Content-Based Image Retrieval Systems - Reviewing and Benchmarking

Harald Kosch
Chair of Distributed Information Systems,
University Passau, Germany
Paul Maier
Chair for Image Understanding and Knowledge-Based Systems,
Technical University Munich, Germany

Abstract

The last detailed review of Content-based Image Retrieval Systems (CBIRS) is that from Veltkamp and Tanase [VT02], updated in 2002. Since then, many new systems emerged, other systems were improved, but many systems are no longer supported. This paper reconsiders the systems described by Veltkamp and Tanase and proposes in addition for a selection of existing CBIRS a quantitative comparison.

For this purpose we developed a benchmarking system for CBIRS based on how accurate they could match ideal (ground truth) results. As measure we introduced the $\widetilde{Rank}_{WRN}$, which we developed based on the *Normalized Avarage Rank*. Our measure allows fair and accurate comparisons (due to multiple similarity values) of different CBIRS with respect to different queries. None of the other benchmarks allow this comparison.

The choice for a system to be compared quantitatively was motivated by availability of the source or runtime code and the liveness (i.e., active development on the CBIRS). The benchmarked systems were SIMBA, SIMPLIcity, PictureFinder, ImageFinder, Caliph&Emir, VIPER/GIFT and Oracle Intermedia. Results show that VIPER/GIFT performs best in the settings of our benchmark, Caliph&Emir is second and SIMBA third. The default parameter settings of the systems tend to show the best results. MPEG-7 descriptors, which are used in Caliph&Emir, show a good performance in the benchmarking.

1 Introduction

Research in Content-Based Image Retrieval is devoted to develop techniques and methods to fulfil image centered information needs of users and manage large amounts of image data. It is nowadays an even more vivid research area than when Veltkamp and Tanase provided an overview of the CBIR landscape [VT02], which showed the state-of-the-art at that time. Since then there has been much development in multiple directions. The question arises how those projects evolved: which have been concluded, which canceled, which are still active or have spawned new projects. Yet another question the report by Veltkamp and Tanase didn't answer is how the systems perform. This question alludes to the broad topic of benchmarking information retrieval systems in general and content-based image retrieval systems specifically. A number of research papers have addressed this topic ([MMS$^+$01b, MMW06, MMS01a]), focusing on how to establish standards for benchmarking in this area.

We tried to follow the paths of the systems described in [VT02] and reviewed new developments. Assuming a bird's eye perspective, we found that on the one hand many of the older projects were finished or have not been further developed. For many of these efforts it remains unclear what further developments have occurred, if any. On the other hand there are a lot of new projects. Among them we found a tendency to implement standards like MPEG-7 as well as a simplification of user interfaces. Of the recent systems none expects the user to provide image features as a query, for example. Most of the efforts try to address the problem of CBIR in a general fashion, they don't focus on specific domains. The projects are often spin-offs or specialisations of other, general CBIR systems or frameworks. Since presenting all of our findings would exceed the scope of this article we just present an overview of the systems we see as "live": active and ongoing projects, which are either under active development or research, or are being used in products. Furthermore, the availability of the source or runtime code for testing was an obvious prerequisite. The complete report can be requested from Paul Maier[1].

In this article we focus on the mentioned systems and their development as well as several new systems and on benchmarking a small subset of these systems. We don't review the development of CBIR as a whole. In their recent work Datta et al. ([DJLW08]) provide a broad overview of recent trends, ideas and influences in CBIR.

The rest of the article is organized as follows: section 2 provides the overview of "live" projects. Also, it identifies the systems chosen for the benchmark. Section 3 explains how the systems were benchmarked, particularly our new measure $\widetilde{Rank}_{WRN}$ is introduced. Section 4 discusses the benchmark results and section 5 finally sums up and provides an outlook for future work.

2 Reviewed Systems

Table 1 shows an overview of all "live" systems. The leftmost column lists the systems, sorted alphabetically. The second column holds references for the systems, then follow the columns showing the features: support of texture, color, shape, search by keywords, interactive relevance feedback, MPEG-7 descriptors [DK08] (for example *DominantColor*), whether they are specifically designed to be used as web search engine, whether they use classification techniques or segment images into regions.

The column *designed for web* is primarily a matter of scalability: a web search engine must be able to handle millions of images, performing searches on vast databases within seconds. But it also means that some kind of image accumulation mechanism, such as a web crawler, must be provided.

The column *classification* specifies if a system uses some kind of classification mechanism, clustering images and assigning them to classes. The last column, *image regions*, indicates whether a system applies some kind of image segmentation. This is not the same as *shape*: image regions might be used to identify shapes in an image, but they could also simply be a means of structuring the feature extraction (see for example VIPER in section 4.5).

One can recognize that keyword search is very often supported as well as relevance feedback. There seems to be a consensus that image retrieval in general cannot solely rely on content-based search but needs to be combined with text search and user interaction.

MPEG-7 image descriptors are still seldom used, but especially new systems or new versions of systems tend to incorporate these features. Many of the systems are designed to work on large image sets ($>$ 100 000 images in the database), but they are rarely designed specifically as a web search engine. Currently, only Cortina seems to be designed that way.

Seven CBIR systems have been chosen to for our benchmark:

> SIMBA, SIMPLIcity, PictureFinder, ImageFinder, Caliph&Emir, VIPER/GIFT, Oracle Intermedia

The systems were chosen following three criteria: (1) availability and setup: we needed to obtain and setup the CBIRSs in order to benchmark them within our own environment. Setup failed with some

[1]Contact by email: maierpa@in.tum.de

System	reference	texture	color	shape	keywords	interactive relevance feedback	MPEG-7 support	designed for web	classification	image regions
Behold	[YHR06]	*	*		*					*
Caliph&Emir	[LBK03]	*	*		*		*			
Cortina	[QMTM04]	*	*		*	*	*	*	*	
FIRE	[DKN04]	*	*		*	*				*
ImageFinder	[att07]	?	?	?					?	*
LTU ImageSeeker	[LTU07]	*	*	*	?	?			*	*
MUVIS	[GK04]	*	*							
Oracle Intermedia	[oim07]	*	*	*	*[2]					
PictureFinder	[HMH05]	*	*	*	*	*				*
PicSOM	[KLLO00]	*	*	*		*	*			*
QBIC	[FSN$^+$01]	*	*	*	*	*				
Quicklook	[CGS01]	*	*	*	*	*				
RETIN	[CGPF07, FCPF01]	*	*			*				
SIMBA	[Sig02]	*[3]	*							
SIMPLIcity	[LWW00, WLW01]	*	*	*					*	*
SMURF	[VV99]		*	*						
VIPER/GIFT	[Mül02]	*	*			*				*

Table 1: Overview of all live CBIR systems. For each CBIRS (rows), its features are listed (columns). A star ('*') in a column marks the support for a feature and a question mark ('?') if the support is unknown. If a system doesn't support the feature, the table entry is left empty.

[2]Keyword search in Oracle Intermedia depends on the design of the database being used to store the images. Generally, multimedia data can be arbitrarily combined with e.g. text data and searched with corresponding SQL queries.

[3]Whether a feature in SIMBA is actually a color or texture feature depends on the kernel function used.

obtained systems, which were therefore excluded; (2) visual similarity only: in our benchmark we focus on visual similarity search. Therefore, systems were required to only use visual similarity properties for their search. We excluded additional features, like text search or semantic analysis (e.g., automatic annotation); (3) query by example: this method was required to be supported by all CBIRSs.

There are a number of interesting systems which comply with the mentioned requirements, but have not been tested for other reasons. There is for example the Blobworld [CTB$^+$99] system, which turned out too time-consuming to setup. Three other interesting systems which have not been tested mainly due to time constraints are PicSOM [KLLO00], Quicklook^2 [CGS01] and FIRE [DKN04][4].

3 Benchmark Structure and Measurement

A new benchmark was developed to test and compare the chosen systems. This section outlines the aim of the benchmark, describes its overall structure, states which images where used in the database and how a ground truth was build from those images. Finally, our performance measure is introduced, as well as two new methods to compare CBIR systems based on these measures.

3.1 Benchmark Methods

In recent years efforts were made to analyse the possibilities for benchmarking CBIR systems and to propose standard methods and data sets ([MMS$^+$01b, MMW06, MMS01a, ben]). Since 2003 the *cross language image retrieval track* ImageCLEF ([ima05]) is run every year as part of the *Cross Language Evaluation Forum*. The focus of this track is on image retrieval from multilingual documents. A review of evaluation strategies for CBIR is presented in [DJLW08]. Still there seems to be no widely accepted standard for CBIR benchmarking, neither a standard set of performance measures nor a standard data set of images for the benchmarks.

A CBIR system has many aspects which could be subjected to a benchmark: the retrieval performance of course, its capability to handle large and very large image sets and the indexing mechanism to name just a few. In this work we focus on the retrieval performance aspect.

Aim Of The Benchmark We compared CBIRSs to each other based on their *accuracy*, i.e. their ability to produce results which accurately match ideal results defined by a ground truth. We assume a CBIRS does a good job if it finds images which the user himself would choose. No other queues were used, like e.g. relevance feedback. We used *query by example* only, no query by keyword, sketch or others. Also, no optimization for the image sets was done. The intent was to test the systems without providing them with prior knowledge of the image database. We concentrated on visual similarity only, the benchmark does not address any other aspects, e.g. resource usage and scalability. Those aspects, while of at least equal importance, where left out mainly due to time constraints.

The basic requirement of a CBIRS, is to fulfil a user's information need: finding useful information in form of images. We assume a CBIRS fulfils an information need if it finds *visually similar* images. Accordingly, we define *similarity* of images as follows

Definition: Image Similarity Two or more images are taken to be *similar* if spontaneously viewed as visually similar by human judges.

Accuracy in this work is simply defined through the ideal order imposed on result images by similarity: Images more similar to a query are appear before less similar images in the result. If the actual

[4]The authors of PicSOM and Quicklook^2 both offered to run the tests within their environments and send back the results. FIRE is freely available as open source software.

ordering in the result violates this ideal ordering accuracy decreases. In the following we characterize a system's performance by its accuracy.

3.2 Image Database

The images used for our benchmark are divided into domains of roughly similar images. The intention was to have a fairly representative sample of images, roughly grouped by some sort of image type. The following list explains which image domain represents which common type of image (and lists the query images):

Band: [5] Images of a street band. All the images are grey-scale. They represent black and white photographs and typical images showing a group of people.

Blumen (flowers): [6] Photographs of single flowers, taken in a natural environment. Typical examples are photographs of single (natural) objects with nature as background (meadows, rocks etc.).

Kirschen (cherry): [7] Photographs of blossoming cherry trees, taken in some kind of park. This is representative for trees, bushes, colorful images, mixed with artificial structures like pathways and buildings.

Textur (texture): [8] These are texture images. That means, they are photographs of various surface structures and intended for use as texture in computer graphics. Typical examples are texture-only images, like photographs of tissue, cloth, wall paint and the like.

Urlaub (vacation): [9] Photographs from someones vacation. Typical examples are vacation/leisure time photographs.

Verfälschung-Wind (distortion-wind): A graphical filter (creating a motion blurr which looks like wind) was applied to a source image (which is used as query image), creating a sequence of images with increasing filter strength. This domain will also be referred to as *Verfälschung* for short.

Each domain additionally contains several images from all other domains to create a "background noise".

Image Sources

The images have been taken from several sources which provided them for free use. The sources are indicated as footnotes in the above list except for *Verfälschung*. The images of this domain are altered versions of an image which is distributed together with Caliph&Emir [LBK03].

The images from the *University of Washington* and *Benchathlon.net* seem to be public domain, no licence is given. *Benchathlon.net* clearly states that up-loaders should ensure that images are not "copyright encumbered". The terms for images from *ImageAfter* allow free use except for redistributing them through an online resource like *ImageAfter*[10].

[5]`http://www.benchathlon.net/img/done/512x768/0175_3301/3787/`

[6]`http://www.benchathlon.net/img/todo/PCD1202/`

[7]`http://www.cs.washington.edu/research/imagedatabase/groundtruth/cherries/`

[8]`http://www.imageafter.com`

[9]`http://www.cs.washington.edu/research/imagedatabase/groundtruth/cannonbeach/`

[10]For the licence see the web page at `http://www.imageafter.com`

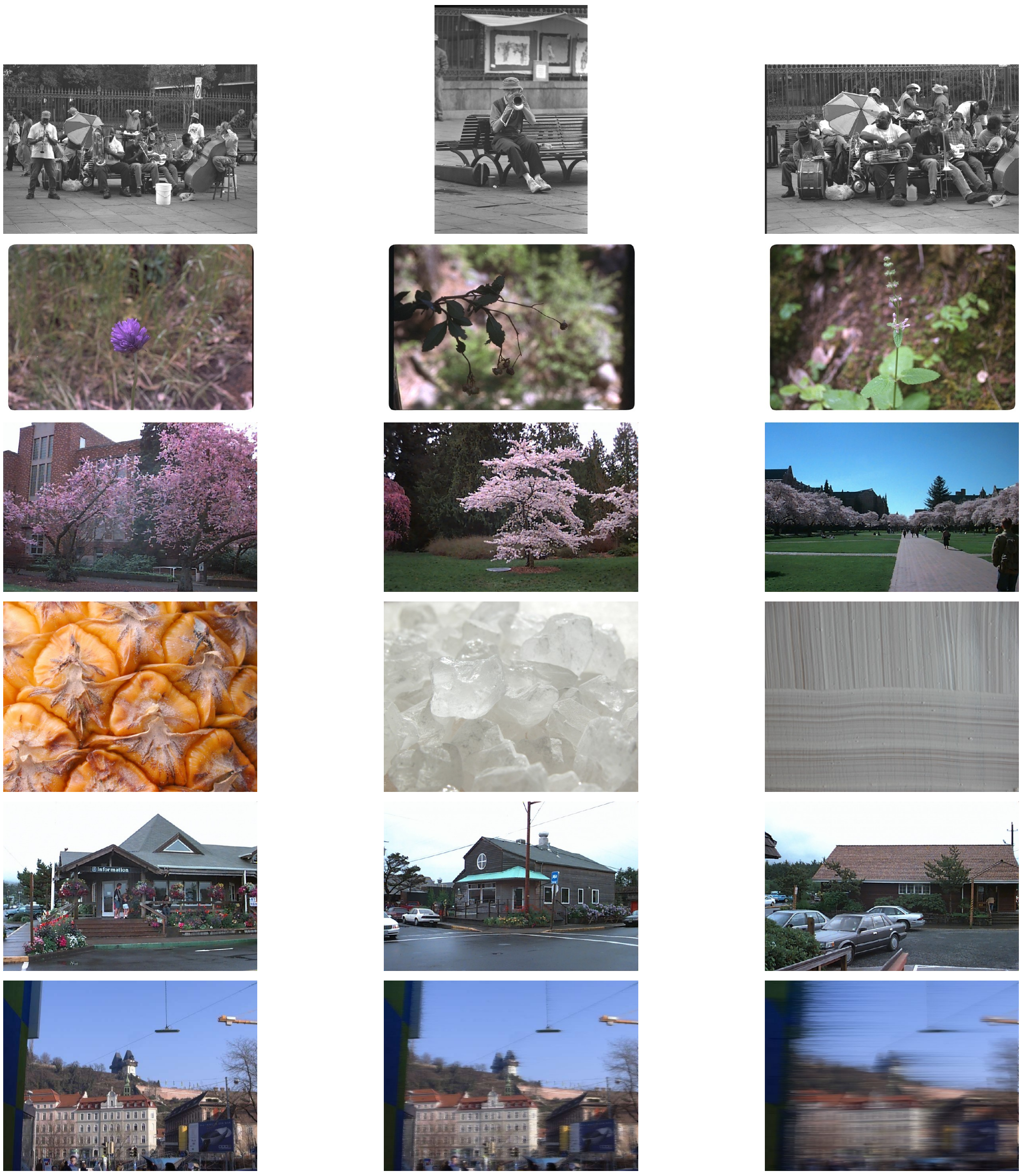


Figure 1: Example images from the domains. Each row shows 3 images from the same domain. From top to bottom: Band, Blumen, Kirschen, Textur, Urlaub, Verfälschung. For the domain Verfälschung the original image is shown left most, then two images with increasing distortion to the right.

3.3 The Benchmark

The CBIRS are benchmarked by running and evaluating a set of well defined queries on them. A query is simply an image for the search engines to look for similar images. The search is done within the images of the query domain. The reason is that building the ground truth was manageable this way, since only the images within a domain needed to be evaluated for similarity. Query images were chosen such that a reasonable number of similar results was possible.

Each CBIRS has a number of parameters which allow to adapt and modify it in various ways. Changes in those settings can improve or worsen the result for certain or even all queries. Consequently, we benchmarked different parameter settings, or *psets*, for each CBIRS. For example, for VIPER/GIFT one pset uses texture features only, another is restricted to color features. In the following, we refer to a CBIRS with a specific *pset* as *system.* We ran the benchmark on 23 psets overall. From these 20 were fully evaluated; the 3 psets of SIMPLIcity had to be excluded due the missing-images-problem explained below.

Psets are named with letters of the alphabet starting with 'b'[11]. For example, Caliph & Emir has three psets, b,c,d. Across different CBIRS psets have nothing in common, i.e., two psets c for two different CBIRS only share the letter. The only exception is pset b, which for all CBIRS is their standard setup. In the results section, the psets for each CBIRS will be described in detail.

3.4 Ground Truth

The ground truth for this benchmark has been established through a survey among 5 randomly chosen students. The students had no prior knowledge of CBIR techniques and were not familiar with the image sets (other than one could expect from fairly normal photographs). 14 queries have been defined, for each the students had to compare the query image with all images from the query's domain. They were asked to judge the similarity from 0 (no similarity) up to 4 (practically equal images)[12]. The ground truth thus consists of tuples of query image and result image which are mapped to a similarity value. This value is the mean of all similarity values from the different survey results and thus is an element of $[0,4]$.

3.5 Performance Measures

The information in this section has mainly been excerpted from [MMS$^+$01b, MMW06].

From the domain of text-based information retrieval a number of methods are known which can also be applied to content-based image retrieval. One of the simplest, but most time-consuming methods is *before-after user comparison*: A user chooses the best result from a set of different results to a predefined query. 'Best' in this context means the most relevant result.

Then, a wide variety of single valued measures exist. *Rank of the best match* measures whether the most relevant image is under the first 50 or the first 500 images retrieved. In [MMS$^+$01b] the *Normalized Average Rank* (NAR) is proposed as a performance measure for CBIR systems. We use it as basis for our own measure, which will be explained later. Popular and a standard in text retrieval are the measures *precision* and *recall*:

[11]A spread sheet has originally been used to describe the parameter sets, and since the first column 'a' was taken all sets start with 'b'.

[12]The motivation behind the 5 similarity values is simply the assumption that human judges with no prior knowledge of CBIR or familiarity with the images can fairly easy differentiate 5 (or maybe 6 or 7) similarity levels, but not more. This bears some similarity to a Likert scale, although it was not the aim to implement it.

$$precision = \frac{\text{No. relevant images retrieved}}{\text{Total No. images retrieved}},$$
$$recall = \frac{\text{No. relevant images retrieved}}{\text{Total No. relevant images in the collection}}$$

They are good indicators of the system performance, but they have to be used in conjunction, e.g. precision at 0.5 recall or the like. An entirely different method is *target testing*, where a user is given a target image to find. During the search, the number of images is recorded which the user has to examine before finding the target. The *binary preference measure* keeps track of how often non-relevant images are misplaced, i.e. before relevant images. Further single valued measures are *error rate, retrieval efficiency* and *correct and incorrect detection*. For the sake of brevity these are only listed here, for detailed explanations the reader is referred to [MMS$^+$01b].

A third category of measures can be subsumed as graphical methods. They are graphs of two measures, the most popular being *precision vs. recall graphs*. They are also a standard measure in IR and can be readily interpreted by many researchers due to their popularity. Another kind of graphs are *precision/recall vs. number of images retrieved* graphs. The drawback of these two graphical methods is their dependency on the number of relevant images. *Retrieval accuracy vs. noise* graphs are used to show the change in retrieval performance as noise is added to the query image. A number of other graph methods exist, again the reader is referred to [MMS$^+$01b] for further reading.

3.6 Worst Normalized Rank

The afore mentioned measures all use binary similarity/relevance values, i.e. only the distinction relevant/not relevant or similar/not similar is made. In contrast, our measure uses multiple similarity values and thus allows fair and accurate comparisons of different CBIRSs with respect to different queries. None of the other benchmarks allow this comparison.

To account for five degrees of similarity we developed the new measure *Worst Normalized Rank* $\widetilde{Rank}_{WRN}$, based on the NAR measure [MMS$^+$01b]. NAR is defined as

$$\widetilde{Rank} = \frac{1}{NN_R}(\sum_{i=1}^{N_R} R_i - \frac{N_R(N_R-1)}{2}) = \frac{1}{NN_R}(\sum_{i=1}^{N_R} R_i - \sum_{i=1}^{N_R} i)$$

N: no. documents total

N_R: no. relevant documents total

R_i: i-th relevant document's rank in result list

For a given query result, a ranked list of images, $\widetilde{Rank}$ essentially computes a distance between the query result and the ideal result for that query. It does so by summing up the ranks R_i of the similar images and subtracting the sum of ranks of the ideal result, which is $\sum_{i=1}^{N_R} i$ [13]. Thus, the best possible measured value is always 0, while the upper bound for the worst value is 1. The measure was chosen because it expresses CBIRS performance (for a specific query result) in a single value and is fairly easy to adapt. In the following, the term NAR will be used in a general fashion when addressing properties which hold true for both the original measure and the modified one. When referring exactly to one of them, the expressions $\widetilde{Rank}$ for the original measure and $\widetilde{Rank}_{WRN}$ for the modified measure will be used.

As a first step $\widetilde{Rank}$ is adapted to account for multiple similarity values in the following way:

[13] since the ideal result is simply the list with all relevant images at the beginning, thus having ranks from 1 to N_R

$$\widetilde{Rank}_m = \frac{1}{NN'_R}(\sum_{i=1}^{N} \frac{R_i s_i}{maxS} - \sum_{i=1}^{N} \frac{i * s_i}{maxS}) = \frac{1}{NN'_R}(\sum_{i=1}^{N'_R} \frac{R_i s_i}{maxS} - \sum_{i=1}^{N'_R} \frac{i * s_i}{maxS})$$

N**:** like above

i**:** images are sorted by similarity (most similar has lowest i), thus the ideal ranking would be $R_i = i$

s_i**:** similarity value for image i

N'_R**:** no. of relevant images, that is, all images with $s_i > 0$

$maxS$**:** maximum similarity value, five in this case

To account for multiple similarity values s_i, the ranks R_i are weighted with $\frac{s_i}{maxS} \in [0,1] : \sum_{i=1}^{N} \frac{R_i s_i}{maxS}$. The same is done for the ideal result: $\sum_{i=1}^{N} \frac{i * s_i}{maxS}$. The reasoning is that more similar images, when misplaced, should yield a larger distance from the ideal result. Note that the original measure is obtained if one chooses a ground truth with similarity values $\{0, maxS\}$: non-similar images yield $R_i * 0 = 0$, similar images $R_i * \frac{maxS}{maxS} = R_i$.

This also explains why it doesn't matter whether one sums up to N_R/N'_R or N: the non-similar images can be left out and the remaining images are exactly the N_R/N'_R similar images for $\widetilde{Rank}$ and $\widetilde{Rank}_m$ respectively. Note that the s_i need to be sorted by similarity, i.e. as stated above the highest s_i have the lowest i. Finally, the distance is normalized over the number of images in the DB and the number of similar images for the given query, N and N_R/N'_R.

$\widetilde{Rank}_m$ allows to compare CBIRS based on multiple similarity degrees. However, the measurements are not comparable across different queries or different image sets: The worst possible measurement depends on the ratio of N_R to N, i.e. it differs from query to query and image database to image database. Additionally, the worst value depends on the similarity distribution for a given query: it's a difference whether for a given query we have one very similar image and four moderately similar images or the other way round.

Therefore, as a second step, we introduce the *Worst Result Normalization*: We additionally normalize the performance measure over the worst possible result for the given query and image set. The worst result for a query is simply the reverse ideal result, i.e. the most similar images are ranked worst and appear at the end of the list while all non-similar images are ranked first. The such modified measure is then:

$$\widetilde{Rank}_{WRN} = \frac{\frac{1}{NN'_R}\sum_{i=1}^{N'_R} \frac{R_i s_i}{maxS} - \sum_{i=1}^{N'_R} \frac{i * s_i}{maxS}}{\frac{1}{NN'_R}\sum_{i=1}^{N'_R} \frac{(N-i) s_i}{maxS} - \sum_{i=1}^{N'_R} \frac{i * s_i}{maxS}} = \frac{\sum_{i=1}^{N'_R} R_i s_i - \sum_{i=1}^{N'_R} i * s_i}{\sum_{i=1}^{N'_R} (N-i) s_i - \sum_{i=1}^{N'_R} i * s_i}$$

The factor $\frac{1}{NN'_R}$ is left out, as well as $\frac{1}{maxS}$. $\widetilde{Rank}_{WRN}$ yields values in $[0,1]$ with 0 being the ideal and 1 the worst result. This makes it possible to compare the performance of systems across different queries and image sets.

Missing Images Problem

NAR requires that every similar image ($s_i > 0$) in the searched data set or image domain is ranked by the tested CBIRS, or in other words the result list must contain all similar images (*Recall* = 1). Since the CBIRS does not know beforehand which images are similar (unless it's a perfect system), all images must be returned. All search engines which have been tested provide some sort of threshold parameter to adjust the result list length, so in theory this was no problem. In practice, however, a number of systems failed to return all images, probably due to bugs. The result is that some systems did produce rather bad results: they are ranked worse than they probably would have been had they returned all images.

To deal with this problem, a number of approaches have been tested, but none of them led to a satisfying solution. In order to get comparable measures the result lists with missing images were manually completed by simply adding those images, giving them the worst possible rank. The worst possible rank is the rank of the last image in a list of all images in the domain, or simply the number of images in the domain. This is a worst case assumption punishing systems which leave out similar images rather hard. But any other solution would be based on illegal assumptions about how the CBIRS in question would have ranked the missing images, giving it an unfair edge.

The bottom line is that some systems produce results which are not really comparable. A solution might be to use an all together different performance measure than NAR which would allow to measure performance based on result lists of arbitrary length. How much this problem affected the various CBIRS is explained in subsections 4.2 through 4.6.

3.7 Comparing CBIR Systems

The performance measures introduced in the previous section allows one to measure the quality of single query results. To compare systems to one another, one needs to somehow merge these measurements into a single value representing the overall performance of the system, s.t. a ranking of the systems can be build. We used two methods, a ranking method and a scoring method.

3.7.1 Ranking

The ranking method is reminiscent of rankings used in sports competitions, where athletes have to perform in different disciplines. The single queries are analogous to the sport disciplines, the systems to the athletes. First, the systems are ranked only based on single queries. Each system then has a rank for each query or within each discipline. Now, for each system the worst and the best ranking is discarded and an average rank is computed from the rest. This average rank is then used to build an overall ranking of all systems. Systems might receive the same rank in the end, but it is unlikely. Figure 2 shows the resulting rank list.

The method can be slightly modified by simply not discarding the worst and best ranking when computing the average rank. This results in a slightly more distinct ranking (i.e. less systems are ranked equally), but essentially it doesn't differ much from the first method.

3.7.2 Scoring

The ranking produces a clear result, but doesn't say anything about the distance of systems between one another. How close, for instance, are the first three systems? We used a scoring to answer this question. A value in $[0,1]$ is computed for each system, where 0 means worst and 1 best. To be more precise, 0 represents a hypothetical system that scores lowest on all queries and 1 such a system that scores highest on all queries.

As with the ranking, in the first step, performance is evaluated separately for each query. Each system's performance on a given query, computed through $\widetilde{Rank}_{WRN}$, is mapped onto the interval $[0,1]$ using the following simple formula:

$$sc_i = \frac{r_i - b}{w - b}$$

The terms are

r_i**:** $\widetilde{Rank_{WRN}}$ result of system i for the current query

b**:** best result for the current query

w**:** worst result for the current query

sc_i**:** score of system i for the current query

The resulting scores sc_i show the performance of system i in relation to all other systems for the given query, with the worst system having $sc_i = 0$ and the best $sc_i = 1$. This preserves the relative distances from the original measures. Now, the overall score for each system is simply the mean of all per-query-scores of this system. Figure 2 shows the so computed scores for all systems.

The scoring method essentially produces the same ordering among the systems as the ranking method, yet there are some noticeable differences. Some systems which are clearly better than other systems within the scoring fall behind those systems within the ranking. The ranking essentially just counts how often a system wins or looses against the other systems, while within the scoring it is also important by how far the system wins or looses. So while one victory is easily out-weighted by lots of losses in the ranking, in scoring a clear victory might still out-weight a number of close losses. In other words, within the ranking, a system is the better the more often it ranks high with the single queries and within the scoring, a system is the better the greater the distance to the other systems is within the single queries.

4 Results

Figure 2 shows the ranking and scoring for all systems. As can be seen, the VIPER/GIFT CBIRS in its standard configuration (pset b) clearly leads the field, followed by Caliph & Emir, SIMBA and then PictureFinder and Oracle Intermedia. As mentioned in section 3.7.2 one can see that the two comparison methods slightly differ: in the ranked list, for example Caliph&Emir pset d is behind SIMBA pset e, whereas in the scoring Caliph&Emir pset d clearly has a better score than the SIMBA system.

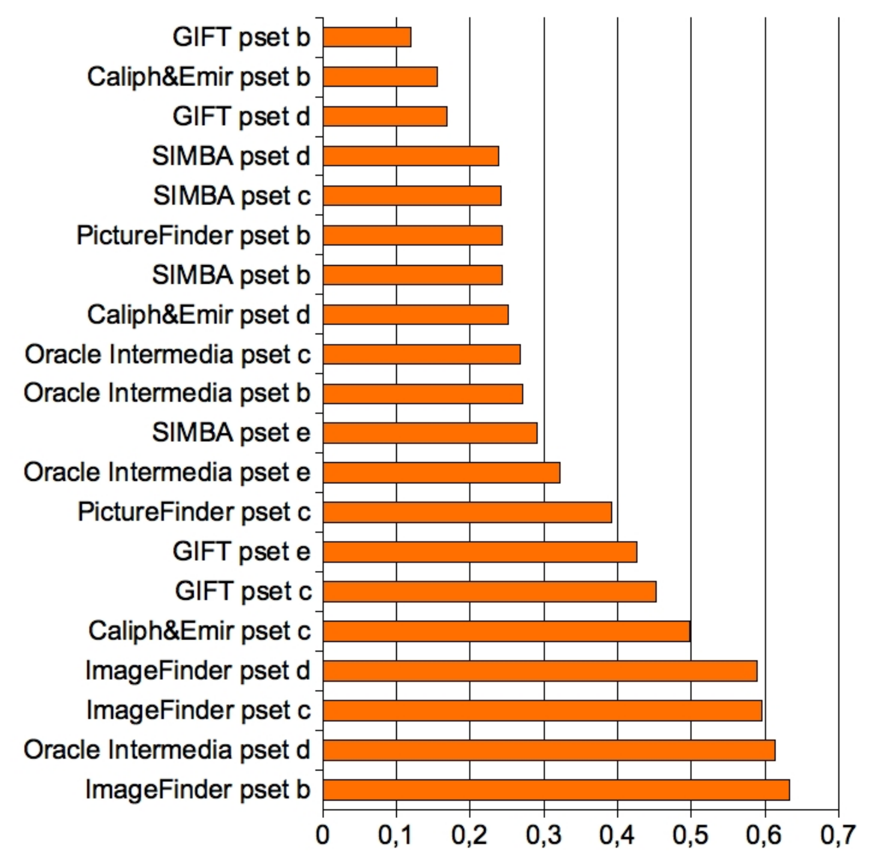
ImageFinder's performance is rather bad compared with the other systems. The most probable cause for this is that ImageFinder needs to be adjusted to the task at hand, a more detailed discussion is given in section 4.1.

Considering the different parameter sets one can see the clear tendency for the standard sets (psets b) to outperform the other sets. Only in the scoring, several CBIRS (Oracle Intermedia, ImageFinder, SIMBA) score better with a different set than the default one, but the difference is small, if not marginal. An explanation for this would be that the default settings already are an optimized parameter set for the respective CBIRS. The changes that we applied (with no explicit parameter optimization in mind) to create the other parameter sets most likely produce a less optimal set.

Table 2 shows an additional ranking of the best five systems per domain. This ranking shows that GIFT/VIPER also leads within four out of the six per domain rankings. Caliph&Emir also shows good performance scoring second within the three domains *Band*, *Textur* and *Urlaub*. The ranking method employed here is the one which also includes worst and best rankings (as explained in section 3.7.1)[14].

A comparison of all queries, based on the average results over all CBIRSs, showed that the *Blumen* queries seem to be the hardest ones. The easiest query seems to be the one from the *Verfälschung* domain, but it exhibits quite a big standard deviation.

[14]As mentioned in section 3.7.1 this method doesn't really differ much from the other ranking method. The reason for applying this method is basically that some domains only have two queries to begin with and thus no worst and best per-query ranking can be left out.



1. GIFT pset b
2. GIFT pset d
3. Caliph&Emir pset b
4. SIMBA pset b, SIMBA pset c
5. PictureFinder pset b, SIMBA pset d
6. Oracle Intermedia pset b,
7. Oracle Intermedia pset c
8. SIMBA pset e
9. Caliph&Emir pset d
10. Oracle Intermedia pset e
11. PictureFinder pset c
12. GIFT pset e
13. GIFT pset c
14. Caliph&Emir pset c
15. ImageFinder pset b
16. ImageFinder pset d
17. Oracle Intermedia pset d
18. ImageFinder pset c

Figure 2: Left: Scoring of all systems. Right: Ranked list of all systems.

	Blumen	**Urlaub**	**Kirschen**
1.	SIMPLIcity pset c	GIFT pset b, GIFT pset e	GIFT pset b
2.	Oracle Intermedia pset b, PictureFinder pset b	Caliph&Emir pset d	SIMBA pset c
3.	Oracle Intermedia pset c	SIMPLIcity pset b	SIMPLIcity pset b
4.	Oracle Intermedia pset e	GIFT pset d, SIMPLIcity pset c	GIFT pset d, SIMBA pset b, SIMBA pset d
5.	GIFT pset d	Caliph&Emir pset c	SIMPLIcity pset d
	Band	**Textur**	**Verfälschung**
1.	GIFT pset d	SIMBA pset b, SIMBA pset c	GIFT pset b
2.	Caliph&Emir pset b	Caliph&Emir pset b, SIMBA pset e	SIMPLIcity pset c
3.	PictureFinder pset b	SIMBA pset d	GIFT pset d
4.	GIFT pset b	Oracle Intermedia pset e	Oracle Intermedia pset b
5.	SIMBA pset e	Oracle Intermedia pset b	Caliph&Emir pset b, SIMBA pset b, SIMBA pset c

Table 2: The five best systems per domain.

4.1 SIMPLIcity and ImageFinder

The SIMPLIcity CBIRS implements some interesting technologies, namely the region based feature extraction, the Integrated Region Match measure and the classification of images in order to improve the search performance. In the setting pset c[15] in the *Blumen* domain, it is the best system showing a high potential. Unfortunately, for most of the other domains and psets results were distorted by missing images. Therefore we decided to exclude SIMPLIcity from the overall comparison and just to include it in the per-domain ranking (figure 2).

ImageFinder provides some kind of basic image search/recognition/match platform which a user modifies and tweaks to her specific needs. That is, compared to the others, ImageFinder is not solely meant for CBIR. Its vast numbers of parameters need to be optimized for CBIR, which was out of this works scope. Therefore, the overall results for ImageFinder are rather poor. However, for the hardest of all queries, *Blumen-10*, ImageFinder produced a good result (6th place out of 20), which shows its potential[16].

[15]For pset c of SIMPLIcity, the program *classify* was called with parameter -f 5.

[16]The result was produced with ImageFinder pset c: An unsupervised filter was used, and Images were processed with the Laplace 5 filter.

4.2 SIMBA

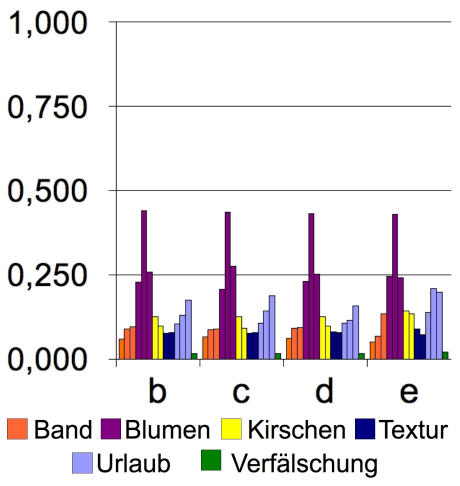


Figure 3: $\widetilde{Rank}_{WRN}$ for all SIMBA psets on all queries. Queries are grouped by pset and color coded by domain.

SIMBA uses *invariant feature histograms*, which are invariant against rotations and translations of parts of the image, e.g. objects. The feature extraction is based on kernel functions. The size of the kernel influences the size of objects SIMBA "recognizes" in images, i.e., with a big kernel, features are invariant only against rotation and translation of bigger objects [Sig02, page 24].

For SIMBA we created three psets besides the default set. Psets c and d were first tests and turned out less meaningful. Pset e provides a bigger kernel function than the default set. Specifically, kernel $(40,0),(0,80)$ was used for all color channels.

The results show that SIMBA performs well and is third in our benchmark. It shows poor within the *Blumen* domain (best pset is e with rank 14) (see table 2) and very good within the domain *Textur,* SIMBA pset b. From figure 3 one recognizes that the bigger kernel of pset e results only in minor changes in the domain *Band* with respect to the default set.

4.3 PictureFinder

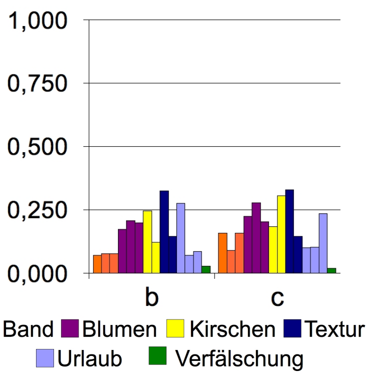


Figure 4: $\widetilde{Rank}_{WRN}$ for PictureFinder psets b and c on all queries. Queries are grouped by pset and color coded by domain. The results of the *Urlaub* domain are the 2.,3. and 4. bar counting from right.

PictureFinder models images with sets of polygons of different sizes, colors and textures. Basically, the polygons represent prominent forms in the images. PictureFinder can thus be seen as shape focussed, which might explain the good results in the *Blumen* and *Band* domains.

We modified one parameter, which determines whether the position of a polygon influences the similarity assessment or not. The default setting (pset b) was that the polygon position is important: roughly speaking, two images are more similar if they contain similar prominent forms in similar positions. Accordingly, in pset c the position was not important: only size, color and texture of polygons determined similarity.

Regarding the position as important (pset b) shows generally better results than not doing so (pset c). An exception is an outlier in domain *Urlaub*. The reason might be that for this query there exists lots of similar images (according to GT) with similar prominent forms in *different* positions. As for pset c it has been generally worse: we suspect that too many less similar images (GT), which however contain similar forms in other positions, are seen as similar to the query by PictureFinder.

Missing images in the domain *Textur* distorted these results somewhat, causing the outlier.

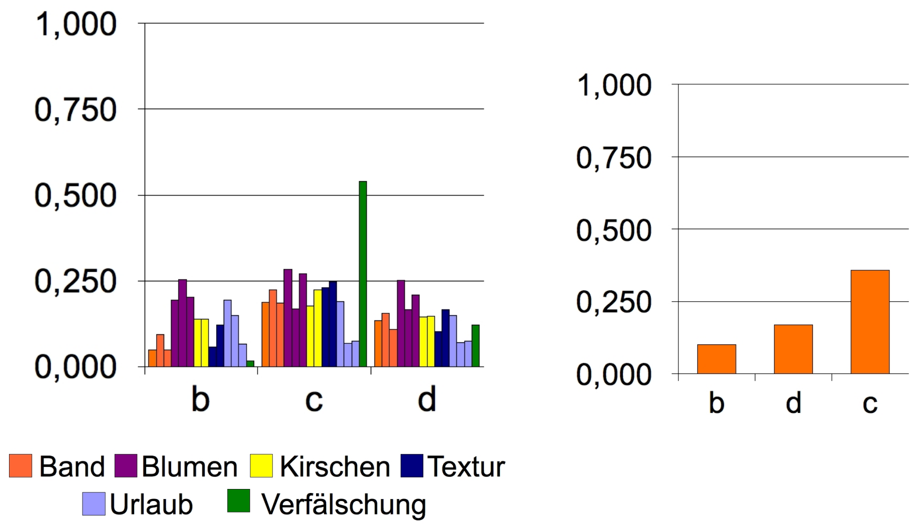


Figure 5: $\widetilde{Rank}_{WRN}$ and Scoring for all Caliph&Emir psets on all queries. On the left, queries are grouped by pset and color coded by domain.

4.4 Caliph&Emir

Caliph&Emir is a combination of two applications for managing image data. Caliph can be used to annotate the images, and Emir is the corresponding search engine. A strong feature of Caliph is its representation of annotations as semantic graphs, which greatly enhances the search. For our benchmark however, we focussed on Emir's visual similarity search. To use Emir as CBIR system, relevant features in form of MPEG-7 descriptors have to be extracted and stored as MPEG-7 documents by Caliph.

Caliph&Emir offer the three descriptors *ColorLayout*, *EdgeHistogram* and *ScalableColor* to choose from when performing a search. Unfortunately, *ScalableColor* didn't seem to work (the search didn't seem to start with this descriptor), so only *ColorLayout* and *EdgeHistogram* were used. The parameter settings were as follows: the default setting b was used for *ColorLayout*, pset c was used only for *EdgeHistogram* and pset d combines the two descriptors for the search.

The Caliph&Emir CBIRS is the second best system in this test behind VIPER/GIFT, as shown by the ranking and scoring (see figure 2). There is one outlier with pset c in the *Verfälschung* domain caused by missing images, but otherwise the results are rather balanced.

It turned out that *ColorLayout* in general is clearly better than *EdgeHistogram* or the combination of the two descriptors, see figure 5. There are however a few queries where *EdgeHistogram* beats *ColorLayout*, for example the query *Blumen-10*, shown in figure 6.

4.5 VIPER/GIFT

The design of the VIPER search mechanism, which is now part of the GIFT architecture as a plugin, borrows much from text retrieval. It uses an inverted index of up to 80 000 features, which are like terms in documents. For each image roughly 1000 features are extracted.

The numerous features can be grouped as color features and texture features. Each group in turn consists of histogram features and block or local features. The color features are based on the HSV color space, which is quantized into bins. From these, the color histogram features are extracted. Unlike other

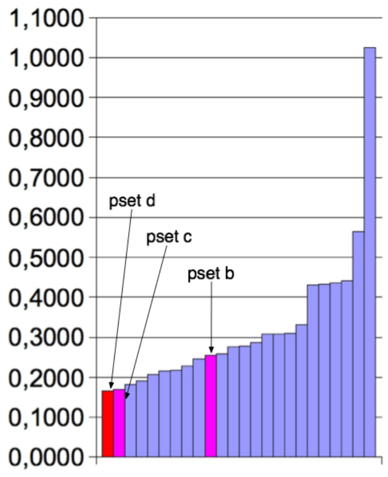


Figure 6: Query *Blumen-10* for all systems.
Caliph&Emir pset d is marked red (darker), c and b are marked light magenta (lighter).

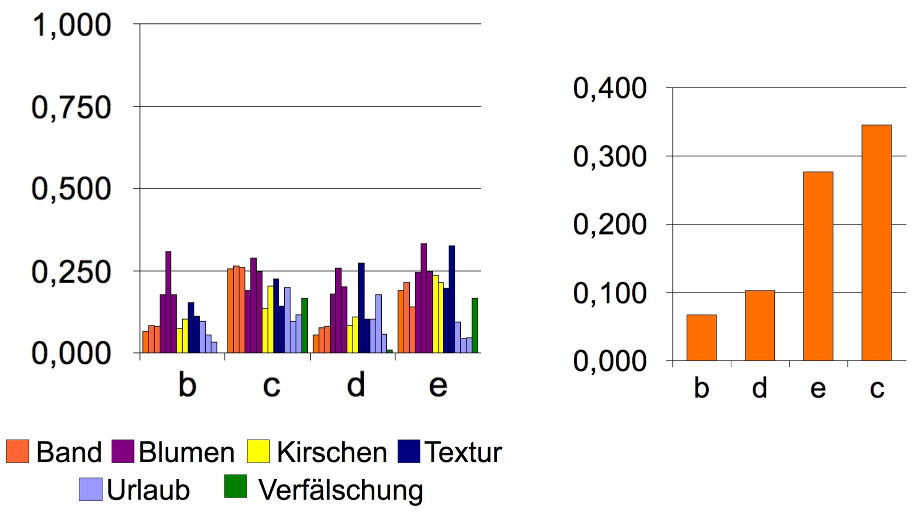


Figure 7: $\widetilde{Rank}_{WRN}$ and Scoring for all psets of VIPER/GIFT.

CBIRSs, in VIPER each bin of a histogram counts as feature. As local color features the *mode color*s[17] of small blocks are used the images are split into. The texture features are based on *Gabor filters*[18]. The filters yield local texture features which are also based on small blocks of the image. Additionally, histograms of the average filter outputs are stored as a measure for an image's overall texture qualities.

The similarity measure in VIPER is similar to methods used in text retrieval. Two methods are used for feature weighting and the computation of a similarity score: (1) classical IDF, the classical method from text retrieval. It weights and normalizes all features together. (2) *Separate Normalisation*, which weights and normalizes the different feature groups (color histogram, color blocks, Gabor histogram, Gabor blocks) separately and merges the results.

VIPER/GIFT ranks and scores best of all systems. Furthermore it ranks best in 4 of the 6 separate domain rankings. This clearly marks VIPER/GIFT as the best CBIRS in this benchmark. Note that except for the *Blumen* domain it's always VIPER/GIFT's default setting (pset b) which achieves the best results.

The different parameter sets are the following: pset b uses separate normalization and all feature sets. Pset c is like b, but with classical IDF instead of separate normalization. Pset d uses separate normalization and color blocks and histogram only (color focussed). Pset e uses separate normalization and Gabor blocks and histogram only (texture focussed).

Unfortunately this CBIRS is also affected by the missing-images-problem. The outlier with pset d in the domain *Textur* is caused by this, see figure 7 (6th bar from the right). But apart from this, the results are not influenced by this problem.

Another outlier can be seen in the texture domain *Textur* with pset e (only texture features). The reason could be that the texture images of this particular query also have strong overall color features, which are ignored in pset e. Interestingly, for both queries in domain *Textur,* the *color centered* pset d seems to outperform the *texture centered* pset e[19]. This trend is even stronger in the overall result: color features and the combination of all features are better than texture features only, see figure 7.

Our benchmark confirms a result shown in [Mül02], namely that the weighting method separate normalisation (psets b) is better than classical IDF (pset c). Classical IDF gave the texture feature groups (i.e. Gabor histogram and blocks) too much weight due to much higher numbers of features (according to [Mül02, page 50]). We conclude that VIPER/GIFT performs best employing separate normalization and color features (psets b and d). Emphasizing texture too much (directly in pset e, indirectly through classical IDF in pset c) degrades its performance.

4.6 Oracle Intermedia

Intermedia is an extension to *Oracle DB*, providing services and data types for handling media data (sounds, images, videos etc.). Specifically images and according signatures can be stored as objects in a database. A score function is used to compare images by calculating distances between them based on their signatures. This can be used to write SQL-queries with example images which produce a ranked list of similar images. Four parameters govern the image search: *color*, *texture*, *shape* and *location*. The score is computed using the *Manhattan distance*: a weighted sum of single distances for *color*, *texture* and *shape.* It remained unclear to us how *location* integrates in the score. The weights are normalized so that their sum equals 1.

The Oracle Intermedia CBIRS shows very good results with the exception of one big outlier. The parameter sets b - e are based on the four parameters and are defined as follows: pset b: (color = 0.6, texture = 0.2, shape = 0.1, location = 0.1); pset c: (color = 0.5, texture = 0.5, shape = 0, location = 0); pset d: (shape = 1.0, all others set to 0); pset e: (color = 0.33, texture = 0.33, shape = 0.34, location = 0).

[17]The most frequent color within the block.

[18]These are often used to describe the neuron layers in the visual cortex which react to orientation and frequency.

[19]Assuming that the outlier result for pset d caused by missing images would be significantly better without the missing-images-problem.

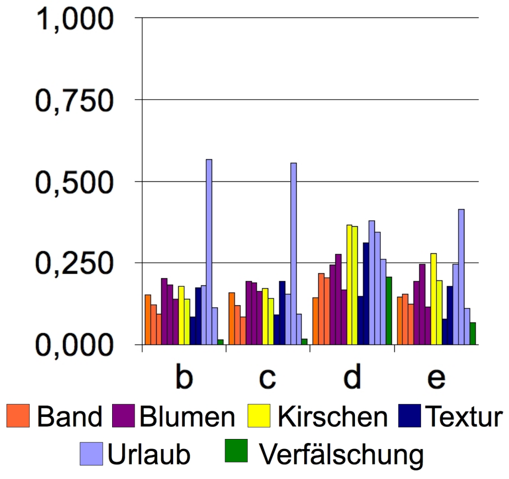


Figure 8: $\widetilde{Rank}_{WRN}$ for all psets of Oracle Intermedia

The mentioned outlier can be seen easily in the *Urlaub* domain. The causes remained unclear to us. The query (*urlaub-Image15*) doesn't seem special, none of the other CBIR systems show a similar behaviour. The outlier is less dominant in psets d and e, but still clearly visible. These parameter sets put more weight on shape search, which seems to improve performance on this query but degrade most of the others.

As mentioned the results are otherwise very good, especially in the *Blumen* domain, where Oracle Intermedia (pset b) ranks second behind SIMPLIcity. It would be interesting to see how the outlier could be improved and in how this would influence Oracle Intermedia's ranking and scoring.

Focussing on shape in pset d degrades performance practically for all queries. Also, equally using color, texture and shape is not as good as concentrating on color (pset b) or color and texture (pset c), see the ranking and scoring in figure 2. The shape-only pset d is actually much worse then the other three sets, which could mean that the shape parameter is only useful in special cases.

5 Summary and outlook

We have benchmarked seven CBIR Systems: SIMBA, SIMPLIcity, PictureFinder, ImageFinder, Caliph &Emir, VIPER/GIFT and Oracle Intermedia. Different parameter settings of the CBIRSs have been created by modifying interesting parameters.

The ground truth for the benchmarks was determined through an online survey. The image database was built from freely usable images taken from various sources. The images have been grouped roughly by similarity and forming image domains.

The systems were benchmarked by running a number of defined queries using the query by example approach. The systems were then compared to each other based on their visual retrieval accuracy. The results from the runs were measured using the custom accuracy measure $\widetilde{Rank}_{WRN}$, which we derived from *Normalized Average Rank*[MMS$^+$01b]. Based on these measurements the systems have been ranked and scored.

Results of the Benchmarks

VIPER/GIFT (with its default parameter setting) was found to be the best system in our tests. It ranks and scores best in the overall comparison, and also is first in many of the domain specific rankings. Caliph&Emir and SIMBA are second and third, respectively. We found that some systems have weak spots, like for example Oracle Intermedia.

Color features seem to work best for most systems, other features seem to be more domain specific. Different parameter settings produce significant differences, but mostly do not result in big jumps in the overall performance. From the very good performance of Caliph&Emir it can be seen that MPEG-7 descriptors are a good choice for CBIR, especially the descriptor *ColorLayout*. Changes to the default parameters tend to worsen the result, most likely because the default parameters are already optimized.

Two significant problems were encountered: First, ImageFinder's parameters obviously need to be optimized to specific domains. Second, SIMPLIcity kept crashing during the benchmark runs resulting in missing images, which distorted the results. Therefore, we decided to not include SIMPLIcity in the overall ranking and the scoring.

Outlook

Clearly the trends go towards semantic search, for example by using automatic annotation techniques, and towards Web engine capabilities (addressing scaling problems). It is also agreed that the reliance on standards for the interoperable usage is important. It seems that these are one of the most important challenges in content-based image retrieval right now: closing the semantic gap (at least a little) and agreeing on standards, for benchmarks as well as for the CBIR systems themselves. Interesting future directions would be to combine our benchmark with text retrieval benchmarks and to measure the system performance in terms of memory and CPU usage.

6 Acknowledgements

We would like to thank the authors and developers of SIMBA, SIMPLIcity, PictureFinder, ImageFinder, Caliph&Emir, VIPER/GIFT, Oracle Intermedia, for their support and many helpful thoughts and suggestions. Specifically, we would like to thank Gerd Brunner and Hans Burkhardt from the University of Freiburg for granting access to a server running SIMBA, Prof. James Z. Wang for providing SIMLIcity, Thorsten Hermes from TZI in Bremen for providing PictureFinder, the staff at Attrasoft for providing a copy of ImageFinder and Wolfgang Müller from the University of Bamberg for his helpful suggestions regarding VIPER/GIFT. Furthermore we would like to thank Markus Koskela and Jorma Laaksonen for their help and support with PicSOM, Gianluigi Ciocca and Raimondo Schettini for their efforts and help with Quicklook^2.

References

[att07] Attrasoft, company web page. `www.attrasoft.com`, May 22 2007.

[ben] Benchathlon website. `http://www.benchathlon.net/`.

[CGPF07] M. Cord, P.-H. Gosselin, and S. Philipp-Foliguet. Stochastic exploration and active learning for image retrieval. *Image and Vision Computing*, 25:14–23, 2007.

[CGS01] Gianluigi Ciocca, Isabella Gagliardi, and Raimondo Schettini. Quicklook2: An integrated multimedia system. *Journal of Visual Languages & Computing*, 12(1):81–103, 2001.

[CTB+99] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.

[DJLW08] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):60, 2008.

[DK08] Mario Döller and Harald Kosch. The MPEG-7 multimedia database system (MPEG-7 MMDB). *Journal of Systems and Software*, 81(9):1559–1580, 2008.

[DKN04] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Fire – flexible image retrieval engine: Imageclef 2004 evaluation. In *Working Notes of the CLEF Workshop*, pages 535–544, Bath, UK, September 2004.

[FCPF01] J. Fournier, M. Cord, and S. Philipp-Foliguet. Retin: A content-based image indexing and retrieval system. *Pattern Analysis and Applications Journal, Special issue on image indexation*, 4(2/3):153–173, 2001.

[FSN+01] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: the QBIC system. pages 255–264, 2001.

[GK04] M. Gabbouj and S. Kiranyaz. Audio-visual content-based multimedia indexing and retrieval - the MUVIS framework. In *Proceedings of the 6th International Conference on Digital Signal Processing and its Applications, DSPA*, pages 300–306, Moscow, Russia, March 31 - April 2 2004.

[HMH05] Th. Hermes, A. Miene, and O. Herzog. Graphical search for images by PictureFinder. *Journal of Multimedia Tools Applications*, 27(2):229–250, 2005.

[ima05] Imageclef. `http://ir.shef.ac.uk/imageclef/`, November 2005.

[KLLO00] Markus Koskela, Jorma Laaksonen, Sami Laakso, and Erkki Oja. The picsom retrieval system: Description and evaluations. In Proceedings of Challenge of Image Retrieval (CIR 2000). Brighton, UK, P.O. BOX 5400, Fin-02015 HUT, Finland, May 2000. Laboratory of Computer and Information Science, Helsinki University of Technology.

[LBK03] Mathias Lux, Jutta Becker, and Harald Krottmaier. Caliph & emir: Semantic annotation and retrieval in personal digital photo libraries. In *Proceedings of CAiSE '03 Forum at 15th Conference on Advanced Information Systems Engineering*, pages 85–89, Velden, Austria, June 16th-20th 2003.

[LTU07] Ltu technologies. `www.ltutech.com`, August 2007.

[LWW00] Jia Li, James Z. Wang, and Gio Wiederhold. IRM: Integrated region matching for image retrieval. In *Proc. ACM Multimedia*, pages 147–156, Los Angeles, CA, October 2000. ACM, ACM.

[MMS01a] Henning Müller, Wolfgang Müller, and David McG. Squire. Automated benchmarking in content-based image retrieval. *IEEE International Conference on Multimedia & Expo*, page 290, 2001.

[MMS+01b] Henning Müller, Wolfgang Müller, David McG. Squire, Stéphane Marchand-Maillet, and Thierry Pun. Performance evaluation in content-based image retrieval: overview and proposals. *Pattern Recogn. Lett.*, 22(5):593–601, 2001.

[MMW06] Stéphane Marchand-Maillet and Marcel Worring. Benchmarking image and video retrieval: an overview. In *MIR '06: Proceedings of the 8th ACM international workshop on Multimedia Information Retrieval*, pages 297–300, New York, NY, USA, 2006. ACM Press.

[Mül02] Henning Müller. *User Interaction and Performance Evaluation in Content-Based Visual Information Retrieval.* PhD thesis, l'Université de Genève, 2002.

[oim07] Oracle Intermedia. `http://download.oracle.com/docs/html/B14302_01/toc.htm`, August 2007.

[QMTM04] Till Quack, Ullrich Mönich, Lars Thiele, and B. S. Manjunath. Cortina: a system for large-scale, content-based web image retrieval. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 508–511, New York, NY, USA, 2004. ACM Press.

[Sig02] Sven Siggelkow. *Feature Histograms for Content-Based Image Retrieval.* PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2002.

[VT02] Remco C. Veltkamp and Mirla Tanase. Content-based image retrieval systems: A survey. Technical report, Department of Computing Science, Utrecht University, 2002.

[VV99] Jules Vleugels and Remco C. Veltkamp. Efficient image retrieval through vantage objects. In *Visual Information and Information Systems*, pages 575–584, 1999.

[WLW01] James Z. Wang, Jia Li, and Gio Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 23, pages 947–963, 2001.

[YHR06] A. Yavlinsky, D. Heesch, and S. Rüger. A large scale system for searching and browsing images from the world wide web. In *Proceedings of the International Conference on Image and Video Retrieval*, Lecture Notes in Computer Science 3789, pages 530–534. Springer, 2006.