Long-Term Learning from User Behavior in Content-Based Image Retrieval

Muller, Henning; Muller, Wolfgang; Squire, David; Marchand-Maillet, Stéphane; Pun, Thierry

UNIVERSITE DE GENEVE

CENTRE UNIVERSITAIRE
D'INFORMATIQUE
GROUPE VISION

Date: Mai 31, 2000
N° 00.04

TECHNICAL REPORT

VISION

Long-Term Learning from User Behavior in Content-Based Image Retrieval

Henning Müller Wolfgang Müller David McG. Squire[1]
Stéphane Marchand-Maillet Thierry Pun *

Computer Vision Group
Computing Science Center, University of Geneva
24 rue du Général Dufour, CH - 1211 Geneva 4, Switzerland

[1]Computer Science and Software Engineering, Monash University, Melbourne, Australia

e-mail: muellerh@cui.unige.ch pun@cui.unige.ch

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

Abstract

This article describes an algorithm for obtaining knowledge about the importance of features from analyzing user log files of a content-based image retrieval system (CBIRS). The user log files from the usage of the *Viper* web demonstration system are analyzed over a period of four months. Within this period about 3500 accesses to the system were made with almost 800 multiple image queries. All the actions of the users were logged in a file.

The analysis only includes multiple image queries of the system with positive and/or negative input images, because only multiple image queries contain enough information for the method described. Features frequently present in images marked together positively in the same query step get a higher weighting, whereas features present in one image marked positively and another image marked negatively in the same step get a lower weighting. The *Viper* system offers a very large number of simple features. This allows the creation of flexible feature weightings with high values for important and low values for less important features. These weightings for features can of course differ between collections and as well between users. The results are evaluated with an experiment using the relevance judgments of real users on a database containing 2500 images. The results of the system with learned weights are compared to the system without the learned feature weights.

Keywords: long term learning, log file analysis, content-based image retrieval, web usage analysis, multimedia retrieval

1 Introduction

Much has been written about Relevance Feedback (RF) in content-based image retrieval (CBIR) [18]. Most feedback methods only takes into account one query step and the knowledge obtained from older query steps of the same session or of other query sessions is forgotten. Often, the feedback is limited to one positive image [16] or several positive feedback images [4]. Only few systems offer both positive and negative feedback as Surfimage [6, 27] and *Viper* [26]. Even these systems often have problems with too much negative feedback as described in [11], although solutions similar to those already used in text retrieval (TR) [17] exist.

Image browsers like *PicHunter* [5] offer the possibility to have feedback over more than one step and thus to really learn from the user interaction in order to find one target image. Using a sequence of queries to discover the user's goal creates another problem whenever the user changes the goal of a query in the querying process. Solutions to this problem referred to as "moving targets" are given in *TrackingViper* [13].

Yet, existing learning algorithms mostly try to find out the goal of a user over one or a few feedback steps. Minka [10] proposes across-session learning for *FourEyes* in *PhotoBook*. In [8], an approach to cluster images marked together positively and divide images marked negatively from the clusters is explained. In the domain of *collaborative filtering* [7], user judgments have been used to propose new items to users based on items being marked together positively by other users. This has been applied to art images of a museum as well [1]. The search for user preferences by giving positive and negative examples for web pages has also been studied [15]. Bayesian networks have been used to find out if an unknown page might fit to the users' profile or not. This supervised learning is out of the scope of this paper as we want to use unsupervised learning techniques to avoid additional work for the user. We also want to learn information for new queries and not just improve one already known query by augmenting important features.

In the domain of electronic commerce, log files resulting from web usage have been analyzed for a long time and the knowledge from this analysis is employed to improve new systems and to adapt them to the users' needs [28]. Part of this research concentrates on analyzing the behavior of users within webpages and the links they use [2]. In the domain of electronic commerce, there are many different concepts to identify users and track their activities, but problems arise with people just trying out pages and making very short visits. Longer visits can be analyzed to facilitate the design of a web page.

The quality of user data gained from the internet might not be the highest. Nevertheless, we can learn from the usage information, and the related analysis in this paper shows that we can get qualitatively and quantitatively better results, even by using potentially poor web user data.

2 The *Viper* system

The *Viper* system is a CBIRS that is described in more detail in [23, 24]. The system uses many techniques known from TR applications and aims at incorporating them into the domain of CBIR.

2.1 System Architecture

The main difference compared with other systems is the presence of a very large number of more than 85000 possible features. Most images contain between 1000 and 2000 of theses features. The access method to the features is the *inverted file*, which is the most common access method used in TR. Thus, *Viper* allows a fast and efficient access to the large number of features [12].

The emphasis of the project is on user interaction. Hence, it embeds several interaction strategies using several steps of positive and negative feedback. Both online and offline learning are employed in the system. *Viper* offers a good flexibility for learning as it has a very large number of features for the creation of feature weights. Especially the extensive use of negative feedback has shown to be very effective [11] and is also very important for the long term learning approach in this paper.

2.2 *Viper* Features

The system used for this study implements four different groups of image features:

- A global color histogram based on the HSV color space which corresponds roughly to the human color vision [22];
- local color blocks at different scales for fixed regions by using the mode color for each of the fixed blocks; the image is successively partitioned into four equally sized blocks and each block is partitioned again four times;
- global texture characteristics are represented by the histograms of the response to gabor filters of different frequencies and directions; gabor filters are known to be a good model for the human perception of edges [9];
- local Gabor filters at different scales and regions by using the same blocks as for the local color features and applying Gabor filters with different directions and frequencies to these blocks.

These features are only low level features, but because of the high number, very complex queries can be constructed with them. Higher level features like image regions may provide better results, but we still suffer from the semantic gap between the semantics the user is looking for and the visual content the system can offer.

2.3 Weighting schemes

We have implemented several weighting schemes known from the TR literature [21]. They are all based on the collection and document frequencies of the features. For the experiments in this paper, we use the *inverse document frequency* weighting, which weights the features in the following way:

$$relevance_j \quad : \quad \frac{1}{N}\sum_{i=1}^{N}(tf_{ij}\cdot R_i)\cdot log^2\left(\frac{1}{cf_i}\right) \tag{1}$$

$$score_{kq} \quad = \quad \sum_j (tf_{kj}\cdot relevance_j)\,, \tag{2}$$

where *tf* is the *term frequency* of a feature, *cf* the *collection frequency* of a feature, j a *feature number*, q corresponds to a query with $i = 1..N$ input images, k is one *result image* and R_i is the *relevance* of an input image i within the range $[-1;1]$.

We can see in Equation 1 that the final result mainly depends on the collection frequency of a feature. Rare features are weighted high, whereas features very common in the collection are weighted low because they contain less information. The term frequency of a feature in the input images has a has a minor influence. We can see in Equation 2 that besides the relevance factor for a feature, the term frequency of the feature in the resulting image has a small influence on the final score.

3 Learning Feature Weights from User Behavior

Reference [26] points to the web demonstration of the CBIRS *Viper* we used for this study. Every time a user accesses this page and does an action, it is logged with a time stamp. Like this, we can always see what the user did and which problems he might have encountered with the system. This also offers the possibility to make an offline analysis of the data to better suit the information needs of a user. The host name of the user is also saved, but no other private data.

3.1 Analyzing the Log Files

This section gives a general overview of the data we logged into a file. Between September 1999 and January 2000, we had 3500 accesses to the system. About half of the accessors just looked at random or sorted image sets or watched the parameters, but about 1700 accesses actually were queries. This shows that many people visited the page, but a large number of them just played around with the system. This can be confirmed with the fact that only 24 of the 201 hosts which accessed the system had more than 20 actions with the system. About 40 percent of the queries came from different hosts within the University of Geneva. Of the 1700 queries, 786 where multiple image queries. Only multiple image queries contain enough information for the algorithm we want to employ.

Table 1: The different functions of the system and their use in the web demonstration

Chose Database	668 times
Browse Image Names	251 times
Image Queries	1586 times
Random Images	586 times
Change Options	114 times
Clear Judgments	100 times

In the log files, the query data from 10 different databases is regarded. It is hard to map the importance of features from one database to another database although they use the same set of features. The distribution of the features actually present in the database is very different for every database. Only the histogram features for color and texture are present in a very large number of images in every database.

From the log files, we could also analyze the problems the user had with our system. Several people did queries without marking any image as relevant. As a result, we inserted a comment telling the user that at least one image needs to be marked. Another problem encountered while analyzing the log files was related to using too much negative feedback. This can as well remove all the important features from the query and lead to bad results. We therefore have implemented a modified version of Rocchio's formula [17] for separately weighting positive and negative relevance feedback [11].

3.2 Learning from Log Files

The two rationales for our learning algorithm are:

- Features which occur often in two images marked together positively in the same query step should have a higher weighting than others;

- features which occur often in images marked once positive and once negative in the same query step should have a low weighting.

Based on these principles, we identified all pairs of images marked together. Queries with two input images just have one pair, whereas queries with three images have three and queries with four images have six. Thus, the number of image pairs in a query with n images is:

$$\text{number of image pairs} = \frac{n \cdot (n-1)}{2}. \tag{3}$$

The 786 multiple image queries lead to more than 31.000 image pairs marked together. If images are marked together, both negatively, the image pair is discarded, as this does not contain much information. Images can be marked together negatively for different reasons and may not have anything negative in common.

We then analyzed which features the two images of a pair have in common. Positive image pairs lead to a positive mark for the features they have in common and negative image pairs to a negative mark. Negative pairs have in general a smaller number of features in common. On average the image pairs have slightly more than 300 features in common. In total, the 31.000 image pairs lead to 10 million feature marks (6.1 million positive and 3.9 million negative). We separately analyzed the image database of the Télévision Suisse Romande (TSR) because our user experiments are based on this database. For the TSR database, we had 3.800 image pairs and 1.02 million feature pairs (0.47 million positive and 0.55 million negative), which represents about 10% of all the accesses to the system.

Features with a very high collection frequency like the histogram features occur about the same time as positive and negative pairs. Hence, their respective weight should stay very similar as before.

The additional factor we want to calculate should be in the range of $[0;2]$ to allow poor features to disappear completely and good features to be weighted significantly high. Features which occur only negatively should have a value of zero and features which occur only positively should have a value of 2.

This leads to the following simple formula for the additional factor $factor_j$:

$$factor_j = 1 + \frac{p_i}{p_i + n_i} - \frac{n_i}{p_i + n_i}, \tag{4}$$

where j is the feature number, p_j then number of positive marks for feature j and n_j the number of negative marks.

The new weighting formula for a feature is basically the same as it was before with only the additional factor from Equation 4 being calculated and included into Equation 1 as can be seen in Equation 5.

$$relevance_j = factor_j \cdot \frac{1}{N} \sum_{i=1}^{N} (tf_{ij} \cdot R_i)\, log^2 \left(\frac{1}{cf_i} \right). \tag{5}$$

Because improvements were lower than expected for the queries with relevance feedback (see Figure 2), we implemented a second factor similar to the factor in Equation 4 for comparison. We think that a complete disappearance of poor features might reduce the possibility to move in feature space, an effect which is stronger visible in feedback queries. Hence, we implemented a factor where the negative value can only reach a minimum of 0.25, whereas the maximum factor can be up to four, when only positive marks occur. If positive and negative marks occur with the same frequency, the factor stays at one. The resulting $factor2_j$ is obtained by rescaling the positive and negative parts of $factor_j$ in a different way as can be seen in Equation 6.

$$factor2_j = \begin{cases} 0.25 + \frac{factor_j}{0.75} & : \; factor_j < 1 \\ 1 & : \; factor_j = 1 \\ 1 + (factor_j - 1) \cdot 3 & : \; factor_j > 1 \end{cases} \tag{6}$$

For the calculation of the weight, *factor2*$_j$ is used in exactly the same way as *factor*$_j$ in Equation 5.

The fact that there are slightly more pairs marked together positively than there are negative pairs may lead to a different quantization of positive and negative parts, but does not alter the quality of the results.

4 Experimental Results

To analyze the success of this method, we use a user experiment performed in [12]. This includes a very heterogeneous database of 2500 images from the Télévision Suisse Romande (TSR). 14 queries were presented to 3 users for relevance judgments. The users had to mark all the images in the database they regard as being similar to each of the 14 query images. Interestingly, the result sets for each user differ strongly in size and also in the images being selected. Similar effects were already reported in [25].

To evaluate the performance, we use precision/recall (PR) graphs which are the standard evaluation method in TR [19] and are more and more used in CBIR [24]. The results shown below are the PR graphs averaged over the relevance sets of all users and all queries from the user experiment. To simulate relevance feedback based on the user judgments, we used the algorithm explained in [11]. We feed back all images the user regards as relevant and which are in the first 20 images the system returns for the initial query.

The training data is only taken from the usage of the web demonstration system and does not have any connection with the user experiment we performed.

We see in Figure 1 that the results of the system with the additional factor are up to 10% better than the original graph when all queries of the same database (TSR) are used to calculate the weights. Using all queries of all the different databases still gives an improvement of 7% to 8%, but only in the beginning of the graph. The overall improvement is lower when using the data of all databases.

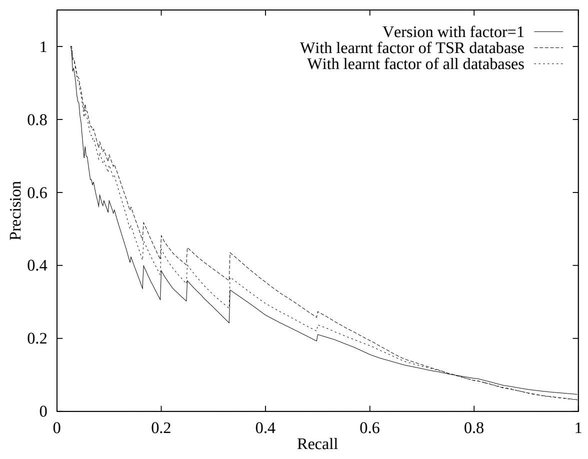

Figure 1: PR-Graph for a system with and without a learned factor (without feedback).

In Figure 2, we can see that the results of the first feedback step are much better (up to 100% in the middle parts) than the results before feedback (compare Figure 1). An improvement in the beginning of the graphs is especially important because this part represents the images the user actually views. The results with the learned factor are significantly better than without the factor, even on this high level. This shows that the gain with the additional $factor_j$ is not just limited to one query step as it favors image pairs already marked together.

When we use the factor learned from all the different databases, the results are about 3% to 5% better than without the learned factor. We think that this improvement was only small because the additional factor can become 0 for bad features which limits the flexibility to move in feature space. As a consequence, we repeated the experiments with a second factor explained in Equation 6.

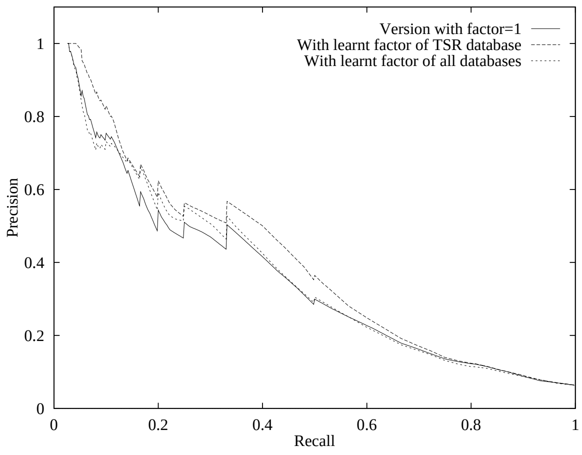

Figure 2: PR-Graph for a system with and without a learned factor (with feedback).

Figure 3 compares the results obtained using the two factors, respectively learned on the queries of the TSR database and learned with all databases. We can see that the results with the second factor are in both cases better for $factor2_j$. The beginning part of the graphs is almost identical, but in the middle parts of the graph the results improve with the second factor.

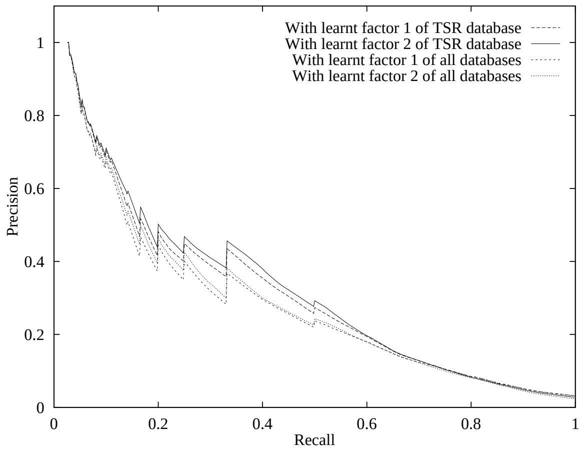

Figure 3: Comparison of the two different weighting factors (without feedback).

Figure 4 shows a comparison of the results obtained using the two factors for the queries with feedback. Here, we can clearly see the improvements of $factor2_j$ compared to $factor_j$ of up to 7%, especially in the middle parts of the graph. This shows that it might be better to let the factor always be above zero to not reduce the mobility in feature space. The small drop off in the beginning of the curve for $factor2_j$ can be explained with one query with only very few features and basically no textures, where the first returned image was non-relevant.

We also made some experiments where we tried to learn $factor_j$ based on feedback queries performed on a completely different database. The results were basically the same as without learning. The results using all the feedback from every database show clearly that not much feedback of the same database is necessary to improve the results of a query. Feedback from other databases does not change the results much as the feature space is only very sparsely populated and the databases populate different areas of the feature space.

We see that calculating weights from user log files brings strong improvements, especially, when the factor is learned based on queries of the same database. Learned over all queries and all databases, the improvement was not extremely strong, but clearly visible. Defining a user profile for learning could bring even stronger improvements, especially if the user is often performing

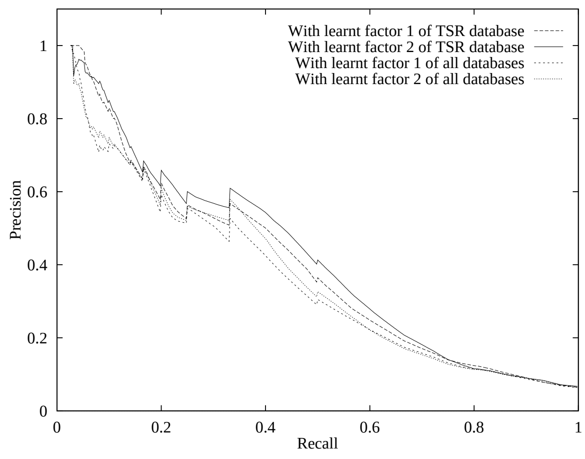

Figure 4: Comparison of the two different weighting factors (with feedback).

similar search tasks. Therefore, we propose to have a hierarchy of factors, corresponding to a user, a domain and a global factor to be learned. To do this, a user identification needs to be inserted into the log file.

5 Conclusions and Further Work

In this paper, an approach is presented on how to learn the importance of features in CBIR from log files containing user behavior of a web demonstration system. The problems of log files on the web is of course that we do not know much about the quality of the user data. Many people may come to a web page to try out the system and to see how it reacts and might even challenge the system with inconsistent data. This means that we can not always learn much from this kind of data. With the proposed approach, artifacts can be minimized as combinations of image pairs with high feature similarity have much more importance than image pairs with low feature similarity. The experiments with the factors we use show that, even with this kind of data, a significant improvement in retrieval quality can be reached. This is mostly true when the feature importance is learned on the same database. In this case, the results are very good.

Much better results will be possible once the data is obtained from serious users and even better if the study is restricted to a certain domain or a certain user. Like this specific user profiles or group profiles can be learned. We propose a hierarchy of learned feature weightings on a user, domain and global level.

Besides the learning of a feature weight for future queries we can evaluate the usefulness of features. This can also be used for the creation of new features. New features can be extracted for the old images and can directly be evaluated by using this method with the old log files.

More work needs to be done on finding an optimal factor to calculate a feature weight. We only proposed a very simple factor without any optimization. Another promising approach is to not only analyze pairs of images marked together, but directly evaluate multiple image queries by looking at all the images marked in a query. Features contained in $n > 2$ images marked together in the same query step should for example get a much higher weighting than features only contained in two images.

References

[1] Active web museum. `http://abyss.eurecom.fr:1111/AWM/login.html`, 2000.

[2] B. Berendt and M. Spiliopoulou. Analysis of navigation behaviour in web sites integrating multiple information systems. *VLDB Journal: Special Issue on Databases and the Web - to appear*, 2000.

[3] *IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'99)*, Fort Collins, Colorado, USA, June 22 1999.

[4] Compass web page. `http://compass.itc.it`, 2000.

[5] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos. Target testing and the `PicHunter` Bayesian multimedia retrieval system. In *Advances in Digital Libraries (ADL'96)*, pages 66–75, Library of Congress, Washington, D. C., May 13–15 1996.

[6] Surfimage webdemo. `http://www-rocq.inria.fr/cgi-bin/imedia/surfimage.cgi`, 1999.

[7] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, Vienna, Austria, February 1999. IOS Press.

[8] C. S. Lee, W.-Y. Ma, and H. Zhang. Information Embedding Based on User's Relevance Feedback for Image Retrieval. In Panchanathan et al. [14]. (SPIE Symposium on Voice, Video and Data Communications).

[9] W. Y. Ma, Y. Deng, and B. S. Manjunath. Tools for texture- and color-based search of images. In B. E. Rogowitz and T. N. Pappas, editors, *Human Vision and Electronic Imaging II*, volume 3016 of *SPIE Proceedings*, pages 496–507, San Jose, CA, February 1997.

[10] T. Minka. An image database browser that learns from user interaction. Master's thesis, MIT Media Laboratory, 20 Ames St., Cambridge, MA 02139, 1996.

[11] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun. Strategies for positive and negative relevance feedback in image retrieval. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR 2000)*, Barcelona, Spain, September 2000. IEEE.

[12] H. Müller, D. M. Squire, W. Müller, and T. Pun. Efficient access methods for content-based image retrieval with inverted files. In Panchanathan et al. [14]. (SPIE Symposium on Voice, Video and Data Communications).

[13] W. Müller, D. M. Squire, H. Müller, and T. Pun. Hunting moving targets: an extension to Bayesian methods in multimedia databases. In Panchanathan et al. [14]. (SPIE Symposium on Voice, Video and Data Communications).

[14] S. Panchanathan, S.-F. Chang, and C.-C. J. Kuo, editors. *Multimedia Storage and Archiving Systems IV (VV02)*, volume 3846 of *SPIE Proceedings*, Boston, Massachusetts, USA, September 20–22 1999. (SPIE Symposium on Voice, Video and Data Communications).

[15] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Journal on Machine Learning*, 27:313–331, 1997.

[16] QBIC^{TM} – IBM's Query By Image Content. `http://wwwqbic.almaden.ibm.com/~qbic/`, 1998.

[17] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System, Experiments in Automatic Document Processing* [20], pages 313–323.

[18] 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(5):644–655, September 1998. (Special Issue on Segmentation, Description, and Retrieval of Video Content).

[19] G. Salton. Evaluation parameters. In *The SMART Retrieval System, Experiments in Automatic Document Processing* [20], pages 55–112.

[20] G. Salton. *The SMART Retrieval System, Experiments in Automatic Document Processing.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1971.

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

[22] 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.

[23] D. M. Squire, H. Müller, and W. Müller. Improving response time by search pruning in a content-based image retrieval system, using inverted file techniques. In CBAIVL99 [3], pages 45–49.

[24] 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 11th Scandinavian Conference on Image Analysis (SCIA'99)*, pages 143–149, Kangerlussuaq, Greenland, June 7–11 1999.

[25] D. M. Squire and T. Pun. Assessing agreement between human and machine clusterings of image databases. *Pattern Recognition*, 31(12):1905–1919, 1998.

[26] Viper webdemo. web page: `http://viper.unige.ch/`, 1999.

[27] A. Winter and C. Nastar. Differential feature distribution maps for image segmentation and region queries in image databases. In CBAIVL99 [3], pages 9–17.

[28] K.-L. Wu, P. S. Yu, and A. Ballman. Speedtracer: A web usage mining and analysis tool. *IBM Systems Journal on Internet Computing*, 37(1), 1998.