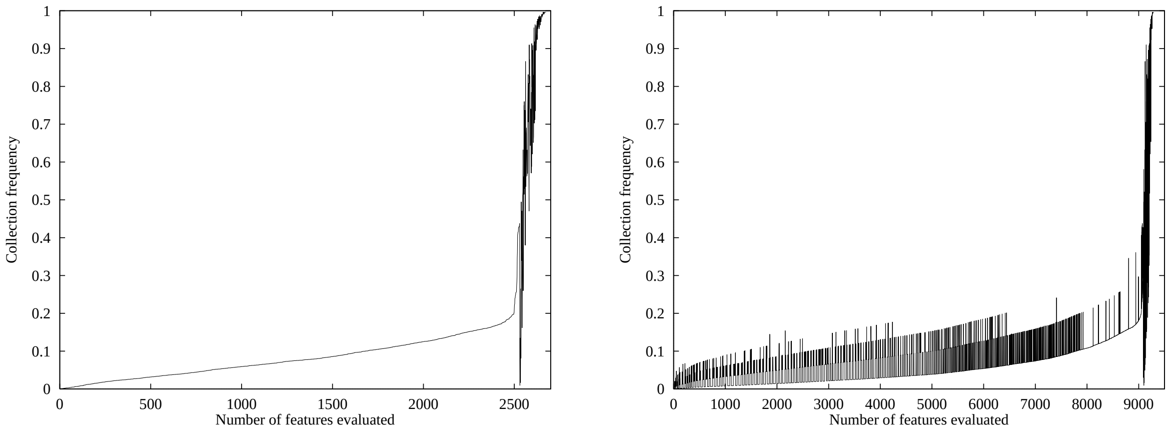


(a) single image query

(b) multiple image query

Figure 2. Evaluation time for 50 features for a query with 2674 features and a feedback query with 9283 features

Considering Figure 3, it is clear that the time taken for feature evaluation is strongly correlated with collection frequency. More frequent features need more disk accesses and calculations because they occur in a greater numbers of images. Moreover, these computationally expensive features generally receive lower weightings than the less frequent features, and thus they contribute less to the final image scores.

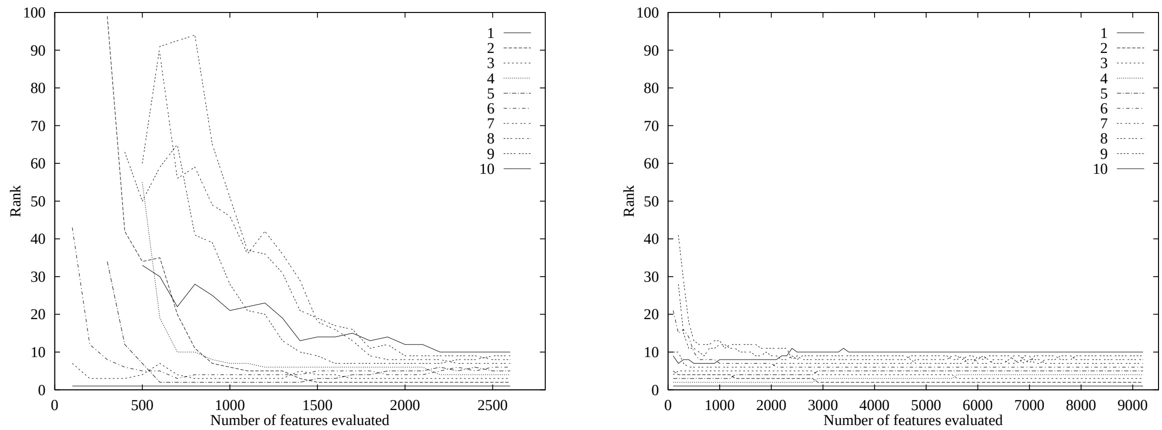


(a) single image query

(b) multiple image query

Figure 3. Collection frequencies of the features for a single and a multiple image query

Since it is clear that the last features evaluated take most of the time, it is interesting to know how much these features actually influence the final result of the query. In order to study this, we tracked the ranks of the images known as the final top ten during the evaluation of the query. This was done for a single image query with 2673 features and a multiple image query with 9283 features. The results can be seen in Figure 4.



(a) initial query

(b) with feedback

Figure 4. Development of the ranking of the final top ten images

The order of the final top ten becomes relatively clear even at quite an early stage. This is especially so in the multiple image query. In the single image query, all the top ten images are already in the top 40 after 50% of the features have been evaluated. For the multiple image query this is the case after only 10% of the features have been evaluated.

5. METHODS FOR REDUCING QUERY EVALUATION TIME

Methods for reducing query evaluation time fall into two general classes. Evaluation can be stopped when it is known that the top n ranked images can no longer change (a "lossless" method), or one can decide to prune the search while tolerating some (small) changes in the final image ranking (a "lossy" method). When choosing between these classes, it is important to know whether different results are necessarily worse results, and if so, to what extent.

Earlier experiments indicated that lossless pruning did not give a great gain in speed, so here we concentrate on lossy methods. The performances of the pruned evaluation methods are compared with that of the unpruned system, in order to assess the impact on response accuracy.

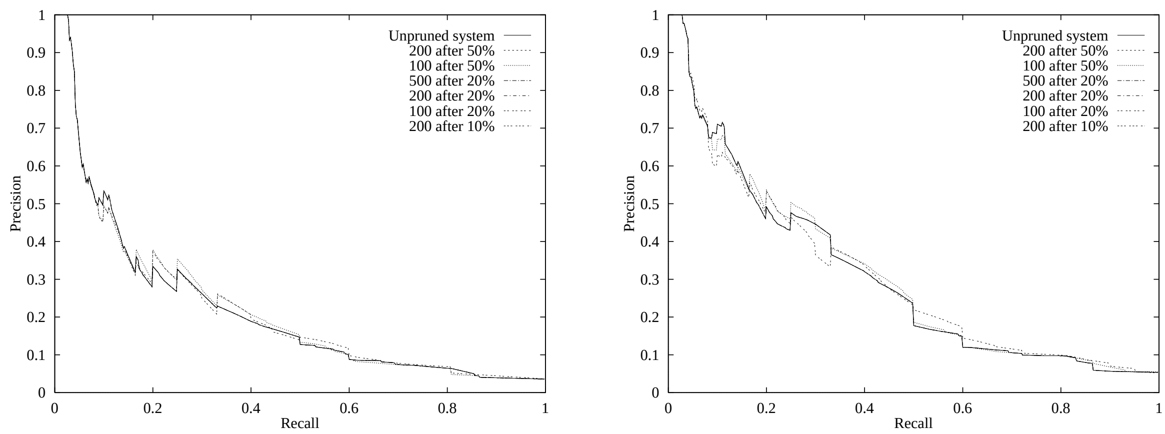
The only lossless method reported in this paper is basic parallel access to the features. The inverted file facilitates parallel access to features as there is no need for writing accesses or synchronization. It even allows us to have a number of small inverted files, each containing the features of one feature group. Thus resources can be used more efficiently.

5.1. Reducing the number of images evaluated

This pruning method is based on the fact that images which have a low score after the most highly weighted features have been evaluated are not very likely to be highly ranked after all features have been evaluated. The update of the scores of such images can thus be stopped early. Disk access is not reduced by this method, but calculation time is. In order to evaluate the impact of this method, the number of images retained in the scoreboard was reduced to to 100, 200 and 500. These reductions were done after 10, 20 and 50% of the features had been evaluated.

In order to be able to compare the the performance of the pruned searches with that of the non-pruned version, the scores for all images must be calculated, and the resulting list then reduced to the same number as the pruned versions. The PR graphs are continued using the expected precision for randomly distributed responses after this point. Here all lists were reduced to 100 images before comparison.

Figure 5 shows that in most cases the pruned versions do not perform worse than the unpruned system.



(a) initial feedback

(b) with feedback

Figure 5. PR graphs for scoreboard pruning

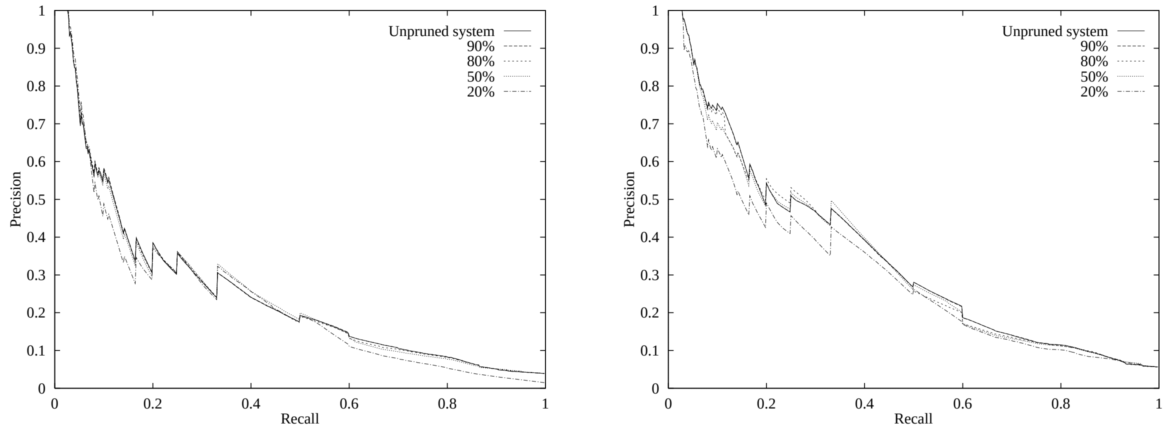
Table 1 shows that time savings of up to 30% can be achieved, with very little loss in accuracy. As would be expected from Figure 2, the evaluation time is dependent on the percentage of the features which are evaluated before the cut-off as well as on the number of images to which the list is reduced. We would suggest that a reduction of the scoreboard to 100 images after 20% of the features have been evaluated as a suitable trade-off between speed and accuracy.

No. images retained	Cutoff point	Average evaluation time
100	20%	1.38s
100	50%	1.45s
200	10%	1.41s
200	20%	1.43s
200	50%	1.47s
500	20%	1.50s
unpruned		2.02s

Table 1. Evaluation times for scoreboard pruning.

5.2. Reducing the number of features evaluated

Figure 4 shows that the final top ten images were in the top ten quite early in the query evaluation. This suggests that query evaluation could be stopped after a certain percentage of features without seriously affecting performance. Since the low-weighted features are very computationally expensive, we can expect a better than linear reduction in computation time. To evaluate this method, performance was evaluated for cutoff points of 20, 50, 80 and 90 percent.



(a) initial query

(b) with feedback

Figure 6. PR graphs for cutoff after various percentages of features evaluated

Figure 6 shows that only the curve for cutoff after 20% percent of features is notably worse that the unpruned case. The other cutoffs even give better performance in some parts of the PR graph, although not significantly. This could mean though that some of the very frequent features are essentially noise.

Cutoff point	Average evaluation time
20%	0.09s
50%	0.28s
80%	0.72s
90%	0.96s
unpruned	2.02s

Table 2. Evaluation times for cutoff after various percentages of features evaluated

Table 2 shows the great improvements provided by this pruning technique. By reducing the number of features evaluated by only 10% the evaluation time is more than halved. Evaluating 20% of the features takes less than 5%

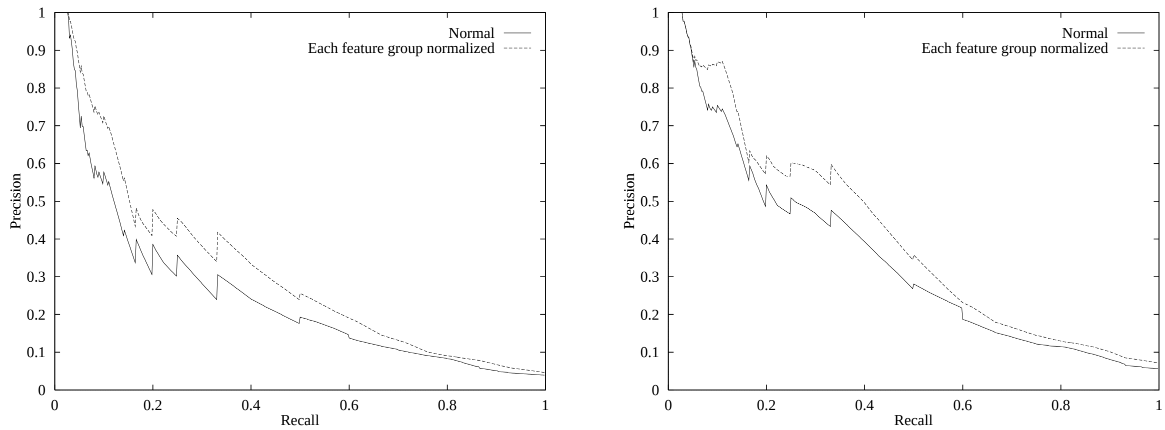
of the full evaluation time, but the performance is significantly worse than that of full evaluation. This mode still might be useful if an extremely quick response is needed, or if the the user just wants to browse the images – some noise may perhaps even be desirable.

5.3. Parallel access to features

The parallelization of the inverted file access is very easy and there is no cost for synchronization. Consequently, large improvements could be expected. On the other hand, in the absence of a dedicated cluster, parallelization makes us dependent on network traffic, which we cannot control. We used PVM (Parallel Virtual Machine) as a parallelization library.

The features for each of the four feature groups were evaluated by processes running on separate machines. This allowed us to change the weighting scheme so that each feature group was separately normalized (see §3.1.3).

Figure 7 shows that this improves retrieval performance in our experiment significantly. A non-parallel version of the separate feature normalization was implemented so that results could be better compared.



(a) initial query feedback

(b) with feedback

Figure 7. PR graphs for the standard version and for separately normalized feature groups

	processor time	real time
normal feature access	2.02s	2.03s
separately normalized	2.86s	3.77s
parallel version	0.59s	11.51s

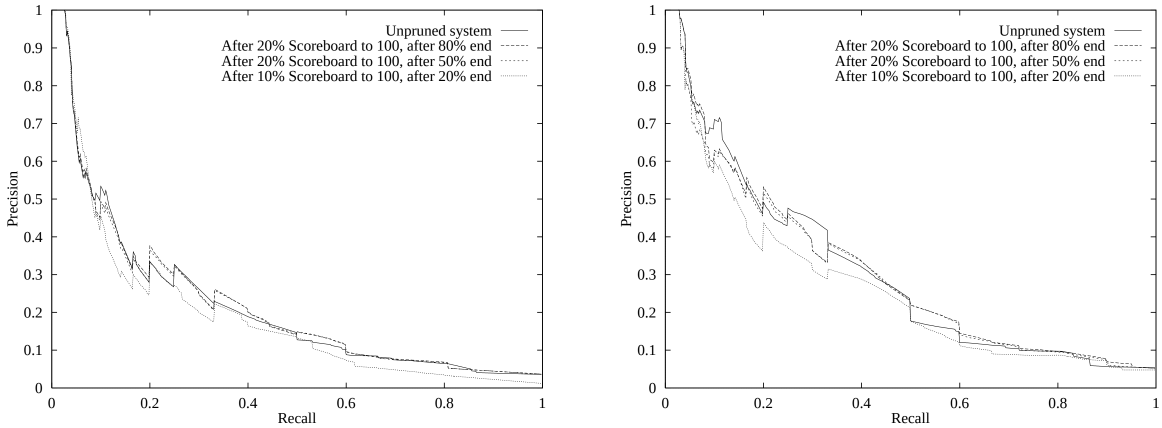
Table 3. Comparison of evaluation times for parallel and non-parallel feature evaluation

Table 3 shows that the processor time is significantly reduced in the parallel version, though network traffic destroyed this advantage. This suggests that more attention must be paid to inter-process communication.

5.4. Combining methods

The easiest combination of pruning methods to implement is that of scoreboard and feature pruning. Figure 8 shows that only the curve for cutoff after 20% of features is significantly worse than that for the evaluation of all features.

The differences in performance between the combination of scoreboard and feature pruning and feature pruning alone are thus very small whereas, as shown in Table 4, the gain in speed is between 10 and 30 percent, which is significant.



(a) initial query

(b) with feedback

Figure 8. PR graphs for various combinations of scoreboard and feature pruning

Pruning method	Average evaluation time
scoreboard to 100 after 20%, cutoff after 50%	0.23s
cutoff after 50%	0.28s
scoreboard to 100 after 20%, cutoff after 80%	0.52s
cutoff after 80%	0.72s
scoreboard to 100 after 10%, cutoff after 20%	0.08s
cutoff after 20%	0.09s

Table 4. Comparison of evaluation times for parallel and non-parallel feature evaluation

The combination of the parallel access methods with the pruning methods could offer further improvements. Each feature group could be pruned separately. This great potential, since the different feature groups have very different collection and document frequency characteristics and thus should be pruned in different ways.

Scoreboard reduction is especially important for the parallel version because it shortens the lists which need to be transmitted over the network, making it less dependent on network traffic. Future experiments will be performed in order to discover the best ways of combining all the methods.

6. CONCLUSIONS

This paper addresses an issue which we believe is often ignored in CBIRS studies: the need to investigate ways of reducing response times for queries. We propose to this end three strategies which are both fairly simple to implement and effective, where effectiveness means low response time with simultaneously good retrieval performance as measured by precision/recall curves. All experiments have been conducted using the *Viper* system, a CBIRS based on inverted files and relevance feedback techniques. The image database used contains 2500 pictures, each of them being described by at most a few thousand features from a set of about 80’000.

The three strategies for decreasing response time are: 1) reducing the number of images evaluated, 2) reducing the number of features evaluated, 3) parallel access to features. For a given retrieval quality, the first strategy provides time savings of up to 30%, which is already a noticeable improvement. The second strategy may yield much higher decreases in response time, down to about 5% of the original one. This might, however, lead to a decrease in retrieval accuracy; a tradeoff between speed and accuracy has to be chosen by the user depending on his needs. The third strategy, parallel access to features, allows the search load to be balanced on different computers, and to have smaller inverted files to access. It offers perhaps the largest potential, especially when the number of features and feature groups gets bigger. Finally, we suggest how these three strategies can be combined in effective ways.

ACKNOWLEDGEMENTS

This work was supported by the Swiss National Foundation for Scientific Research (grant no. 2000-052426.97).

REFERENCES

1. J. Nielsen, "The need for speed." Alertbox (web page: `http://www.useit.com/alertbox/9703a.html`), March 1997.
2. A. P. Berman and L. G. Shapiro, "Efficient content-based retrieval: Experimental results," in *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, pp. 55–61, (Fort Collins, Colorado, USA), 22 June 1999.
3. J. Nielsen, *Usability Engineering*, Academic Press, Boston, MA, 1993.
4. M. Markkula and E. Sormunen, "Searching for photos - journalists' practices in pictorial IR," in *The Challenge of Image Retrieval, A Workshop and Symposium on Image Retrieval*, J. P. Eakins, D. J. Harper, and J. Jose, eds., Electronic Workshops in Computing, The British Computer Society, (Newcastle upon Tyne), 5–6 February 1998.
5. M. Markkula and E. Sormunen, "End-user searching challenges indexing practices in the digital photo archive," *Information Retrieval* , 1999. (to appear).
6. G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American Society for Information Science* **41**(4), pp. 288–287, 1990.
7. M. E. Wood, N. W. Campbell, and B. T. Thomas, "Iterative refinement by relevance feedback in content-based digital image retrieval," in *Proceedings of The Fifth ACM International Multimedia Conference (ACM Multimedia 98)*, pp. 13–20, (Bristol, UK), September 1998.
8. Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: A power tool in interactive content-based image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology* **8**, pp. 644–655, September 1998. (Special Issue on Segmentation, Description, and Retrieval of Video Content).
9. D. M. Squire, W. Müller, H. Müller, and J. Raki, "Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback," in *The 10th Scandinavian Conference on Image Analysis (SCIA'99)*, (Kangerlussuaq, Greenland), June 7–11 1999.
10. F. Mokhtarian, S. Abbasi, and J. Kittler, "Efficient and robust retrieval by shape content through curvature scale space," in *Image Databases and Multi-Media Search*, A. W. M. Smeulders and R. Jain, eds., pp. 35–42, Intelligent Sensory Information Systems, Faculty of Mathematics, Computer Science, Physics and Astronomy, Amsterdam University Press, (Kruislaan 403, 1098 SJ Amsterdam, The Netherlands), August 1996.
11. D. M. Squire and T. Pun, "A comparison of human and machine assessments of image similarity for the organization of image databases," in *The 10th Scandinavian Conference on Image Analysis (SCIA'97)*, M. Frydrych, J. Parkkinen, and A. Visa, eds., pp. 51–58, Pattern Recognition Society of Finland, (Lappeenranta, Finland), June 1997.
12. Y. H. Kim, K. E. Lee, K. S. Choi, J. H. Yoo, P. K. Rhee, and Y. C. Park, "Personalized image retrieval with user's preference model," in *Multimedia Storage and Archiving Systems III (VV02)*, C.-C. J. Kuo, S.-F. Chang, and S. Panchanathan, eds., vol. 3527 of *SPIE Proceedings*, pp. 47–55, (Boston, Massachusetts, USA), November 1998.
13. C.-R. Shyu, A. Kak, C. Brodley, and L. S. Broderick, "Testing for human perceptual categories in a physician-in-the-loop CBIR system for medical imagery," in *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, pp. 102–108, (Fort Collins, Colorado, USA), 22 June 1999.
14. G. Salton, "The state of retrieval system evaluation," *Information Processing and Management* **28**(4), pp. 441–450, 1992.
15. J. Tague-Sutcliffe, "The pragmatics of information retrieval experimentation, revisited," in *Readings in Information Retrieval*, K. Spark Jones and P. Willett, eds., Multimedia Information and Systems, ch. 4, pp. 205–216, Morgan Kaufmann, 340 Pine Street, San Francisco, USA, 1997.
16. J. R. Smith and S.-F. Chang, "VisualSEEk: a fully automated content-based image query system," in *The Fourth ACM International Multimedia Conference and Exhibition*, (Boston, MA, USA), November 1996.

17. R. Srihari, Z. Zhang, and A. Rao, "Image background search: Combining object detection into content-based similarity image retrieval (CBIR) systems," in *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, pp. 97–101, (Fort Collins, Colorado, USA), 22 June 1999.
18. B. Ozer, W. Wolf, and A. N. Akansu, "A graph based object description for information retrieval in digital image and video libraries," in *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, pp. 79–83, (Fort Collins, Colorado, USA), 22 June 1999.
19. Q. Iqbal and J. K. Aggarwal, "Applying perceptual grouping to content-based image retrieval: Building images," in *Proceedings of the 1999 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 42–48, IEEE Computer Society, (Fort Collins, Colorado, USA), June 23–25 1999.
20. B. Moghaddam, H. Biermann, and D. Margaritis, "Defining image content with multiple regions of interest," in *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, pp. 89–93, (Fort Collins, Colorado, USA), 22 June 1999.
21. S. Belongie, C. Carson, H. Greenspan, and J. Malik, "Color- and texture-based image segmentation using EM and its application to content-based image retrieval," in *Proceedings of the International Conference on Computer Vision (ICCV'98)*, (Bombay, India), January 1998.
22. A. Vellaikal and C.-C. J. Kuo, "Content-based image retrieval using multiresolution histogram representation," in *Digital Image Storage and Archiving Systems*, C.-C. J. Kuo, ed., vol. 2606 of *SPIE Proceedings*, pp. 312–323, (Philadelphia, PA, USA), October 1995.
23. A. Gupta and R. Jain, "Visual information retrieval," *Communications of the ACM* **40**, pp. 70–79, May 1997.
24. W. Ma and B. Manjunath, "Texture features and learning similarity," in *Proceedings of the 1996 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 425–430, (San Francisco, California), June 1996.
25. R. Zarita and S. Lelandais, "Wavelets and high order statistics for texture classification," in *The 10th Scandinavian Conference on Image Analysis (SCIA'97)*, M. Frydrych, J. Parkkinen, and A. Visa, eds., pp. 95–102, Pattern Recognition Society of Finland, (Lappeenranta, Finland), June 1997.
26. T. Pun and D. M. Squire, "Statistical structuring of pictorial databases for content-based image retrieval systems," *Pattern Recognition Letters* **17**, pp. 1299–1310, 1996.
27. K. Han and S.-H. Myaeng, "Image organization and retrieval with automatically constructed feature vectors," in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, eds., pp. 157–165, (Zürich, Switzerland), August 1996.
28. J. G. Dy, C. E. Brodley, A. Kak, C.-R. Shyu, and L. S. Broderick, "The customized-queries approach to CBIR using using EM," in *Proceedings of the 1999 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 400–406, IEEE Computer Society, (Fort Collins, Colorado, USA), June 23–25 1999.
29. A. Vellaikal and C.-C. J. Kuo, "Hierarchical clustering techniques for image database organization and summarization," in *Multimedia Storage and Archiving Systems III (VV02)*, C.-C. J. Kuo, S.-F. Chang, and S. Panchanathan, eds., vol. 3527 of *SPIE Proceedings*, pp. 68–79, (Boston, Massachusetts, USA), November 1998.
30. K. Sparck-Jones, "The Cranfield tests," in *Information Retrieval Experiment*, K. Sparck-Jones, ed., pp. 256–284, Butterworth-Heinemann, Oxford, U.K., October 1981.
31. I. H. Witten, A. Moffat, and T. C. Bell, *Managing gigabytes: compressing and indexing documents and images*, Van Nostrand Reinhold, 115 Fifth Avenue, New York, NY 10003, USA, 1994.
32. A. Jain and G. Healey, "A multiscale representation including opponent color features for texture recognition," *IEEE Transactions on Image Processing* **7**, pp. 124–128, January 1998.
33. G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing and Management* **24**(5), pp. 513–523, 1988.
34. D. M. Squire, W. Müller, and H. Müller, "Relevance feedback and term weighting schemes for content-based image retrieval," in *Third International Conference On Visual Information Systems (VISUAL'99)*, D. P. Huijsmans and A. W. M. Smeulders, eds., no. 1614 in Lecture Notes in Computer Science, pp. 549–556, Springer-Verlag, (Amsterdam, The Netherlands), June 2–4 1999.