UNIVERSITE DE GENEVE

CENTRE UNIVERSITAIRE
D'INFORMATIQUE
GROUPE VISION

Date: February 8, 1999
N° 99.01

TECHNICAL REPORT

VISION

Improving Response Time by Search Pruning in a Content-Based Image Retrieval System, Using Inverted File Techniques

David McG. Squire Henning Müller Wolfgang Müller

Computer Vision Group
Computing Science Center, University of Geneva
24 rue du Général Dufour, CH - 1211 Geneva 4 SWITZERLAND
e-mail: David.Squire@cui.unige.ch, Henning.Mueller@cui.unige.ch,
Wolfgang.Mueller@cui.unige.ch

Abstract

This paper describes several methods for improving query evaluation speed in a content-based image retrieval system (CBIRS). Response time is an extremely important factor in determining the usefulness of any interactive system, as has been demonstrated by human factors studies over the past thirty years. In particular, response times of less than one second are often specified as a usability requirement. It is shown that the use of inverted files facilitates the reduction of query evaluation time without significantly reducing the accuracy of the response. The performance of the system is evaluated using precision *vs.* recall graphs, which are an established evaluation method in information retrieval (IR), and are beginning to be used by CBIR researchers.

KEYWORDS: content-based image retrieval, search pruning, inverted file, response time

1 Introduction

Response times in the interaction between computer systems and human users are of great importance to user satisfaction. At present, this fact is not widely explicitly addressed in CBIR: many authors discuss mechanisms for reducing search time, but few quote *actual times*. The goal should be a suitable trade-off between response time and the quality of the results. The classic advice concerning response times is reiterated by Nielsen [1]:

- 0.1 second is about the limit for having the user feel that the system is reacting instantaneously.
- 1.0 second is about the limit for the user's flow of thought to stay uninterrupted.
- 10 seconds is about the limit for keeping the user's attention focused on the dialogue.

Traditional human factors research has also shown the need for response times faster than one second [2]. A CBIRS should attempt to stay below this limit for any query, although in a distributed system, such as a web-based CBIRS, network bandwidth can be the limiting factor.

In this paper we describe several methods for reducing response time, and also for enforcing an upper bound. We compare the results of these time-limited query evaluations with those for which there was no search pruning. We use an image database and queries for which users have provided relevance judgments, described in [3]. Results are presented using precision *vs.* recall graphs.

2 Other systems

Most current CBIRSs represent images as points in a multidimensional feature space. Image similarity is defined as the Euclidean or Mahalanobis distance between points in this vector space. Many authors propose methods for reducing the time taken to search this space, such as dimensionality reduction using Principal Components Analysis (PCA) [4, 5], clustering [6, 7], or spatial search structures such as KD-trees [5, 8]. The suitability of PCA as preprocessing for information retrieval has been challenged: it can eliminate the "rare" feature variations which can be very useful for creating a specific query. The other techniques all limit search by pruning the number of images for which distances are calculated.

2.1 Inverted files

IR researchers, however, have more than 30 years of experience with the problem of reducing query evaluation time in text retrieval systems. They have generally taken a different approach, based on the *inverted file* (IF) data structure most commonly used in IR. An IF contains an entry for each feature consisting of a list of the items which have that feature. In general, an item has only a small subset of all possible features: similarity computation is restricted to the subspace spanned by the features present in the query.

Several IF-based search pruning methods are described in Witten *et al.* [9]. The principal difference is that in IF systems similarities are evaluated feature by feature, rather than item (document, image) by item. This search may be pruned in two basic ways:

- do not evaluate all features present in the query
- do not evaluate all features for all possible response documents

The choice of which features to evaluate depends on their importance, typically defined by their weights.

3 System structure

Viper employs more than 80000 simple colour and spatial frequency features, both local and global, extracted at several scales.[1] The intention is to make available to the system low-level features which correspond (roughly) to those present in the retina and early visual cortex.

The fundamental difference between traditional computer vision and image database applications is that there is a human "in the loop". Relevance feedback (RF) allows a simple classifier to be learnt "on the fly", corresponding to the user's information need. Unlike some other CBIRSs, we do not attempt to find the "right features" in advance.

3.1 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. This gives more tolerance to changes in saturation and value, which is desirable since these channels can be effected by lighting conditions and viewpoint [10]. 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.

3.2 Texture features

Two dimensional Gabor filters (Gabors) have been used to describe the orientation- and frequency-selective properties of neurons in the visual cortex [11]. *Viper* employs a bank of real, circularly symmetric Gabors, at three scales and four orientations. 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 quantized into 10 bands, for each of the smallest

[1] Visual Information Processing for Enhanced Retrieval:
`http://cuiwww.unige.ch/~viper/`

blocks in the image. A feature is stored for each filter with energy greater than the lowest band. Histograms of the mean filter outputs are used to represent global texture characteristics. Full details may be found in [3].

3.3 Feature weighting and relevance feedback

In an earlier study, the performances of several weighting schemes for CBIR were investigated, both with and without RF [12]. This study confirmed the efficacy of RF, and allowed the best-performing weighting method to be selected. This turned out to be a classic "$tf \cdot icf$" weight.[2] The weighting function depends on both the *term frequency* tf and the *collection frequency* cf of a feature, as well as its type (block or histogram). tf_j is the frequency with which feature j appears in an image. cf_j is the frequency with which images containing feature j occur in the entire image collection. The motivation for using tf and cf is very simple: features with high tf characterize an image well; features with high cf do not distinguish that image well from others [13].

We consider a query q containing N images i with relevances $R_i \in [-1, 1]$. Features from all N images are merged to form a "pseudo-image", with frequencies defined by

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

We now consider the evaluation of q for an image k in the collection. The term frequency component of the weighting function is

$$w_{tf_{kqj}} = \begin{cases} tf_{qj} & \text{block} \\ \operatorname{sgn}(tf_{qj}) \cdot \min\left\{ \left| tf_{qj} \right|, tf_{kj} \right\} & \text{histogram} \end{cases}. \tag{2}$$

(The histogram case is a generalized histogram intersection). The collection frequency component is

$$w_{cf_{kqj}} = \begin{cases} \log\left(cf_j^{-1} \right) & \text{block} \\ 1 & \text{histogram} \end{cases}. \tag{3}$$

The complete weighting function is

$$wf_{kqj} = w_{tf_{kqj}} \cdot w_{cf_{kqj}}^2. \tag{4}$$

4 Analysis of query evaluation time

The time taken by *Viper* to compute a response depends on the particular query or feedback image(s). Individual images have differing numbers of features (between ~ 1000 and ~ 2600 in our database), and the total number of features increases (sub-linearly) with the number of query or feedback images. The evaluation times reported in this paper were measured on a Sun Ultra 10 with 128 MB of memory and a 8 GB hard disk. The inverted file is read from the disk for each query. The other information is in the memory. The image database consists of 500 images for which relevance judgments from 5 users have been obtained for several queries. The images are taken from a set of 10000 images provided by Télévision Suisse Romande.

[2] term frequency · inverse collection frequency

The time taken to evaluate a feature depends on its collection frequency: those with high *cf* have correspondingly long lists of images for which similarity scores must be updated. Features are sorted according to their weights before evaluation. The evaluation time for 100 features typically varies between 0.01 and 0.30s, as shown in Figure 1. Similar distributions were observed for different queries and for different numbers of query images. It is clear that the evaluation of the features with high collection frequencies takes much longer.[3]

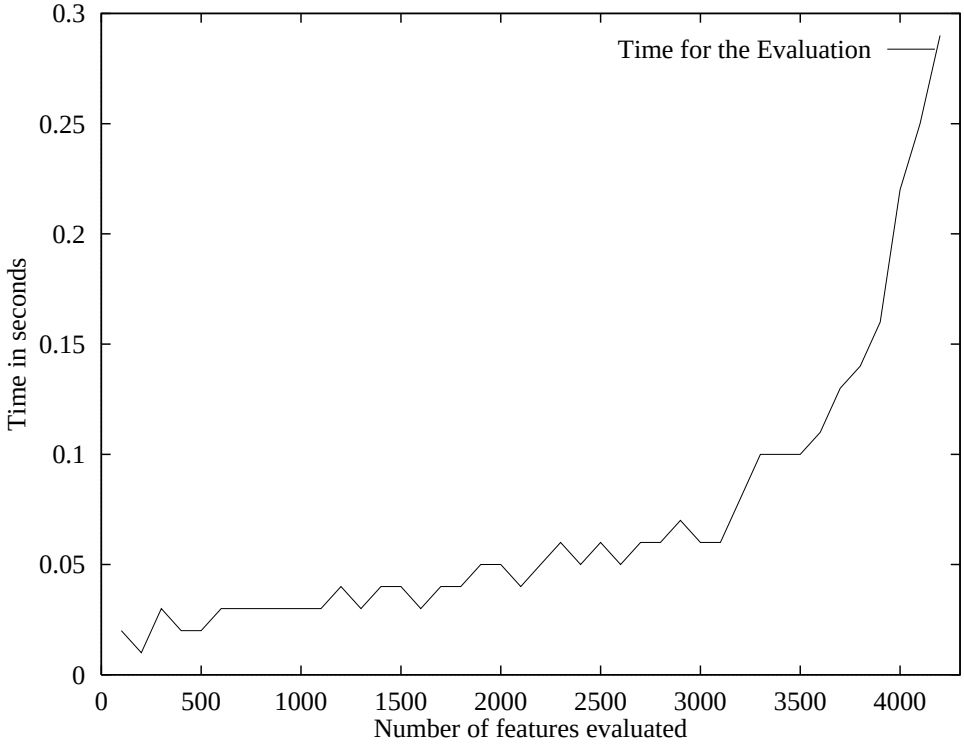

Figure 1: Evaluation times for each set of 100 features for a query image with 4224 features.

Typical response times when all features are evaluated are 0.85s for a single image query with 1667 features and 1.61s for a three image query with 4224 features. Since these response times are much longer than the "feeling of instantaneous reaction" goal discussed in §1, the query evaluation process was analyzed in more detail. The results appear in Table 1. It is clear that the operation for which there is the most scope for

number of features	1667	4224
creation of pseudo-image	0.01s	0.04s
calculation of normalizer	0.01s	0.03s
feature sorting	0.02s	0.03s
file access & score calculation	0.80s	1.50s
score sorting & normalization	0.01s	0.01s
total	0.85s	1.61s

Table 1: Breakdown of query evaluation times.

improvement is access to the inverted file and the calculation of the scores for the feature/image combinations.

[3]It should be noted that for a single image query, only histogram features have $tf \neq 1$.

5 Methods for reducing query evaluation time

Methods for reducing query evaluation time fall into two philosophically different classes. Evaluation can be stopped at a point where it is known that the top n ranked images can no longer change, (a "lossless" method), or the decision can be made to tolerate some (small) changes in image ranking in the final result (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.

5.1 Changes of image rank during query evaluation

Figure 2 shows the ranks of the images that are finally ranked 1 to 10 during the evaluation of a query. After 50% of the features have been evaluated, these images are all ranked in the top 40 (*i.e.* 8% of the total database). This already suggests that feature evaluation could be stopped early without too great a loss of precision.

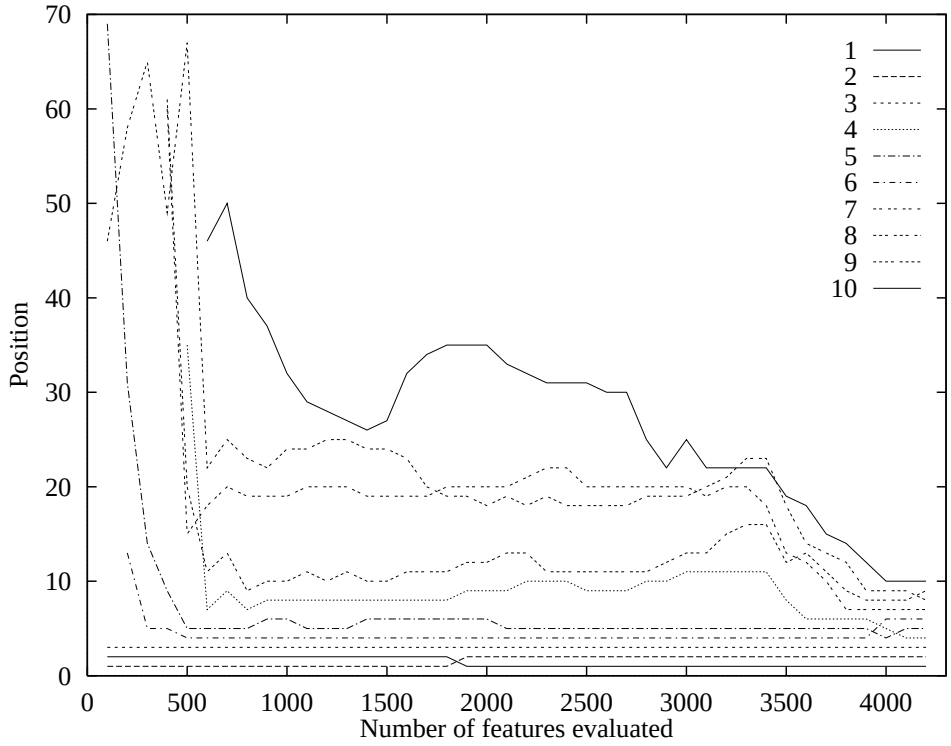

Figure 2: Ranks of the final top 10 images during query evaluation, without sorted features.

If the features are sorted according to their weights (as in Figure 1), the results are dramatically better, as shown Figure 3. In this case all 10 images are in the top 8% after less than 25% of features have been evaluated, and indeed all but one are in the top 20 (*i.e.* 4%) after 50% of the features have been evaluated. It is clear that early stopping could result in great improvements in response time, without dramatic changes in the final results. It is important to note that the amount of time saved by early stopping will be much better than linear in the number of features not evaluated, since it is precisely those features with high collection frequencies that take the longest time to evaluate.

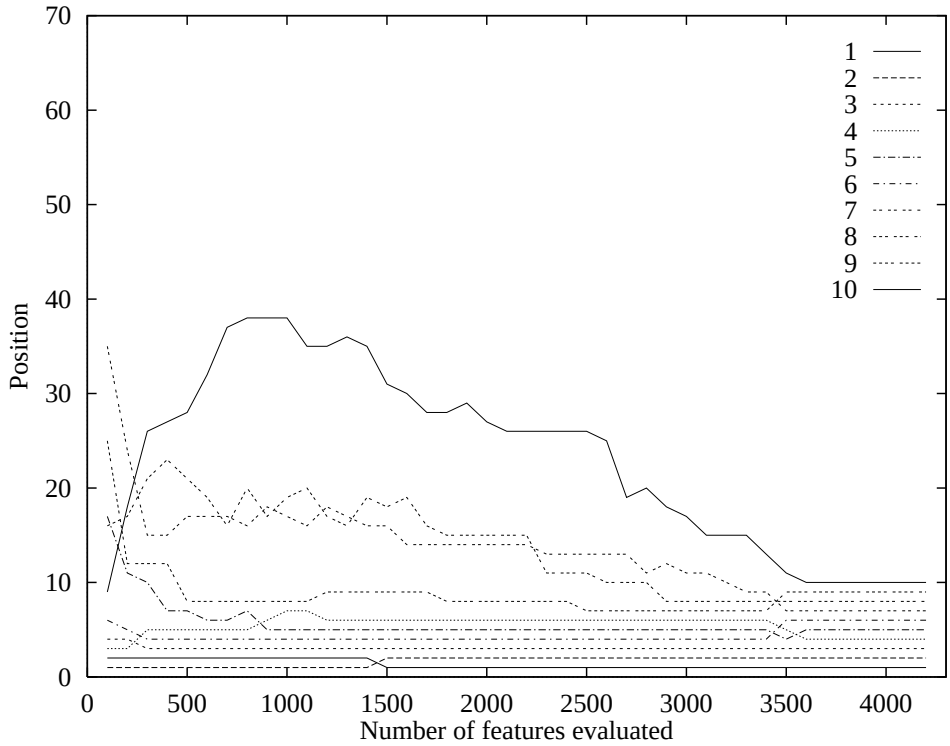

Figure 3: Ranks of the final top 10 images during query evaluation, with sorted features.

6 Lossless reduction of query evaluation time

Both lossy and lossless pruning techniques are based on sorting features according to their weights. In the lossless case, feature evaluation can be stopped when it is known that the maximum possible score increase from the remaining unevaluated features cannot change the composition of the top n ranked images (where n is specified by the user). If the exact order of the images in the top n is not important, evaluation can cease at this point. Alternatively, calculation of the weights of the top n images can be continued (ideally using an "uninverted" file). The equation giving the point at which evaluation can be stopped is

$$s_n(j) - s_{n+1}(j) > (J - j) \log^2 \left(cf_j^{-1} \right), \tag{5}$$

where J is the number of features in the query, j indexes features evaluated and $s_i(j)$ is the score of image i after the evaluation of feature j.

In our system, this approach is of little practical value, since this limit is not reached until a large proportion of the features has been processed. Any time saved by pruning at this point is offset by the expense of maintaining a sorted scoreboard and testing for this condition after each feature is evaluated.

7 Lossy reduction of query evaluation time

As indicated by Figure 1, great gains in query evaluation speed can be expected if the number of features evaluated is reduced. First, the number of costly accesses to the inverted file will be reduced. More importantly, since the features are sorted according

to their weights, the number of images for which scores must be updated for each feature increases dramatically as j increases. A typical distribution of collection frequencies is shown in Figure 4, which shows an almost exponential increase in the collection frequency

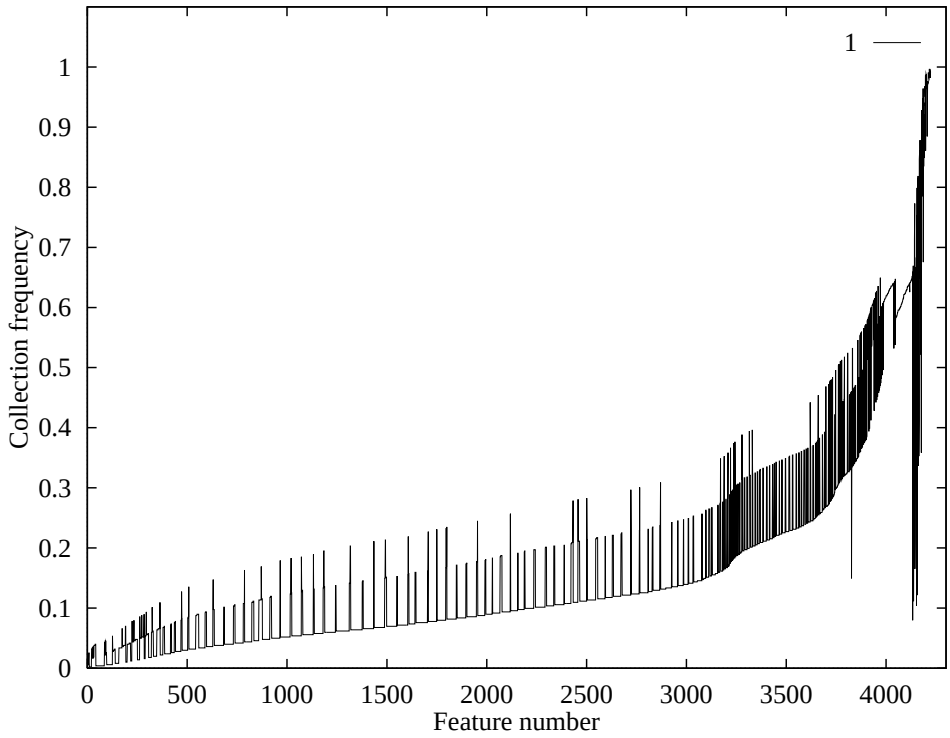

Figure 4: Feature collection frequencies when features are sorted by weights.

with j. It is not monotonic, because the weights according to which the features are sorted also depend on the document frequency in the query. It is clear, however, that the final features will have the greatest contribution to the evaluation time.

Whilst the cut-off point given by Equation 5 is too conservative, query evaluation time can be greatly reduced if lossy pruning can be tolerated. We must thus quantify the changes in system performance introduced by lossy pruning. To measure the quality of the results we use queries and relevance judgments from a study reported in [12]. Five users selected sets of images they regarded as relevant from the database of 500 images mentioned earlier. These data allow us to construct precision *vs.* recall graphs for the system response to each query for each user. Averaging over all users and queries indicates the overall performance of the system.

Figure 5 shows the average precision *vs.* recall graphs for a variety of feature evaluation cut-off points. It can be seen that stopping evaluation after only the first 20% of features results in only slightly worse performance than the evaluation of all features. Stopping after the evaluation of 80% or 90% of features can even yield better results than complete evaluation, suggesting that the contribution of the final features is essentially noise.

As can be seen from Table 2, the reduction is query evaluation time is, as expected, better than linear in the percentage of features evaluated. Evaluating 80% of the features, for instance, takes only 61% as long as full evaluation, and the results are at least as good. It is clear that this is an effective search pruning method. These values are within the "uninterrupted flow of thought" target described in §1. Better knowledge of individual feature significance might allow further improvement.

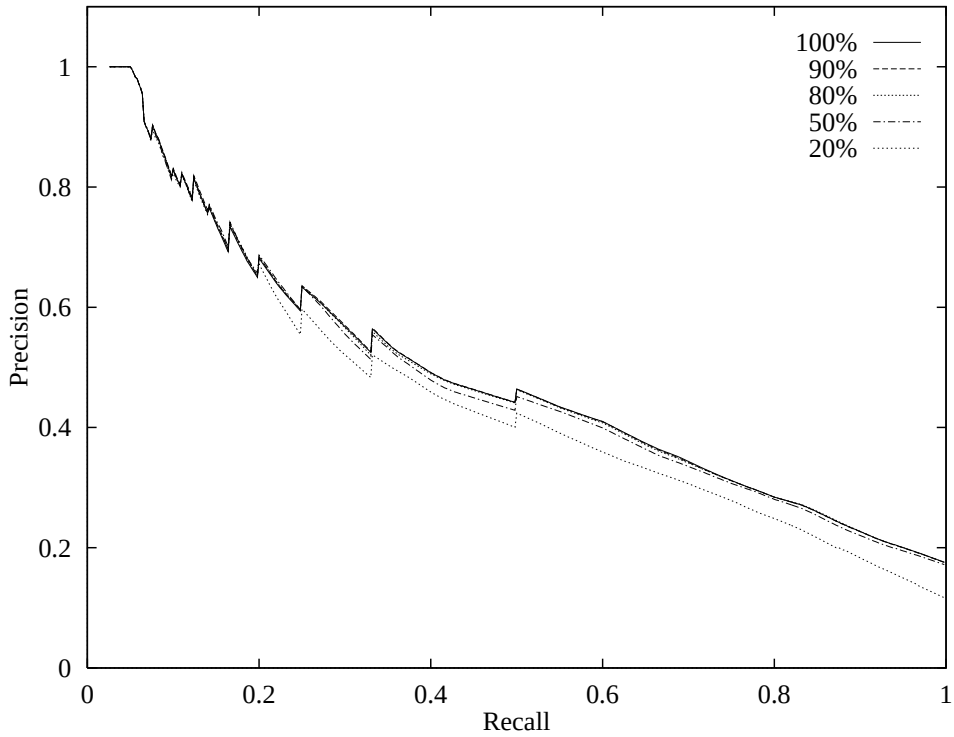

Figure 5: Average precision *vs.* recall when evaluation is stopped after a fixed percentage of features.

20%	0.08s
50%	0.20s
80%	0.43s
90%	0.59s
100%	0.71s

Table 2: Average evaluation times for varying percentages of features evaluated.

8 Fixed-time responses

In some situations it may be desirable to allow the user to specify the maximum time which the system is permitted to spend on query evaluation, or to set this as an internal parameter according to the usability criteria discussed in §1. Figure 6 shows the effect of several fixed time limits on average precision *vs.* recall. It can be seen that none of the limits causes a significant reduction in precision. The performance with the 0.5s limit is indistinguishable from the unlimited case. These results are for single image queries, with the average execution times given in Table 2. Worse performance might be expected for multi-image queries with larger numbers of features.

9 Conclusion

We have shown that the use of an inverted file data structure leads naturally to effective search pruning strategies for CBIR. Weights are assigned to features, which are then evaluated in sorted order. Such a simple linear pruning strategy is impossible when access is image by image. We have shown that a "lossy" pruning strategy does not lead to

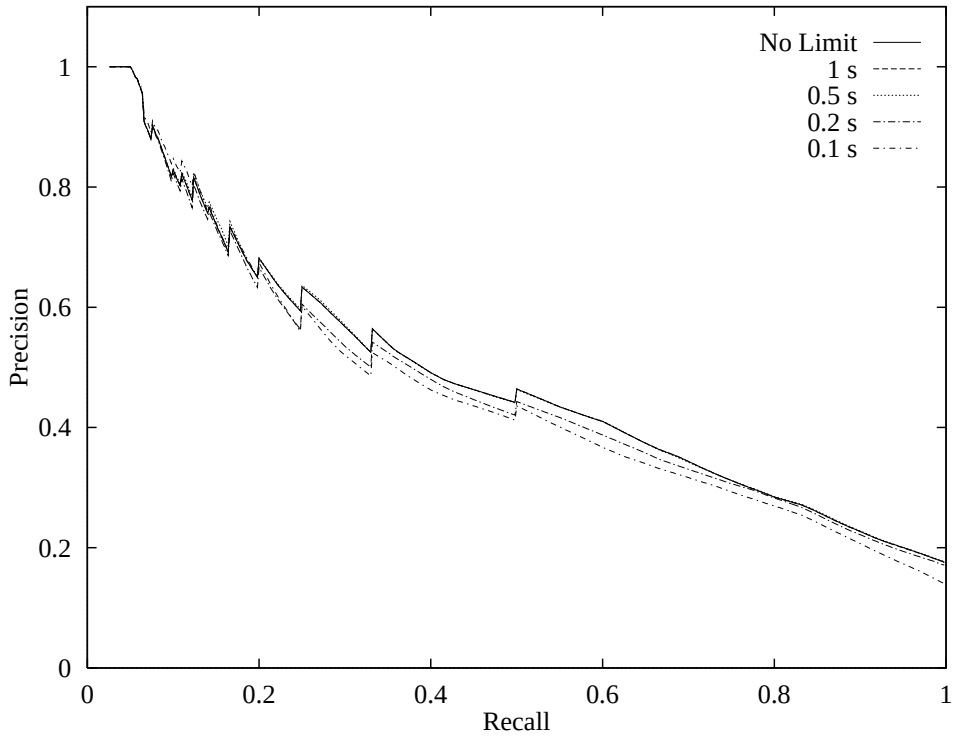

Figure 6: Precision *vs.* recall when evaluation is stopped after a fixed time.

significantly impaired system performance. Indeed, it can even improve performance when the contribution of very numerous features with low weights is essentially noise.

This pruning technique leads to system response faster than the key 1s usability criterion, and approaching the 0.1s "instant response" target. It also leads to graceful degradation of system performance as evaluation time is reduced. This will allow the user to specify an upper bound for evaluation time, thus selecting the desired trade-off between response time and the quality of query results.

10 Acknowledgments

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

References

[1] Jakob Nielsen. *Usability Engineering.* Academic Press, Boston, MA, 1993.

[2] Jakob Nielsen. The need for speed. Alertbox (web page: http://www.useit.com/alertbox/9703a.html), March 1997.

[3] David McG. Squire, Wolfgang Müller, Henning Müller, and Jilali 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. (to appear).

[4] Thierry Pun and David McG. Squire. Statistical structuring of pictorial databases for content-based image retrieval systems. *Pattern Recognition Letters*, 17:1299–1310, 1996.

[5] Stan Sclaroff, Leonid Taycher, and Marco La Cascia. ImageRover: a content-based browser for the world wide web. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 2–9, San Juan, Puerto Rico, June 1997.

[6] Anil K. Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, August 1996.

[7] Asha Vellaikal and C.-C. Jay Kuo. Hierarchical clustering techniques for image database organization and summarization. In C.-C. Jay Kuo, Shih-Fu Chang, and Sethuraman Panchanathan, editors, *Multimedia Storage and Archiving Systems III (VV02)*, volume 3527 of *SPIE Proceedings*, pages 68–79, Boston, Massachusetts, USA, November 1998.

[8] David A. White and Ramesh Jain. Algorithms and strategies for similarity retrieval. Technical Report VCL-96-101, Visual Computing Laboratory, University of California, San Diego, 9500 Gilman Drive, Mail Code 0407, La Jolla, CA 92093-0407, July 1996.

[9] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Van Nostrand Reinhold, 115 Fifth Avenue, New York, NY 10003, USA, 1994.

[10] John R. Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Storage & Retrieval for Image and Video Databases IV*, volume 2670 of *IS&T/SPIE Proceedings*, pages 426–437, San Jose, CA, USA, March 1996.

[11] John G. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, 1993.

[12] David McG. Squire, Wolfgang Müller, and Henning Müller. Relevance feedback and term weighting schemes for content-based image retrieval. In *Third International Conference On Visual Information Systems*, Amsterdam, The Netherlands, June 2–4 1999. (to appear).

[13] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.