UNIVERSITÉ DE GENÈVE
Département d'informatique

FACULTÉ DES SCIENCES
Professeur T. Pun

User Interaction and Performance Evaluation in Content-Based Visual Information Retrieval

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur des sciences, mention informatique

par

Henning Müller

Thèse N° xxxx

GENÈVE
2002

La Faculté des sciences, sur le préavis de Messieurs Thierry Pun, Stéphane Marchand-Maillet codirecteurs de thèse et xxx, autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 31 mai, 2002

Thèse -XXXX-

Le Doyen, Jacques WEBER

Contents

Chapter 1

Introduction and Motivations

> To see is ... to think!
>
> Salvator Dali

This chapter gives a short introduction to the world of images and in particular digital images. It also explains the main motivations to write this thesis in the field of content–based image retrieval (CBIR) and it states the achievements of this thesis in the field.

1.1 Introduction

Images and colors have always been very fascinating for humans. From cave paintings more than $30,000$ years ago to modern digital cameras, humans have always produced images for widely varying reasons, with different contents and in varying quantities. Whereas cave paintings and drawings up to the 19th century were still produced in rather small quantities, the advent of photography gave the possibility to produce "real" images of the world for a large group of people and also produced an easy way of duplicating these images. The oldest remaining photograph was taken in 1826. Starting with the 20th century, photographs became mainstream, even more so since the 1930s when color photographs became available to the public.

This can be compared with the invention of the printing press by Gutenberg in the mid 15th century that made it possible to produce copies in large quantities rather than one at a time.

Billions of holiday pictures and pictures of friends or family members are produced every year, and the numbers are still on the rise. These images are organized in albums or boxes and only the owner can tell stories about the pictures and the persons or locations on the images. Without the stories of the owner or another common background, these pictures become basically useless. Only sometimes, this information is given in a textual form in the album. For a few images of very famous places or persons only might a larger number of people know their significance. Private persons might have hundreds of images but image libraries and press or newspaper archives already have the problem of handling millions of images that are produced on a regular basis. To keep some information on all these images is even more crucial when the images are intended to be reused for illustration. This can cut down the cost of producing a new image but of course an image has to be found in the entire collection.

A recent rise in image production is due to digital cameras that have massively enlarged the number of produced digital images since the mid 1990s. As opposed to analog images, there is no need (and subsequently no necessary cost) for printing because the images can be shown on screen and thus picture taking incurs no additional cost once a camera is purchased. The problem of organizing digital images is even harder than analog ones as they cannot be put in a simple album or box. Often they end up in some directory and then have to be re–found.

By far the biggest influence on the management of images recently has been the internet, or more exactly the World Wide Web (WWW). The WWW allows the distribution of images from a local computer or a press server to millions of people throughout the world. With growth from only

50 http servers in January 1993 to millions of http servers today, more and more data has become available, and the proportion of visual data is rapidly increasing. Early web pages sometimes contained one or a few images to illustrate text, but web pages of today are becoming completely image centered. Very often, web pages contain only images and even the text is part of the images. This visual world creates a need to manage the data and find the wanted information on the web. It also creates problems, for example, with the accessibility of web pages by visually impaired people. Text can be read aloud by a machine, but what about an image? [219] treats the problem of making available image information to visually impaired users, which proves to be a hard task.

The more web pages are available, the more text search engines such as *Google*[1] or *Altavista*[2] create a challenge to be used and to find the wanted documents. With the use of an increased quantity of visual and even multimedia data (images, videos, music, sounds) the search becomes even more difficult.

The large, popular search engines (*Altavista*, *Google*, *Lycos*[3], ...) actually allow one to search for images, but usually this is done using text next to the images on a web page. So what can be done if the web page exclusively contains visual information? Can we search for images based on queries in the images themselves (same media, visual information) in the same way that we search for textual documents by using text? For example a search on Altavista for images concerning "soccer" brought up a nude woman because the owner of the page used the word soccer close to the image. The *semantic web* [15] proposed by Tim Berners Lee might change this, but will we ever have the reality that all objects available on the web are properly annotated?

1.2 Motivation

The questions asked in the preceding section comprise the main motivation behind the field of content–based image retrieval (CBIR). CBIR is one of the most active research areas since the end of the 1980s, although some systems were developed even earlier.

Similar to the *general object recognition* problem in computer vision that remains unsolved, there are several serious problems for CBIR systems (CBIRSs). One such problem is the so–called *sensory gap* [247] that describes the loss of information that occurs by recording an image, for example by occlusions of parts of the image. Another, and maybe the most difficult problem is the *semantic gap* that describes the difference between the semantic categories a user is looking for and the low level features that todays CBIRSs offer. Both problems will be described in more detail in Chapter 2

Despite these problems, improvements to the existing systems seem possible and are one of the main motivations for the work described in this thesis. From the start the idea was not to invent yet–another perfect set of features for retrieval, but to try out a new architectural approach for image retrieval. Many of the techniques used in CBIR such as relevance feedback were already used long before in text retrieval, a closely related and well studied field of research (*i.e.* relevance feedback since the 1960s in the SMART system [226]). Thus, it sounded logical to take a more formal approach in using techniques adopted from text retrieval for the image retrieval domain. This resulted in a first version of our search engine *Viper* and showed that the use of a very large feature set ($> 85,000$) was feasible and queries could be done reasonably fast with good retrieval results [262–264].

Since the first version, we have always wanted to change the feature set, but it still has not happened. Instead, other research domains within the CBIR field seemed to be more interesting with respect to improving the usability and performance of our system. The main research areas described in this thesis are the field of *user interaction*, where much of the potential to shorten the *semantic gap* lies [171, 175, 177], and in the *performance evaluation* of retrieval systems. The user interaction field also includes aspects such as reducing the time it takes to execute a query by search pruning [181, 260].

[1] `http://www.google.com/`
[2] `http://www.altavista.com/`
[3] `http://www.lycos.com/`

By far the most serious problem when starting to do research in this field was the impossibility of comparing any two CBIRSs. Every system is evaluated in a different way, using different performance measures and different image databases. Again, the field of text retrieval helped us because it is currently much more mature as a research discipline. Performance evaluation in text retrieval has been a big issue since the 1960s [40, 41, 226] and a much more formal approach has been taken to create proper document collections for benchmarking and regular evaluations [256, 257]. The cycle of Text REtrieval Conferences (TREC, [91, 287]) is now more than ten years in existence. It has been held every year since 1992 and has become the unquestioned standard in the field. Therefore, a review of the performance evaluation in CBIR was done and proposals based on the experiences of the text retrieval field were made, particularly based on TREC [178]. With the help of *MRML*, the Multimedia Retrieval Markup Language, a completely automatic evaluation including web accessibility was created [172, 173].

We can see that most of the motivations of this thesis come directly from the day to day work and experiences in the field of content–based image retrieval. Many problems in the field can be identified easily and solutions can be worked out.

1.3 Thesis overview

This first chapter gives a short introduction into the research field of CBIR and explains the main motivations for the research described in this thesis. The principal scientific contributions of the thesis are listed as well.

Chapter 2 describes the vast field of content–based visual information retrieval and many of the different aspects that are involved with it. It provides a number of references for different approaches to CBIR, details some of the history of the field and explains many of the current problems and limitations of CBIRSs. A few CBIRSs are described in more details, but the main focus is on the technologies commonly used. A separate short section is on MPEG–7, its use and its implications for the field of CBIR.

Chapter 3 explains our own CBIRS, *Viper*, including many of the features described beforehand. The main parts of the distributed architecture are described, including *MRML* that proved to be important for many other parts of this thesis like for the evaluation and the long–term learning. An evaluation of *Viper*, including several steps of feedback that use various image databases, is done as well.

Chapter 4 describes user interaction in the field of CBIR. The user interaction is not limited to the user interface but starts with methods to improve retrieval speed, so–called *search pruning*. It explains the main query paradigms as well as the problem of finding a proper starting point or description to start a visual query, called *page zero problem* [241]. One of the main contributions of this thesis, the comparison of several relevance feedback strategies, is explained and technologies for positive and negative relevance feedback are evaluated. The learning over several temporal scales and several users/databases is explained in detail and a hierarchy for learning is proposed.

Chapter 5 is the most important part of this thesis and is about the evaluation of CBIRSs. This chapter finds analogies to evaluation in the closely related field of text retrieval (TR), also called information retrieval (IR), which it is far more advanced than CBIR with respect to evaluation. A framework for benchmarking, including a completely automatic benchmark, is being presented. This benchmark includes a simple–to–use web interface and several example evaluations with a number of databases and varying sorts of relevance judgments. This chapter describes measures and databases commonly used for CBIR evaluation at the moment. With an example evaluation using the Corel Photo CDs collection it shows that evaluation is useless when it is not done properly and in a standardized way.

Chapter 6 describes two possible applications of the field of CBIR. It describes a test we performed with the IPI (Institut pour la proprieté Intellectuelle) in Bern on the retrieval of trademarks and what needs to be improved to make *Viper* a proper trademark retrieval system. It also describes some scenarios of how *GIFT* can be used for medical image retrieval. Existing publications from both fields are reviewed.

Finally, Chapter 7 presents my conclusions of this work including future research directions that might prove to be fruitful in the domain.

In addition to the main content, this thesis contains several parts that are supposed to make it easier to read. The most important part is the *index* starting at Page 181. This index may not be absolutely complete, but it at least lists the most important keywords of the different chapters and thus helps to find the corresponding sections in the main text. The *bibliography* as well cannot claim to be complete in a field such as CBIR, but it contains a number of articles that were found useful plus a number of often cited articles, whether they were used as positive or negative examples. The bibliography also contains all the articles that were produced during this research. The *glossary* does not only contain abbreviations that are used in the text, but also many commonly used abbreviations used in other articles on CBIR. Often they are assumed to be known by everybody, which can cause problems for people who are new to the field. "Abbreviationitis" is a common disease in the field of CBIR.

1.4 Scientific contributions of this thesis

Certainly, the main scientific contributions are in the two fields of *user interaction* (including search pruning and several forms of relevance feedback) and in the *evaluation* of retrieval systems. Some smaller contributions concern the *architecture* of the *Viper* system plus several tests and enhancements of the retrieval performance.

Contributions to the *Viper* system include the following:

- Ideas about the structure of the *Viper* system and the distribution of the system were developed [179]. This includes ideas about the distribution of the computation via CORBA [179].
- A better feature weighting approach based on separately weighting the different feature groups was implemented. Such a weighting massively reduced the influence of very frequent small textures that caused problems in earlier *Viper* versions. This new weighting was first used in [260] and in all publications afterwards.

To improve on the usability of the system and especially to make it more interactive, several ways of *search pruning* were implemented to speed up the system response time:

- Part of the weighting factor were pre–calculated for a computational gain to speed up the query process.
- A parallel version of *Viper* was tested and implemented to produce methods for parallelization of the computation using PVM [181].
- Search pruning was tested in many different ways to enhance the usability and interactivity of *Viper* [181, 260]. This includes the evaluation of the information content of frequent features and how search pruning affects the retrieval performance of a CBIRS.

Several other aspects in the field of *user interaction* or more precisely relevance feedback were analyzed within the framework of this thesis:

- An analysis of different strategies for relevance feedback was performed to compare their performances [177]. This can *i.e.* be used to automatically help novice users by adding negative feedback to a feedback query to receive better results. It can also be used to automatically generate relevance feedback for evaluating the system performance [172].
- Relevance feedback with separately weighting positive and negative documents in a query as proposed by Rocchio [218] was implemented. This helps to avoid the previous problems with the use of too much negative feedback in a query. Too much negative feedback can also eliminate good and wanted features from a query [177] by a negative weighting.

- Analysis of large logfiles of user interaction data and how we can improve the system performance by using this information [175].
- Importance factors were derived for features based on the interaction log files we stored, [171, 175]. This long–term learning can massively improve the performance of a system. A comparison with the very similar problem of market basket analysis (MBA) is done and fast algorithms to calculate association rules are used.

The main contributions of the thesis are certainly in the field of retrieval system *evaluation and benchmarking*:

- Several user experiments were performed to gain relevance judgments of real users to evaluate the *Viper* system, used e.g. in [185, 261, 262, 264].
- Methods to evaluate the performance of CBIRSs were reviewed and compared [178]. Proposals were made for the evaluation of CBIRSs based on evaluation in text retrieval, a much more mature field with respect to evaluation. The Text REtrieval Conference (TREC) especially inspired this research.
- Automatic scripts were created to evaluate retrieval systems based on *MRML*. This includes the creation of several steps of relevance feedback and the evaluation of the performance with relevance feedback [172].
- A web–based benchmarking framework[4] was developed to support the evaluation separate from the location of a system [173, 174]. Such a permanently available benchmark server can be a good complement to a benchmarking event, where research groups can compare systems and techniques.
- The *Benchathlon* [5] benchmarking effort for CBIR at the SPIE Photonics West conference was supported and proposals were made for a technical infrastructure.
- The creation of a freely available image database for evaluating CBIRSs[6] was supported by making available several groups of images.
- Tests with the Corel database were performed to show that evaluations using different subsets of the Corel Photo CDs can lead to very differing results [170]. This underlines the need for an image database for evaluation, including query tasks and ground truth data.

Other articles written on this project include [169, 176, 180, 186–190].

[4] http://viper.unige.ch/evaluation/
[5] http://www.benchathlon.net/
[6] http://www.cs.washington.edu/research/imagedatabase/groundtruth/

Figure 1.1: There are many challenges in image retrieval.

Chapter 2

Content–Based Visual Information Retrieval

> Photographs are as much an interpretation of the world as paintings and drawings are.
>
> Susan Sontag

Content–based image retrieval (CBIR) or, more generally said, content–based visual information retrieval is (in the digital imaging domain) one of the most active research areas of the 1990s and the early 21st century. When exactly it started is hard to reveal, but one of the earliest conferences might have been the *Conference on Database Techniques for pictorial applications* held in Florence in 1979 [17]. Early citations also include [35] in 1980 which already speaks of *query–by–pictorial–example*, although the concept of *query by example* (QBE) is even older in the database community ([312], 1975). In 1981, [36] reviews pictorial applications for databases. A very early overview of image database applications with more than a hundred references from 1983 can be found in [272]. Maybe the first use of the term "content–based image retrieval" occurred in [116] by Kato in 1992.

Some review articles of the CBIR field from different years include [87], which was written as an introduction to a special issue of IEEE Computer on CBIR (1995). Also in 1995, Enser wrote a review on image databases and pictorial information retrieval [69].

The following articles are a very good resource about CBIR with often more than a hundred references each. In [4] (1996), an introduction to image retrieval with a long part on video retrieval is given. [212] (1997) creates a good overview of the techniques used for the indexing of images including annotations and textual search in image databases. Huang *et al.* present the past, present and future of CBIR [220] (1998). [57] gives a good introduction to the basic techniques in the field, and [27] explains various techniques in much more detail. Several PhD theses gives as well a good description of the state of the art [184, 248].

Smeulders *et al.* give a very good and comprehensive overview of the more recent developments in the field in [247], including more than 200 references. Eakins and Graham tried in [66] to give an almost exhaustive list with existing CBIRSs and the technologies used in them. It was planned to compare the performance of all the systems that they could get as executable programs, but unfortunately in [282] only a few of the systems were actually compared based on a few example queries and no in depth evaluation of any system was performed. A list with links to retrieval systems can be found at [1]. [296] is another early review article on image retrieval explaining many of the indexing techniques such as *R–trees*, *K–D–trees* and so on.

A short German introduction to image retrieval is given in [180] and a longer more detailed description of the CBIR field can be found in [168].

[1] `http://viper.unige.ch/other_systems/`

2.1 Introduction

This section gives an introduction to the basics that are important to understand *content–based visual information retrieval*, including the basics of the human perception. Our visual system and our perception should be at the center of image retrieval research because what we really would like to find using a CBIRS are images that are similar for a human and according to his or her perception.

2.1.1 What is multimedia?

Although this thesis concentrates on images and image retrieval, the techniques can be generalized to a variety of media retrieval systems, and even mixed multimedia retrieval. So, what do we really understand as multimedia? Normally it is, as the name says, the connection of several media at the same time. Thus a video could be regarded as multimedia as it does contain a series of images and synchronized with the images normally sound. This list of media is sorted hierarchically:

- textual,
- visual,
  - still: images;
    - picture;
    - clip art;
    - vector graphics;
    - others;
  - animated: video, animations;
- audio,
  - speech;
    - reading;
    - conversation;
  - music;
  - generic (sound);
- combination,
  - web pages;
    - static;
    - dynamic (SMIL, Synchronized Multimedia Integration Language);
- other.
  - *e.g.* gesture information;

Almost all the examples in this and the following sections are based on one media, normally images. Still, often the term multimedia or visual information is used in this context as many of the techniques presented can be generalized to several media without major changes, only the features to extract for each media can be very different. True multimedia retrieval might still need some time to take off, but it will and, thus, solutions have to be developed that leave at least room for using different media.

The W3C has developed a standard language for synchronized display of several media on screen, SMIL[2] (Synchronized Multimedia Integration Language). This HTML–like language makes it easy to create simple web pages with media that need synchronization such as streaming audio and video or animations.

More about the retrieval of audio and videos can be read in Section 2.5.2.

[2] `http://www.w3c.org/AudioVideo/`

2.1.2 At the crossing of many sciences

Figure 2.1 shows how many different fields of science have an influence on the field of content–based visual information retrieval. While *image processing* to preprocess images and extract features and *computer vision* or *image analysis* to analyze the content of images are the most obvious fields, some others might sound surprising. *Medicine*, for example, is needed to learn about the physiology of the human visual system which is extremely important for CBIR.

When we go more into human perception, it gets to the *psychologists* who experimented much with how we perceive things in the real–world. This started very early (prior to World War I) with the so–called *Gestalt theory* [121, 122], and continued on with many other theories on *visual perception* and *perceptual grouping* [16]. Finally, we would like to retrieve images that humans perceive as being similar. Psychologists also had a major influence on the field of *human–computer interaction* [194, 209] that deals with usability issues and interfaces. Especially the interaction speed is extremely important for all interactive programs such as CBIRSs, as detailed in Section 4.1.

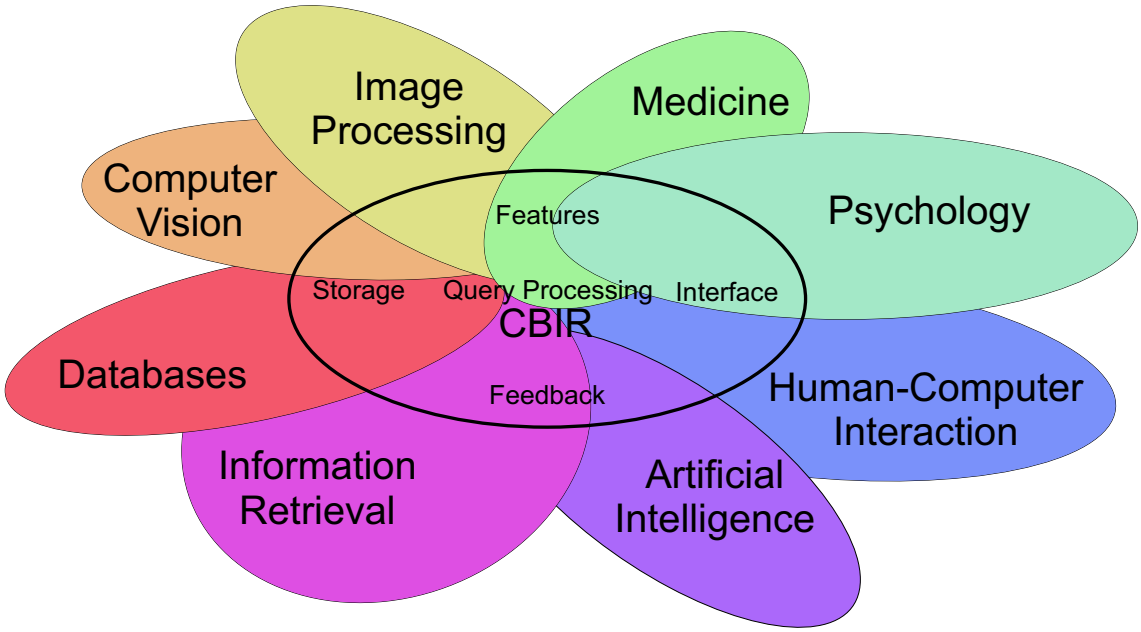


Figure 2.1: Content–Based Image Retrieval at the crossing of many different sciences.

The more computer science–centered fields start with the *artificial intelligence* (AI) field which is often used when analyzing the user behavior and learning from the feedback of the users. We use several AI techniques, for example to calculate better features weightings from user interaction logfiles (Section 4.6.1). *Database management* basically deals with storing the data such as features or meta information and with efficient ways to access the stored characteristics. The techniques applied often depend on the kind of features used and on the organization of the feature space.

The most similar field to content–based image retrieval is certainly the field of *information* or *text retrieval* as it addresses the same kind of problems only with a different media. Many of the techniques used in our retrieval system *Viper* are actually based on technologies developed originally for the text retrieval field.

Of course, there are also other more "more classical" computer science fields being involved in the development of a CBIRS, such as *software engineering* to develop a maintainable system or *data mining* as a subdiscipline of AI and of course *pattern recognition* to find interesting structures in images and also to find patterns in user interaction data.

2.1.3 Goals and problems in visual information retrieval

The main goal of a content–based image retrieval system is certainly to find an image or a set of images that a user is searching for within an image database or image collection. In libraries and many other digital image archives, this search is mostly done using keywords that the images are annotated with. In contrast to this, the field of content–based retrieval is aiming at achieving this search without the use of any annotations to the images and solely by analyzing the image content or, better to say, using the visual characteristics that can be extracted from the image automatically. Often, a mixture of textual and visual characteristics is used to improve the results of retrieval that would be obtained by using only visual features and to have at least some semantics explaining the images [241, 294]. Still, annotating images with text is a strongly subjective task and it is very hard to get consistent annotations for large image databases.

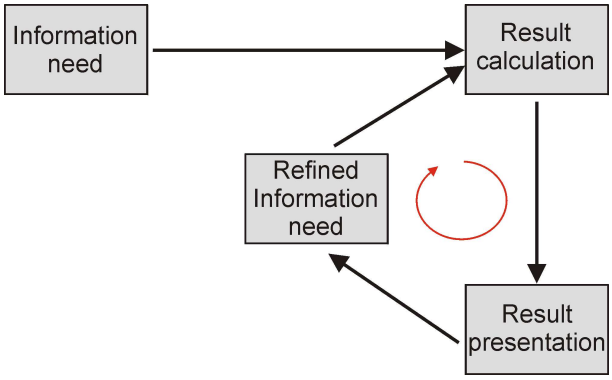


Figure 2.2: The loop of visual information retrieval, starting with an information need.

Figure 2.2 shows the main functionality of a retrieval system: a user has an information need that is expressed in some way to the system. The system then tries to respond to the information need and presents to the user the system's reply. Based on the returned images, the user can then refine the information need to the system and start another query.

Already judging the relevance of texts has a strong *user subjectiveness* [286], but in the search for images this subjectiveness might be even stronger [165, 265, 266] and also depends on the user's search task [274].

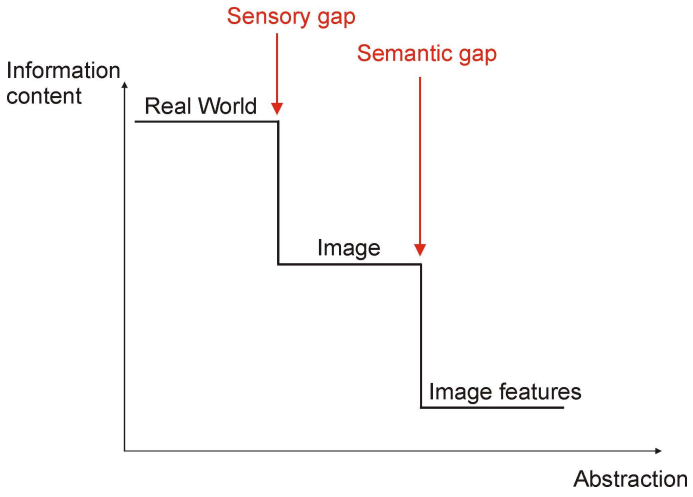


Figure 2.3: The sensory and the semantic gap.

Figure 2.3 shows two of the most prominent problems in image retrieval, the *sensory gap* and the *semantic gap*. The *sensory gap*, described in [247], refers to the information loss already

produced by the imperfectness of the image capturing device. When producing an image (with a camera or another capturing device), we lose information that was present in the real–world. This can be due to missing details because of a too low resolution, or to partially occluded objects. The *semantic gap* describes the difference in information between the actual image that contains many semantics like objects present that are easily perceivable for a human, and the presentation of this image for the retrieval task with the help of low level features that are commonly used by CBIRSs.

Also important is to analyze how image archive users actually search and how they can best be supported with tools for the visual information search. Studies like [151] where journalists practices of searching for images were analyzed are thus very important as they can reveal several important factors of search behavior:

- Journalists try to find acceptable images, not necessarily the "best" one, where time constraints may be an issue.
- Browsing plays an extremely important role in the process of finding images.
- Often a journalist has many ideas in mind for an illustration, not necessarily tied to the subject of the article but often in a more abstract or symbolic sense.
- Many queries are for an image of a certain person (portrait) or a certain object.
- It is important how old a photo is or how recently it appeared in another publication.
- Often the feelings invoked by an image play an important role.

This behavior might be similar in other areas where searches for images are performed such as in library sciences, or in image collections of museums, where large visual archives exist. Often these images are annotated and can be searched based on the annotation [113, 240, 243]. It is important that these fields of image annotation and grouping and the content–based image search do not stay separated, but that the technologies of both fields are used together to have adapted tools for very varying information needs.

2.2 Visual perception

Much has been written about the human visual physiology and the human visual perception. This chapter only gives a short introduction into the field and rather points to interesting publications.

An interesting example for the complex effects of our perception can be seen in Figure 2.4. This figure demonstrates that there are several processes happening when we see and perceive images. The image is not just a flat, one–to–one copy of the real–world in the brain ("movie screen" analogy). Some of the explanations and many other examples for peculiar effects of our perception can be found at[3].

2.2.1 The human visual system

A very good introduction to the human visual system is given in [239], and especially the section [86] gives a very detailed description to physiological processes regarding the human visual system. Other articles to read include [289], that explains tests that revealed the sensitivity of the receptors in our eye.

Color

The human eye is a very complex system and the retina is actually already part of the brain. It is a complex neural network that contains several layers and receptors that are important for our vision. In general, we have *rods* and *cones* as visual receptors. The rods are basically used for

[3] `http://www.yorku.ca/eye/`

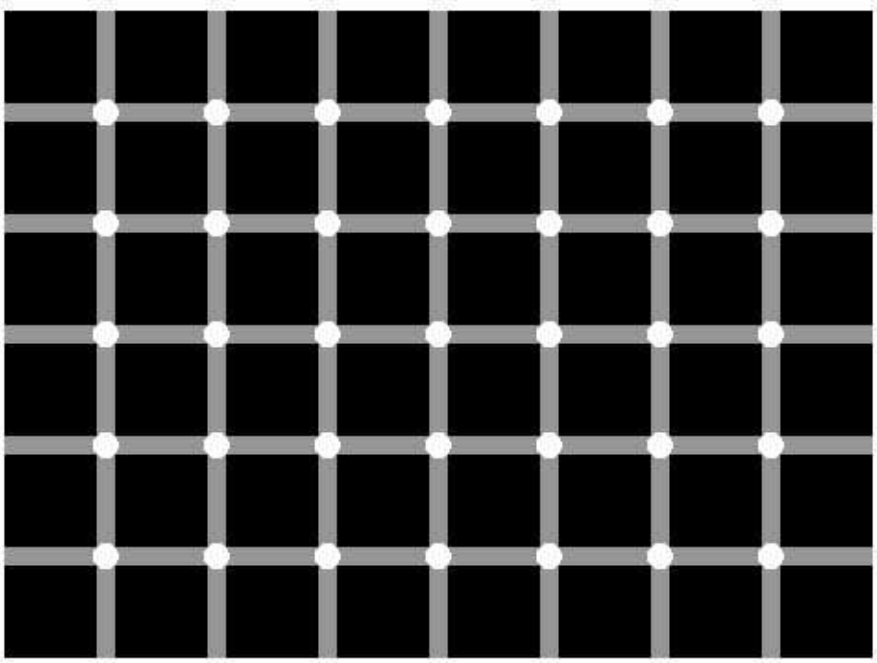


Figure 2.4: Try to count the number of white dots present in the image.

night vision, when there is only little light available (skotopic vision). For daylight seeing (photopic vision) we have three types of cones that are sensitive to different wavelengths, corresponding to the three colors red, green and blue. Sensitivity curves for the different colors and receptors can be found in [289].

As the retina already contains several layers of neurons before the actual visual receptors can be activated, it is impossible even with an infinitely small light source, to activate a single receptor. The electrical signal sent to the brain by the receptors already contains the activation of an area on the retina. Through this process several effects have been recognized such as the enhancement of edges in images. This effect is well demonstrated in Figure 2.5. If we take a look at the image for a short while, we get the impression that the white areas in between four of the black blocks start to have a more grayish color whereas the parts that are directly next to the black boxes seem to be white.

The concentration of the cones and rods in different parts of the retina of humans and other mammals depends on whether they are more active at daytime or at nighttime. For humans, the central *fovea* contains the highest concentration of cones and this areas also occupies an unproportionally large area in the striate cortex. Details can be read in [86].

Despite the fact that we have receptors for the three colors red, green and blue, our perception of colors can better be modeled in other color spaces than RGB (red, green, blue) as explained in Section 2.3.2. Most likely, none of us would describe a given color by its contents of red, green and blue. We rather tend to use characteristics such as hue, saturation, brightness, chroma or luminance to describe the properties of colors we perceive.

Edges, texture

The visual nerves transport the activation patterns of the retina to the brain. Here, we have have a number of receptors for very specialized patterns, for example, for color differences or for light/dark differences. There are also specialized columns of neuron cells in the cortex to detect edges at certain very exact directions. This edge information can further be used to recognize objects.

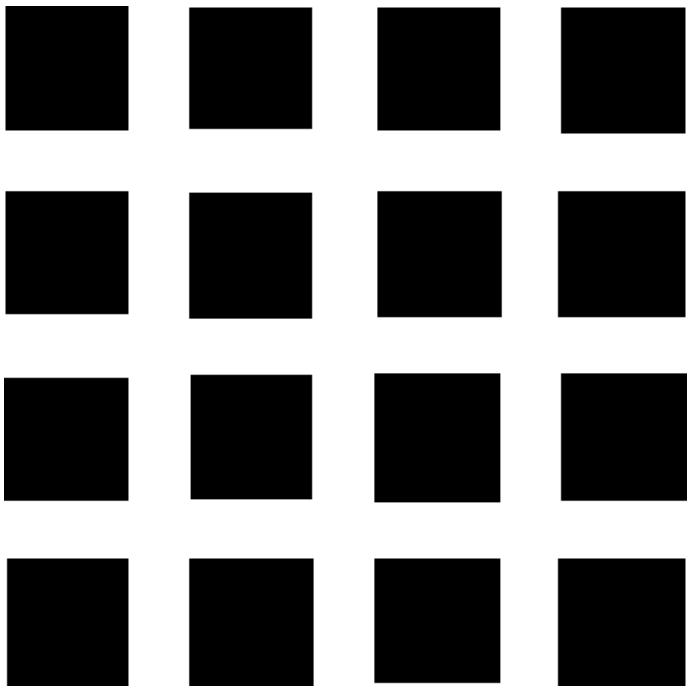


Figure 2.5: Image to demonstrate the reinforcement of areas with high contrast.

Sometimes Gabor filters are mentioned to correspond roughly to the orientation and frequency–selective properties in the visual cortex [53, 54].

2.2.2 Preattentive similarity

Pre–attentive similarity describes the similarity that we perceive when we look at an image for a very short time without having the time to perceive all the details and recognize the objects present. In [231], this effect has been demonstrated strikingly with an example image and the images in Figure 2.6 show the same effect. When we take a look at the images upside down for a short moment, we can recognize that they are normal faces and everything seems to be all right with them. When we take a look at the images with the correct orientation, we realize quickly that the lips and the eyes of one of the images are not correct, they are simply turned upside down. Our visual system has used these three areas for trying to recognize the face and we are thus using *attentive* vision.

Biedermann [16] exposed images with parts and components (also partly occluded) for different time intervals to test persons to find out which of the parts are actually recognized at which speed and accuracy. These tests were subsequently used to define a set of basic objects that we can perceive extremely quickly and that seems to be natural for our recognition. Especially the quality of recognition is very accurate, whereas quantitative measures are much harder to obtain from our brain.

2.2.3 Perceptual grouping

Much has been written about perceptual grouping and many experiments with humans have been done over the last 80 years. Theories from the early Gestaltists [121, 122] to Hebb [95], Gibson [81], Leyton [138], Marr [153], Koenderink [120], Lowe [144] and others have described how we might combine basic elements to forms and how three dimensional objects might be constructed and recognized in our brain.

Figure 2.7 shows simple examples for the way parts of a whole are grouped together. On the left hand side, all the points are of same size and same distance, so they are all grouped together.

(a) Nico normal (b) Nico not normal

Figure 2.6: Images to show the effect of pre–attentive similarity.

The next image contains two different symbols, but at the same distance. As a results, the same symbols are grouped together and basically lines are seen. The last two images show the same symbols, but at different distances, and it can be seen that they appear rather like lines of objects.

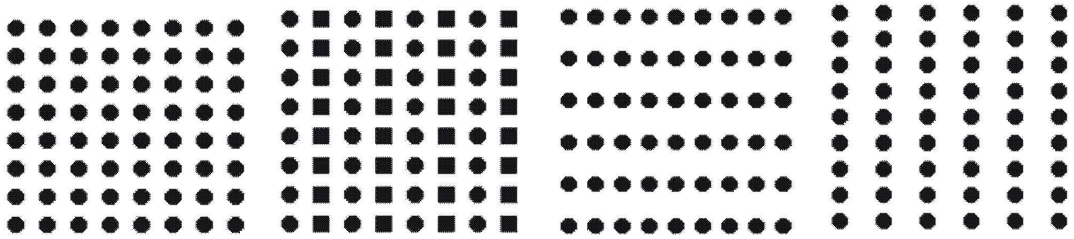


Figure 2.7: Different examples to show how humans group objects together.

Another very famous example is shown in Figure 2.8. What do you see in the image, a vase or two faces looking at each other? The distances between the two parts on the sides (faces) can influence, what we actually see.

A very interesting work on visual perception that introduces the notion of basic forms that we can perceive extremely fast is given by Biedermann [16]. He calls the basic forms *geons* and relates them to *phonems* in speech recognition. He shows in the experiments that these basic forms can be recognized much quicker than other forms, normally constructed out of these basic forms. He also describes characteristics for grouping forms together, that are used in [65, 67, 84] for analyzing images to perform trademark retrieval. Visual parts of the trademarks are grouped together into families with similar characteristics.

Image components should be represented by their boundaries and grouped into boundary fam-

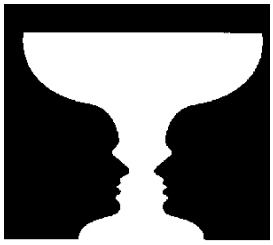


Figure 2.8: An image that contains two different contents, where we can always see one at a time. Foreground and background are exchangeable.

ilies, meeting one or more of the following conditions:

- boundaries must be in close physical proximity;
- significant lengths of their boundary must be collinear or parallel;
- significant lengths of their boundaries can be derived from concentric arcs;
- boundaries exhibit some degree of symmetry or shape similarity;
- boundaries exhibit some similarity in color or texture.

Based on these conditions parts of an image can be grouped together, for example for trademark retrieval.

2.3 Components of a CBIRS

Retrieval systems differ largely in their structure and in the techniques they use but they usually contain a number of common components. The general structure thus derived can be seen in Figure 2.9. Initially, a *user interface* allows to query the system. Sometimes the user interface is directly connected with the system, but often a communication between the user interface and the actual search engine is done, for example, with using sockets. User interfaces are studied in Section 4.4.

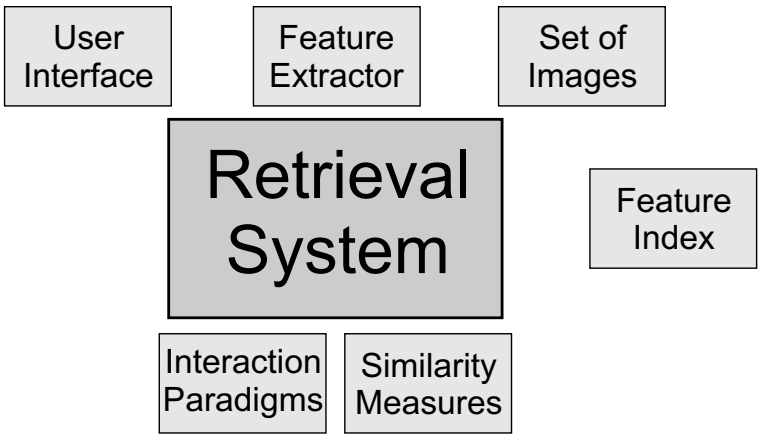


Figure 2.9: Components of a retrieval system.

Most systems work with closed *image databases* that are indexed offline by *extracting* a number of *features* from all the images in the database. These features can then be stored in a *feature index* that can be a relational database or any other index structure such as an inverted file. Often, for simplicity reasons, the index is completely stored in main memory which can limit scalability. The

feature extraction of all features or a subset of features can also be started when a query is done with an image that is submitted as query that has not been indexed beforehand (*external* image).

The biggest variations between retrieval systems are in the way that the searching is performed. There are different strategies for searching, several *similarity measures*, *image features* and *query paradigms*. Other parts of an indexing system such as the preprocessing of the images and the features used are also explained in the following sections.

2.3.1 Preprocessing of the visual document

Often the images are preprocessed or have to be preprocessed to gain more meaningful information or to gain the information in an easier way. This preprocessing can be a necessary step to feature extraction when, for example, image regions are used as the base for the querying [26]. These regions need to be extracted before any features can be calculated to characterize every single region. For points of interest, this is also the case, as the points of interest, or better, their spatial relationships are sometimes used as query features of the system [238, 311].

Image transformations

A common technique for preprocessing is *denoising* applied to images where there are known errors, for example, in the image capturing devices. Sometimes the statistical properties of these errors are known and they can be modeled well. If a good model for the errors exists, they can often be removed effectively from the images.

For color images sometimes a *normalization* of the colors might lead to better results, for example, when the images do not look very bright. Such a color normalization was used for the TSR 2500 database, as described in Section 5.2.3.

Image segmentation

In [247], a classification of different segmentation methods was done into the following groups:

- *Partitioning*, where the image is divided into fixed divisions regardless of the data.
- *Sign detection*, where an image is searched for the location of a certain geometric shape with a certain semantic meaning.
- *Weak segmentation*, where the image is divided into regions which are internally homogeneous according to some criterion.
- *Strong segmentation*, where the image is divided into objects of the real–world.

Clearly, *strong segmentation* is rarely possible and is restricted to very limited domains of images. Most programs that do use image regions as queries actually use *weak segmentation*, where regions with similar properties are extracted from the image. One of the most well–known programs using regions is Blobworld [25] that uses regions and their spatial relationships for the query and is described in more detail in Section 2.4.5. A good approach for region extraction is used in [298], where a segmentation algorithm based on texture and color features is applied. The segmentation algorithm focuses on local distributions of features and their spatial stability.

Another segmentation method is described in [215], where the image is divided into small, overlapping subregions that are approximately hexagons. These subregions are then merged based on their statistical properties. In [252], homogenous color regions are extracted by back projection of certain color sets from the histogram. These homogenous regions can then be used for the query process.

Partitioning of the images in fixed regions is computationally much easier but contains much less information. Such a technique is used in the systems described in [264, 294].

Finding salient regions or points of interest

There are various strategies for finding *salient regions* or *points of interest*. In limited domains it is often assumed that the most important object in an image will actually be in the middle of the image and will use a significantly large part of the space.

In [112], a method to learn the importance of presegmented regions in images is proposed. The approach is demonstrated with massive learning where a query is executed for each one of the 8,600 images in the database and for the first 100 images it is marked whether they are relevant or not for the learning algorithm. Based on these data, similar regions in images marked as relevant get a higher weighting. The evaluation is then done on the same dataset as the learning phase, which might lead to overly good results.

In [238], *points of interest* are extracted from images and the spatial relationships between the points are used for retrieval. In general, these points of interest correspond to points in the image where there is a corner or another salient feature. *Saliency* has also been defined as the points in an image that survive longest when gradually blurring the image (Similar to scale space theory, [18, 165]).

2.3.2 Features and characteristics extracted from images

Although the pixels of an image themselves could be used for the retrieval task, this is rarely done as the pixels itself do not contain enough information for most query tasks. Normally, more meaningful features are extracted from the images to perform querying.

[66] classifies visual descriptors in three groups:

- primitive features (color, shape, texture);
- derived or logical features (objects);
- abstract attributes (feelings evoked).

In [107], another classification of visual descriptors was done. This classification resulted in the feature pyramid shown in Figure 2.10. Although this classification was meant to be for textual descriptors of images, part of it can be reused for the visual descriptors automatically extracted from images.

In general, no semantic features are extracted automatically by CBIRSs. Most of the features are actually global distributions and local structure features that are very high up in the pyramid (Figure 2.10). Attempts to do *cognitive retrieval* [61] usually rely heavily on the user input for classification and annotation. The pyramid shows well that the deeper you go down in the pyramid (semantics), the more knowledge is necessary to actually obtain the features. The features also become more subjective, the more you go down in the pyramid.

Invariances

Human observers recognize objects and patterns despite changes in orientation, scale or lighting conditions. We take this fact for granted but it is something that is very hard to achieve with computer–based methods. When using computers, these invariances often, but not always, produce a loss in information. We thus would like to produce features that are invariant to certain changes in the image but still contain as much information as possible.

In general, invariances can be constructed with respect to geometrical changes of the image like changes in *scale*, *shifting* or *rotation* of the image, but also with respect to different *lighting conditions* of the image. Other invariances can concern the *viewpoint* [311], or certain *deformations* of the object in the image [183]. Sometimes the image coordinates are tranformed into *polar* or *log–polar coordinates* to ease the creation of invariances [159].

A simple method to gain invariance with respect to scale, shift and rotation is the Fourier Mellin transform that has been employed for image retrieval in [160]. An overview of invariant pattern recognition techniques is given in [258] and in [143] a survey on shape analysis techniques including invariance aspects is done. A CBIRS with invariances with respect to color and shape is explained in [79] and a detailed article on several aspects of color invariance can be found in [76].

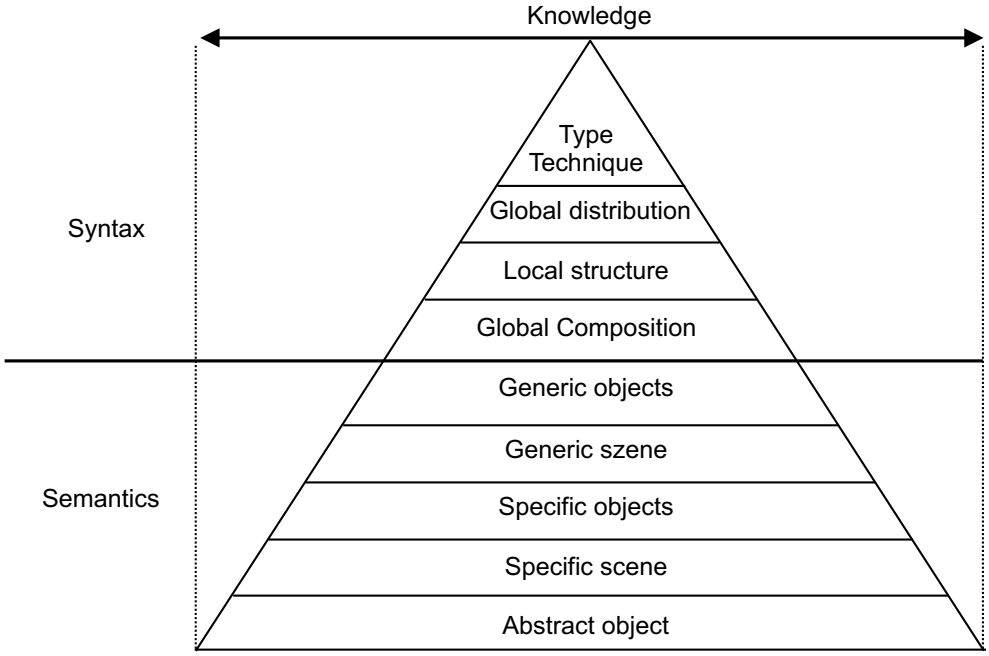


Figure 2.10: Classification pyramid for visual descriptors.

Low and lower level features

Based on the feature pyramid shown in Figure 2.10 we can create a classification of the features commonly used in image retrieval and, as well, some features that are attempted to be used, such as feelings [127]. In general, the features used for retrieval are very low level. The following grouping of features for retrieval can be done:

- *lower*: global color, global texture, such as a color or orientation histogram;
- *low*: local color, local texture or shape features;
- *middle*: shapes of segmented forms, that might correspond to real–world objects (weak segmentation);
- *high (semantic)*: real objects in the image, persons, places, no feelings, but including partly occluded objects (strong segmentation);
- *higher*: invoked feelings, semantics that are very personal to a certain user.

Normally, when higher level features are used for image retrieval [127], the images are still automatically indexed with low level features but then, a mapping to the higher level features is tried through interaction with the user.

Global *vs.* local features

A very important factor for features is whether they are extracted on a global or local scale. Global features are easy to calculate and do not need any preprocessing of the image, whereas local features can be much more specific for certain search tasks. Local features need some sort of preprocessing, at least a partitioning of the image in regions. The comparisons of features used in *Viper* (Section 3.3.4) show that the global features alone perform better than the local features obtained from partitioned regions alone. Still, the study in [174] shows that a system with local features performs better than a simple histogram interaction, especially after feedback.

Local features obtained from segmented regions are expected to lead to much better results than from regions obtained from partitioning regardless of the data. These features can correspond to object or subobject properties.

Annotations

The first image retrieval systems were all based on textual annotations to the images. Often the images could only be search by exact match in database systems, but with free text retrieval techniques being available, also the similarity–based retrieval of free image annotation becomes possible [212].

Care needs to taken with respect to user subjectivity and synonyms of words since these factors can significantly influence the retrieval results. Much research has been done on the development of controlled vocabularies and ontologies for the annotation of large image collections [113, 114]. A classification of textual visual descriptors is done in [106]. The annotation of hard to retrieve characteristics such as emotions and themes is explained in [114]. In [240], an example for an indexed museum collection is given. In [115], a vocabulary is developed to create ground truth data for evaluating CBIR applications.

Tools for facilitating the annotation of image collections have been proposed [206]. Still, there remains much work to be done for a semi–automatic annotation tool. When using images from web pages, annotation can be attached automatically, for example, from the ALT attribute of the IMG tag in HTML or from the filename or even words close to the image [167, 241, 306].

A mixture between manual and automatic annotation is proposed in [302] where manual annotation is proposed to have a probability of 1 to be correct, whereas for automatic annotation a probability of confidence needs to be calculated. This probability can then be taken into account for the retrieval and can be overwritten manually by the user.

As the need for semantic retrieval becomes more and more evident for real–world applications, and the low level features alone failed to model this semantic content, there are several system that combine the two strategies, and use visual and textual features. Systems such as [241, 294] combine visual and textual features and report better results for a mix of features than for any feature group alone. In [38], the retrieval of visual and textual features does not lead to better results than text only, but it is important to take a close look at the evaluation criteria and whether the ground truth was generated from the text only or based on visual and textual cues.

Color

Color has always been a very fascinating topic for humans. At least from the middle ages on, people were trying to have a scientific approach to it. First color models were derived, but at that time most of the color models were not really based on the colors we physiologically perceive. For example, Young already talked about three color receptors in our eyes:

> From three simple sensations, with their combinations, we obtain several primitive distinctions of colors; but the different proportions, in which they may be combined, afford a variety of tints beyond all calculation. The three simple sensations being red, green and violet, the three binary combinations are yellow, consisting of red and green; crimson, of red and violet; and blue, of green and violet; and the seventh in order is white light, composed by all the three united.
>
> Thomas Young, *Lecture on Natural Philosophy*, 1807

Also with the wrong color scheme, Goethe experimented with colors and described the experiments in a book called *Farbenlehre* ("color theory", 1808–1810). Several well–known physicists such as Newton, Young, Helmholtz and Maxwell worked on experiments for revealing the secrets of our color vision.

A very interesting book on different aspects of color including color in art, nature, culture and mechanisms of the human eye is [133]. Especially the chapter on color in language is well worth reading [147].

Color for retrieval Color is being used as a feature in basically all retrieval systems[4] [25, 26, 45, 71, 73, 77, 90, 199, 264]. Still, there are important differences in the way color is used. First, an appropriate color space needs to be found. Normally, digital images are stored in RGB, but RGB can only be used when the images are always taken under the same "perfect" lighting conditions, which might be the case for museum collections or for trademark retrieval, where color only plays a limited role.

In general, we would like to have a color space where distances within the mathematical color space correspond roughly to how humans perceive differences between colors. This is definitely not the case for RGB and CMY, but it is the case, at least approximately, for HSV, CIE Lab, CIE Luv and the Munsell space.

Comparisons of different feature groups in Section 3.3.4 show that the color histogram obtains the best results for all of *Viper*'s feature groups.

RGB, CMY Although most images are stored in RGB (red, green, blue) color coordinates and also our visual system is based on receptors for red, green and blue, this space is rarely used for retrieval because of its perceptual properties that do not correspond well to human perception. CMY (cyan, magenta, yellow) is complementary to RGB and it is used where colors are mixed in a subtractive way, for example when mixing a color of an ink jet printer. RGB, by contrast, is used when light is emitted as in a computer screen or television (additive color mixing).

HSV (hue, saturation, value) is often used for image retrieval [24, 25, 199, 264, 294] because the distances in the color space correspond roughly to the human perception. The color coordinates are also easy to calculate from images stored in RGB format. Often, less importance is given to the V coordinate [264], as human vision is less sensitive to it. Sometimes the V coordinate is even omitted completely [199] as it can be influenced by the lighting conditions when producing an image.

CIE Luv, Lab The two CIE spaces Luv and Lab and their creation process are described in detail in [217]. These color spaces were constructed to be perceptually uniform and thus the properties are well suited for retrieval. The results are based psychovisual experiments. The transformation from RGB into the CIE color spaces is more complex than that of HSV. Both transformations to CIE Luv and Lab are non–linear.

In [242, 273], CIE Luv is used for image retrieval. [80, 193] make use of the CIE Lab space.

Other color spaces QBIC [71] uses the *Munsell* color space for its feature representation. This color space is based on the three components hue, value and chroma. Munsell created a book with color chips that described the relationships of different colors. At the basis, this color space is based on a non–computational model but computational models do exist.

In MPEG–7 a number of different color spaces can be used for description as explained in [149]. This includes RGB, HSV as well as YCrCb and HMMD.

The study on color invariance in [76] takes into account several factors like different illumination, and different material surface characteristics to create invariant descriptors with respect to a number of influences.

Color representations One of the most simple representation of color is the color histogram described in [270] where the relative number of pixels for every color is counted. Histograms are used in many systems [199, 264]. Sometimes the histogram is slightly blurred, to avoid that small changes in color can have a large influence on the results. In [73], histograms are used over multiple (local) scales in the image. Very easy representations are also the mode color [264] or an average color [7]. This can be useful, when a large number of descriptors is used, for example, for image regions. Color moments are another often used representation of color [199].

[4] `http://viper.unige.ch/other_systems/`

The number of colors used for retrieval also plays an important role. Extremely large numbers of different colors as can be produce by 24 Bit RGB coordinates are not needed because humans cannot distinguish such a large number of colors. Normally, the maximum number of colors used is around 500 with most of the systems using between 50 and 200.

In [45], an attempt is made to express color semantics such as harmony and disharmony.

Texture

Texture in images can be defined as all what is left after color and and local shape have been considered or in terms of structure or randomness [247]. Many different features have been developed to characterize texture, and several image databases containing texture images exist to test these algorithms (VisTex, Section 5.2.3). Normally, it only makes sense to extract these features from homogeneously textured regions, but often, global texture measures are used for general, heterogenous collections of photographs [264].

Besides simple one–value measures such as directionality or regularity, transformations like the Wavelet Transformation or Gabor filters exist to describe texture features.

Wavelets Wavelet filter responses are often used to describe texture for image retrieval [198, 199, 308] despite their sensitivity to noise. A good introduction to Wavelets can be found in [267] and a good book describing Wavelets is [283]. An abundance of resources about wavelets can be also be found on the web at [5].

Figure 2.11: Example of a wavelet transform of the image shown in Figure 3.8.

Figure 2.11 demonstrates the basic principle of the Wavelet Transform where an image is subsequently divided into a base image and the changes in different directions. There are several Wavelet filters that can be used for various tasks. Wavelets are often used for image compressions as well.

In [148], a comparison between different Wavelet–based features, Gabor filter responses and MR SAR features was performed with the Gabor filter receiving the best results.

Gabor filters Sometimes *Gabor filters* are mentioned to correspond roughly to the orientation and frequency–selective properties in the visual cortex [53, 54]. Thus, they have been used for

[5] `http://www.mathsoft.com/wavelets.html`

image retrieval regularly [148, 231, 264] to describe the texture or global structure of images. A descriptor based on Gabor filter responses is also used in [305].

An example of Gabor filter replies in different directions and on different scales can be seen in Figure 3.9. More about Gabor filters can be read in [128, 293].

Also MPEG–7 uses texture descriptors based on Gabor filters with different directions and at various scales [149].

Cooccurence Matrices *Cooccurence matrices* are another simple method to analyze textures [295]. These matrices take into account neighborhoods of pixels in certain direction and distances. As the matrices are numerous and very large with their size being the square of the color resolution, normally features are extracted from the matrices in order to more compactly represent them. Common features used are entropy, contrast, homogeneity, symmetry and others.

Features based on cooccurence matrices are used for web image retrieval in [199]. In [244] characteristics based on cooccurence matrices are used for the analysis of high resolution computed tomography (HRCT) lung images.

Other texture descriptors There are many other descriptors for texture in images. In this section, only a few examples of methods used for image retrieval are given. Many *single–valued measures* are employed such as *directionality*, *contrast* or *coarseness* of the image [193] or, similarly in MPEG–7 regularity, coarseness and directionality [149].

Also in MPEG–7, a histogram of *directions of edges* is calculated to describe texture. This is similar to the *orientation histogram* described in [242], where the edge detection often relies on the techniques already described by Canny in [23]. In [131] a description for texture based on color covariance matrices is presented. The quality of the description is demonstrated with generating a synthetic texture based on the extracted features and comparing it to the original.

Other used features are the *MR SAR* characteristics often used for satellite images or in image retrieval [211, 279]. *Wold* features for texture segmentation are employed in [145].

Shapes

The use of shape features makes most sense when certain objects are searched for and when the image has already been segemented. This can be the case for databases with defined objects such as fishes [165], trademarks [39, 65, 67, 110] or cars [200]. These images often contain one object and they can thus be segmented rather easily. Sometimes histograms of edge directions are seen as global shape characteristics [109, 110].

A survey of shape analysis techniques can be found in [143]. The shape descriptors used for MPEG–7 are presented in [18].

Several classifications of shape characterizations are possible. The most prominent classification is between *contour/boundary* based (Scale Space, Fourier descriptors) *vs. region/shape–interior* based (moments) descriptors.

Curvature scale space In *curvature scale space* (CSS) the image is represented by the zero crossings of its curvature, describing the points where the curvature changes between convex and concave. This extraction is done at several scales where in between the measurements the image is low–pass filtered each time, which corresponds to a diffusion process following the heat equation. Therefore, the number of zero–crossings is subsequently reduced between scales. Curvature Scale Space is described in [165]. Scale space has been applied to image retrieval in [214].

As features, the *peaks* in curvature scale space are used as well as eccentricity and circularity at different scales. Curvature Scale Space is also used in MPEG–7 [18].

Invariant moments Moments are well–known properties from physics. They have also often been used to describe shapes as they are easy to calculate and to interpret. Moments correspond to well–known statistical properties as they 0th moment is the mean and the second moment the

variance of a two dimensional shape (distribution). In general the geometrical moment of order $(p+q)$ of an image function $I(x,y)$ is defined by Equation 2.1:

$$m_{pq} = \int\int x^p y^q I(x,y) dx dy \tag{2.1}$$

As moments of various orders do not contain all the desired invariance properties, normalized moments can be created, and in [96], a set of 7 invariant moments of different orders has been proposed. This set of invariant moments is still in frequent use [39, 110, 132], often for trademark retrieval. These moments are invariant with respect to scaling, shifting and rotations.

Zernike moments, *Pseudo–Zernike moments* and their invariants are more complex to calculate but seem to lead to better results as stated and used in [118].

Fourier Descriptors Fourier descriptors can be used to describe the outline of a contour. As the Fourier transform codes shifts in the phase, and a shift in the contour is basically a rotation of the form, this transform can easily be made rotation invariant. Similarly, scale invariance can be achieved.

In image retrieval, Fourier descriptors of forms are employed in [198] and for trademark retrieval they are frequently used, for example in [132].

Single–valued measures There is an abundance of single–valued measures to describe shapes. QBIC, for example uses descriptors such as area, circularity and eccentricity [193]. A non–exhaustive list contains the following shape descriptors:

- area, perimeter, aspect ratio,
- compactness, circularity, principal axis,
- connected components, holes, Euler number,
- eccentricity, transparency,
- right angleness, sharpness, straightness, directedness,
- complexity.

Definitions for most of these measures can be found in [65, 67, 83].

Other descriptors Early shape descriptors for trademark retrieval include simple *pixel values* [116] and *chain code* description and *string matching* techniques [46]. *Elastic template matching* techniques for shapes are described in [307] to retrieve object images from a database. Other shape descriptors include the *medial axis transform*, *skeletons*, *polygonal* or *spline approximations*. In [183], a *shape matrix* is used to achieve deformation–tolerant retrieval of shapes based on image sketches.

Combinatorial features

In [207, 208] an approach is presented to learn a possibly infinite feature set for classification tasks. Basic features are extracted from the images, and combinations of features are created as new features. The basic features have a partial order, so they can be extracted from the lowest level of complexity to higher levels. The compositions of several features can be geometric relations, topological relations or conjunctive or disjunctive presence in an image. In this way very specific features or feature combinations can be created when they are needed because the expressive power of the existent features was not sufficient for a certain task.

The approach relates to the way that little children learn discrimination strategies, where discrimination strategies are learned from simple to complex tasks. The classification is basically seen as an open task where continually new discriminations strategies need to be created.

In [32, 82, 139], a projection of the feature space onto an infinite–dimensional feature space is used to separate between the class of relevant and the class of irrelevant images using Support Vector Machines (SVMs).

Other features used for retrieval

Other feature used for retrieval include in principal *spatial relationships* of objects or regions in the image, or spatial relations between *points of interest*.

In [68] an overview of possible *spatial relationships* for retrieval is given and an application with GIS (Geographical information system) data is presented. [250, 251] describe a system for spatial color–based queries using the absolute location of a region, its size and spatial relationships of several regions for the matching. In [112], an approach for region–based querying including learning of the importance of regions is proposed based on massive learning. The fact that learning and test set are the same might lead to overly good results. [238] uses spatial relationships of point of interest as a basis for the querying.

Organization of the feature space

There are several fundamental aspects in organizing a feature space. Most systems have a fixed number of features that are extracted for every image. These features often have a continuous numerical value. Based on the values, distances between images with respect to a certain feature or the entire feature space can be calculated [77, 193].

Other systems such as [262] use features that are not in every image, for example a color that is not in an image is not regarded as value 0, but it is omitted from the image description and not used for the query process with this image. In the same way, some features can be binary. Such a binary feature space can also be reached by binarizing continuous feature values into several classes, so the search is restricted images with similar feature values that have to be in the same feature class. The feature space of these systems is then only sparsely populated. This is similar to the distributions of words in text retrieval and thus allows the use of text retrieval techniques. Sparsely populated feature spaces have been employed in the systems explained in [262, 294].

2.3.3 Query formulation

In this part, a look is taken at how the system processes a query posed to the system by a user. The actual interaction of the system with the user (including the query paradigm), and the user interfaces are detailed in Section 4.

The first important factor is whether more than *one query image* can be included in every query step, or if only one query image can be used as example for a search. Queries with only one image cannot be effective and the use of several images for relevance feedback in Section 4.5 show this.

If *several images* are allowed as input, there are two basic ways of executing a query for this set of images:

- Create one pseudo–image from all the input images and execute a query with this pseudo–image.
- Make a query with any of the input images and merge the results in the end.

Both things can be done in several ways, but executing a query for every image is more like an "OR" connection of the submitted images. By contrast, the merging of the input images is often performed more like an "AND" connection of the images in the query. This means that features with a low variance or features that are frequent in the query get a higher weighting than features with a low frequency or rare features. Sometimes it would make sense to explicitly state whether and "AND", "OR" or "NOT" relationship is wanted.

Another important factor for the query formulation is how *negative feedback* is being treated, if it is supported at all. Negative feedback has shown its significance [177] and results in Section 4.5.1

show its effectiveness. As we shall see, there are several problems with the use of negative feedback. Typically if there are more images marked non–relevant than relevant, even features that are regarded as important can get a negative weight and the new images that are returned by the system contain less and less features and are not the desired images at all.

Another important question is of course what is accepted as an input from the user. Besides image examples, various systems use user sketches, image regions, or keywords as input to formulate a query. This is explained in more detail in the query starting point section (Section 4.2) and in the interaction paradigm section (Section 4.3).

The next question for the query formulation is in how much *detail* the user can access the *visual data*. Can he only use entire images for the query or is it possible to use regions of the image for querying? If regions are used, they can of course be either user–defined or they can be presegmented, so the user can only select one or several of these regions.

The last option for query formulation can be the *detail in feature space* that the user has access to. Can he make a selection of features to use for a query? In what detail does he have access to the feature space? Of course, part of this feature importance can automatically be calculated based on relevance feedback, and a novice user might not have a large interest in choosing a feature set by hand. Expert users, by contrast, will often be very interested to tune the query into a certain direction with choosing exactly the features that they are interested in.

2.3.4 Query processing and similarity measures

When the images are represented in the system with meaningful features (as presented in Section 2.3.2), we need to define how to measure the fact that two images are similar based on the features they contain. This depends strongly on the representation of the feature space.

Normally, the images are regarded as vectors in feature space and the distance of the vectors now needs to be calculated with a *distance measure* or *similarity function*. Often a simple metric is used although experiments, for example in [274] (Tversky), indicate that human similarity perception does not at all have the properties of a metric. From his experiments, the similarity seems to be more similar to a number of binary sensations that together form a notion of similarity.

Simple metrics are, for example the *Euclidean distance* (*L2 distance*) or *Weighted Euclidean distance* used in [109, 193] that are easy to compute and to understand. Other commonly used distance measure include the *L1 distance*, *Mahalanobis*, *city block distance* [193].

Another simple distance measure that is frequently used is the *histogram intersection* first described in [270] that is used in [109, 198, 264]. Because the histogram intersection is very sensitive to small shifts in the histogram, often the histogram is blurred with a low pass filter [242] or a *histogram cross correlation* [77] is performed.

A different approach for similarity retrieval is the formulation of retrieval as a classification problem, where there is a relevant and an irrelevant image class. The goal is now to minimize the probability of retrieval error. In [279], the underlying assumptions for different similarity functions (Bhattacharyya, Maximum Likelyhood, Kullback–Leibler, Mahalanobis, Quadratic, χ^2 and Euclidean) are compared with the Bayesian formulation to minimize retrieval error and the results are compared. This *probabilistic retrieval* showed good results [275].

Also in [32, 82, 139], image search is seen as a classification problem between a set of relevant and a set of irrelevant images. Support Vector Machines (SVM) are used to find the separation between relevant and irrelevant images in a projection to an infinite feature space.

In [259], a measure is developed that is based on human partitionings of image sets. Another article that is strongly based on human perception of images is [235]. Here, a *Fuzzy Feature Contrast* (FFC) model is produced that takes into account several psychological experiments on human–perceived similarity such as [274]. The article explains in details several of the theories for similarity perception.

Another approach is often used in text retrieval where very sparsely–populated feature spaces exist [228]. The weighting is based on the *frequencies* of the features both in the image (tf) and in the entire collection (cf). For image retrieval, these similarity functions have been used in the systems described in [264, 294]. A comparison of several frequency–based feature weights can also

be found in Section 3.4.

Sometimes a mix of the above–mentioned methods is used [193, 264], for example using different similarity functions for different feature groups.

2.3.5 Feature access and data storage

Most systems [14, 49, 198, 251, 276, 291] do not mention the actual storage and access methods of the data, which might mean that they often store the entire index in the main memory to be able to execute queries as fast as possible. Still, most of these approaches will not scale well as soon as the index becomes larger than the main memory.

Index in memory

Many systems at the moment, such as [241], assume that the image databases will not become too large and thus hold the entire feature index in the main memory. This certainly has many advantages, especially with respect to execution speed as no slow disk accesses have to be made. On the other hand, such a system can not scale well when the databases become larger or when there are a larger number of features extracted for every image. [242] stores the indexes in main memory as long as they fit in there and otherwise creates a file on the hard disk containing the index tree (K–D–tree). In [253], some strategies to reduce the search space are described.

Even when the indices are completely kept in main memory, methods for efficient access to the data items are needed as linear comparisons would take too much time once image databases get large and/or feature spaces have higher dimensions. In [210], an approach to structure the feature space based on statistical structures of the database is presented. [296] present different tree structures for data access (R–, R*–, SS–, VSR–, etc.) and empirically compare their performances with respect to index generation time, search time and error probabilities. In [247], an overview over several tree structures and their properties is given, including various references.

Several of the techniques used to optimize access to features in main memory can also be used when the access is to the hard disk. Absolute times will of course be much higher, but these tree structures will equally improve the access to items stored on the hard disk significantly.

Databases

As image database and image retrieval were originally seen as an extension to existing database technologies, basically all the early systems used databases to store and access the features. Image content–based technologies are available to access most modern relational databases such as Oracle[6]. Oracle 9i contains a module named *inter*Media that can index images, video and audio. *inter*Media is described in more detail in [156].

Also the *Viper* system described in Section 3 uses a database (mySQL) that can be used instead of the inverted file. This system is described in [146]. A more in–depth approach to use a database system to access image features is explained in [56].

In [158] an approach for image retrieval is presented that completely relies on database technology. A content data model is developed and an extension to SQL, *ISQL* is implemented to query multimedia data. Such an extension works with any relational database and is demonstrated with using the Oracle 8i database.

As IBMs QBIC [71] is part of the *DB2* project, it also makes use of this database to store the data. In [77], a database called Monet is used to store the features.

Inverted file

Inverted files have proven to be very useful in text retrieval [299] and also in image retrieval they are employed in systems such as [262, 264]. Inverted files work especially well when the feature space is only sparsely populated. More about the use of inverted files, especially for image retrieval can be found in Section 3.1.3.

[6] `http://www.oracle.com/`

Many techniques exist to handle inverted files [299] and to efficiently build and query them. More about the access to inverted files and search pruning techniques can be read in Section 4.1.2.

2.3.6 Query paradigms, Results display, Relevance feedback

Everything that involves the interaction with the user is described in much detail in Chapter 4. This section is thus merely a list of links to the corresponding section in the user interaction chapter. An overview of user interaction techniques in image retrieval can also be found in [302].

Query/interaction paradigms

The query paradigms are explained in detail in Section 4.3 in more detail and with the different media used. A distinction is made between the query starting point (Section 4.2) and the further interaction (Section 4.3). This difference is worth mentioning although both are parts of the user interaction. The problem to find a good starting point for a user to begin a query session is still different from navigating a user through an image database with the use of interaction.

Results display

In image retrieval research, the results display does not play an extremely important role at the moment although a few applications show that there is much room for improvements [164, 191]. Several articles also work on how to integrate the results display with the query formulation. In [232, 233], it is proposed that the user can move images displayed on screen closer together, so the system can create a similarity function based on the user's goal. More about results display can be read in Section 4.4. The available interfaces of the *Viper* search engine are presented in Section 4.4.1.

It is important to mention that often it is proposed to not necessarily show only the most similar images on screen. Some dissimilar images might provide the user with better means for positive and negative relevance feedback to explore the similarity space.

Relevance feedback

Relevance feedback is one of the most important techniques in CBIR as it allows to learn more about the goal the user has in mind. Several articles have been written on relevance feedback in text retrieval [218, 229] and in visual information retrieval [177, 221, 222, 229, 247, 264, 302]. More details about relevance feedback and a profound review of several feedback strategies can be found in Chapter 4 and more specifically in Section 4.5.

2.4 Example systems

This section presents a few well–known retrieval systems with their main characteristics. An overview of a larger number of systems with the used techniques can be found at[7] or in [282]. Unfortunately the evaluation of the systems is not very profound. [37] gives a taxonomy of retrieval systems and also presents a few systems with their basic capabilities. In this article, the systems are classified by the *data modalities* that they support (image, text, video) the *schema exposure* (how much the user has to know about the structure of features), the *query mode* (how detailed the search attributes are) and the *matching fuzziness*.

This list will not start with the very early systems such as [35], but rather present a few important systems and explain their impact on the domain of image retrieval.

[7] `http://viper.unige.ch/other_systems/`

2.4.1 QBIC – Query By Image and video Content

IBM's QBIC [71, 193] is by far the most cited system as it was maybe the first system (1993) that really performed content–based retrieval on a number of different features that could be chosen by the user. It was also made available on the web as a demonstration[8] and can be purchased as a commercial product.

QBIC allows the indexation of video, but videos are treated as a sequence of single images. QBIC includes tools for shot boundary detection and to find a representative frame for each shot. Motion is one of the features that can be used to query for video sequences.

QBIC allows to search by a user sketch (shapes based on moments, area, circularity and other measures), by textures (a mix of measures such as coarseness, contrast and directionality), and by colors (average colors in different color spaces, color histogram based on the Munsell space). The user is given several methods to turn on and off features, but feedback with several images and negative feedback are, to the best of our knowledge, not possible. Images can be annotated in the system and then the annotations can be used for querying as well.

2.4.2 Virage

Virage[9] is another very early system of the image retrieval field. It is also a commercial product, and thus information about the retrieval techniques is not always published. It is a big commercial system used by reputable clients such as CNN and NBC.

[6] describes the framework of Virage for image retrieval. The features used for querying include local and global color features, but it is proposed to adapt them to special domains of application. Virage also proposes an API (Application Programming Interface) for the development of client applications.

In [90], the video engine of Virage is described. Again, it is more a framework that can be used to design specialized applications. The program allows the extraction of key frames that can be indexed with the image retrieval features. Besides this, the motion information can be used as a feature as well as the audio part of the video.

2.4.3 PicHunter, Photobook, etc.

The MIT Media Lab produced a number of systems for CBIR, including the early *Photobook*[10] that includes publications as early as 1994 [203]. The system can be downloaded from the internet free of charge and it can be adapted to a variety of applications. Demonstrated are face retrieval using Eigenfaces and retrieval of medical images. The *Photobook* framework also contains *FourEyes*, a learning agent which selects and combines models based on examples from the user [161, 162] using a "society of models".

Much work together with NEC research was also done on the image browser `PicHunter` that is described in [47, 49] and in an improved version in [48]. `PicHunter` assumes that the user is looking for an exact image that exists in the database. At each step the system presents to the user images to maximize the gain in information to find the image the user is looking for. The system is evaluated by counting the number of images that a user is looking at before finding the target that he has in mind.

Recent work has been done on CBIR based on Bayesian networks [275, 278, 279] that also include learning over several temporal scales [277].

2.4.4 MARS

MARS[11] (Multimedia Archival and Retrieval System(s)) describes an entire series of different systems that are all developed within the same framework based in the University of Illinois in

[8] `http://wwwqbic.almaden.ibm.com/`
[9] `http://www.virage.com/`
[10] `http://www-white.media.mit.edu/~tpminka/photobook/`
[11] `http://www-db.ics.uci.edu/pages/research/mars/index.shtml`

Urbana Champaign. The systems differ with respect to features and techniques used. In [197, 198], the system uses color, texture, shape and layout features, eventually also integrating keywords. The colors are based on the HSV space with neglecting the V component as it can be influenced by lighting conditions. As color features, coarseness, contrast and directionality plus wavelet–based features are used, whereas the shape features are based on Fourier descriptors. Local layout features by partitioning the image into fixed blocks and calculating local color histograms. [93] contains an evaluation of MARS including tests with real users and a small comparison of the user results.

[199] shows an extension of MARS for managing HTML documents containing several media, such as images and text. A multimedia object model is created and queries are then executed based on the contained objects. The results when combining text and visual features show to outperform any of the two techniques used alone. In general, the HSV color space is used with features being a simple histogram intersection and color moments.

Much work has also been done on the display of the images on screen for the MARS system. In [164], a method to display images on screen with distances being based on their similarity characteristics is shown. The multidimensional features space is reduced to a two–dimensional space by PCA (Principal Component Analysis) and an algorithm for minimal overlap of the displayed images is developed.

In [191], experiments with a three–dimensional interface are shown. The three axes symbolize color, texture and structure features, and the user can move in the display space, mark images as relevant and non–relevant and execute new queries. The use of the axes with static feature groups may be a limitations as the users' queries might rarely comply with exactly one of the axes in feature space.

2.4.5 Blobworld

One of the first and may be the best known retrieval system that uses image regions for the query process is for sure *Blobworld*[12]. It is described in several articles [24–26] and changed significantly over time.

In general, the system first segments the images into a small number of homogenous regions based on 6–8 color and texture features. In the first version [24, 25], 6 features were used for segmentation and the color features were selected based on the HSV space, and in [26], 8 features were used for segmentation and the color features were used in the CIE Lab space. While the first versions used ellipses to symbolize the image regions, the more recent versions used the real boundaries of the regions as shown in Figure 2.12.

(a) A tiger image (b) A segmented tiger

Figure 2.12: An image used by Blobworld and its segmented version.

Blobworld offers to select one or several regions and query with the regions based on the regions

[12] `http://elib.cs.berkeley.edu/photos/blobworld/`

color or texture characteristics and also based on the spatial relationships of regions. Another advantage of the system is that the regions responsible for the retrieval are highlighted in the results section, facilitating the use of negative feedback. Although the segmentation might not always conform with the objects a user might be looking for, but it does give a much more exact way to specify queries.

2.4.6 Other systems

Some other image retrieval systems include the very early $\sum nigma$ [78] that uses different types of images (medical images, topographic maps) that are searched for predefined structuring elements.

Another early system is *Candid* [117] that uses signatures for satellite and medical image retrieval.

So far, the only *meta search engine* for images might be *Metaseek* [10] which combines several well–known search engines such as Virage, QBIC, Webseek and Visualseek. All engines are queried and then the results are combined. Relevance feedback is also possible with all the images of possibly different databases. *VisualSeek* is described in more detail in [251, 252]. It supports query by local regions based on color.

[298] described the system of the *IMedia* group and specializes on image segmentation and retrieval based on regions. In [130], *CHROMA*, a system for hierarchical browsing is presented, rather similar to a directory hierarchy. Sketch retrieval is also proposed. *ImageRover* [273] is another retrieval system that is based a variety of features (colors, edges, textures, shapes). The *El Niño* system [234] offers an interesting user interface, where the images are presented with distances based on their similarity. The user can then move images closer together that he thinks are similar and thus changes the similarity space.

2.5 Related issues

Besides the actual techniques used for image retrieval, there are several fields that might become essential for the further development of all image retrieval systems. MPEG–7 might change the way we search for images as it allows the intensive labeling of images and the storage of image features directly with the image. If these features are supplied, a retrieval engine would not need to do any feature extraction anymore, but could simply use the features supplied with the image.

MPEG–7 not only addresses image annotation but deals with different media which addresses the important part of retrieving other media than images as described in Section 2.5.2. The retrieval of different media and as well mixed media retrieval will become more and more important as there are several media available on the web already at the moment. To manage all the different media in a proper and integrated way will not be an easy task.

2.5.1 MPEG–7

The standards of the Motion Picture Experts Group (MPEG[13]) have had great influence on the development of these media. Already the MPEG–1 standard is being used for the Video CD and also the well–known MP3 is part of this standard as the audio layer. Similarly MPEG–2 has been used for digital television since it became valid, and the MPEG–4 standard is being employed in several applications at the moment.

The development of MPEG–7[14] [15] [16] [17] was meant to create a *multimedia content description interface* [55, 166]. This means that the standard not only contains information about the objects

[13] http://www.mpeg.org/
[14] http://drogo.cselt.stet.it/mpeg/standards/mpeg-7.htm
[15] http://www.mpeg-7.com/
[16] http://www.ctr.columbia.edu/~ana/MPEG7/MPEG7project.html
[17] http://ipsi.fhg.de/delite/Projects/MPEG7/

but that also some interpretation of the information is available. MPEG–7 was not developed with a certain application in mind, but it was to support a very broad range of various applications.

Here is an overview of the different standards developed by MPEG:

- MPEG–1 For the storage and retrieval of moving pictures and audio on storage media (status: finished).
- MPEG–2 For digital television and DVD, it is the timely response for the satellite broadcasting and cable television industries in their transition from analog to digital formats (status: finished).
- MPEG–4 Codes content such as objects and enables those objects to be manipulated individually or collectively on an audiovisual scene (status: finished).
- MPEG–7 Multimedia content description interface to serve for various applications (status: finished).
- MPEG–21 Multimedia framework as infrastructure for the secure delivery and consumption of multimedia content (status: open).

Thus, the different standards all serve a certain objective within the framework of multimedia. All of them have been created in a long process that includes academia and industry to incorporate all the different interests. Many meetings have been held in the process and thousands of documents with proposals for the standards have been produced.

MPEG–7 objectives

MPEG–1, –2, and –4 make content available. MPEG–7 lets you to find the content you need. MPEG–7 is a standard for describing features of multimedia content. Therefore, it addresses diverse data types such as still pictures, graphics, 3D models, video, audio, speech and combinations of these. A number of data types are included to describes these media in the same, standardized way.

MPEG–7 allows the description of several objects separately that form a scene and it allows to define temporal or spatial relationships between these objects. It also allows to describe these objects with different granularities on different levels.

MPEG–7 contains many different description schemes with low level features such as color or texture [149] that can be extracted automatically, and semantic features that need human intervention to be generated.

Content description

In general the content description in MPEG–7 is completely done in XML[18], the standard for information exchange not only on the WWW. This description also contains the encoding scheme for binary information, so a system knows if it can understand all of the information supplied in a file.

Visual descriptors for features like color and texture can be found in [149]. Other descriptors include information about the copyright, classification of the content such as parental rating and possibly links to other relevant material.

Impact on Multimedia Retrieval

MPEG–7 might have a strong impact on the multimedia retrieval community. It allows for the storage of content descriptions in a unique format and could make it easier to reuse generated descriptions for multimedia content. It also allows for the description on several levels of granularity making it possible to have an image already parted into semantically meaningful regions that can then be described by low level features.

[18] `http://www.w3.org/XML/`

Although it might still take some time before MPEG–7 is really used for multimedia data distributed on the web and stored in multimedia databases, it makes available a unique tool or better description scheme to describe the content of documents and makes available a large amount of information that can be used for retrieval once available.

Other annotation initiatives for different media

Other initiatives are also under way to annotate different media, notably on the WWW by the *WWW consortium* (W3C[19]) and within the framework of the *semantic web*[20] [15]. All these different iniatives are based on XML as the standard description language for data as it is also the case for MPEG–7.

One of the activities within the semantic web iniative is the *resource description framework* (RDF[21]). This framework integrates a variety of applications for music, images and other data types that are accessible via the web. The goal is to share knowledge on the described data types. A lightweight ontology is supplied with the specifications of RDF.

Annotea[22] is another initiative within the framework of the semantic web that deals with the annotation of objects on web pages. It is based on RDF for the annotation scheme and a first client implementation is *Amaya*[23], W3C's web editor/browser.

The *scalable vector graphics* format (SVG[24]) is another standard of the W3C that can be used to describe two–dimensional graphics in XML. It is possible to create animations via scripting with SVG. Event handlers allow for actions to be taken once, for example, the mouse is moved over an object on a web page.

The W3C has also developed a standard language for synchronized display of several media on screen, *SMIL*[25] (Synchronized Multimedia Integration Language). This HTML–like language makes it easy to create simple web pages with media that need synchronization such as streaming audio and video or animations.

Other annotation efforts is the *Dublin Core Metadata Initiative* (DCMI[26]) to create interoperable metadata standards.

2.5.2 Retrieval of other media

As already said earlier: although the algorithms described in this thesis have so far only been tested with images but much of the theory can be used for the retrieval of other media and also a mix of several media. The features used will of course be different but the basic techniques for the structure of a system, storage methods and retrieval techniques can be very similar. Although so far most of the work has been on images (and of course on text), in the last couple of years much work has also been done on the integration of music, sound and videos.

Music, sound

The idea for music retrieval is similar to that in image retrieval. Wouldn't it be nice to simply hum or sing a song one remembered from the radio and then the search engine can find the title and the singer, plus maybe the location where the song can be downloaded on the internet? Another scenario is to classify songs into different kinds of music. Like this, someone who likes a certain song can quickly find other songs of the same kind of music.

Since there are several music exchange services on the web such as Napster[27], this music retrieval has become more important through the use of the WWW because more digital music data is

[19] http://www.w3c.org
[20] http://www.w3.org/2001/sw/
[21] http://www.w3.org/RDF
[22] http://www.w3.org/2001/Annotea/
[23] http://www.w3.org/Amaya/
[24] http://www.w3.org/Graphics/SVG/
[25] http://www.w3c.org/AudioVideo/
[26] http://dublincore.org
[27] http://www.napster.com/

available by now. There are several articles on how to search for music or classify songs. [142] describes the creation of playlists with music of a similar genre using a music database containing 8000 songs. Local signatures are used for the comparison, so music can be matched local parts of the song.

Another article on retrieval from acoustic input is [157]. The authors try to retrieve songs from a sung or hummed example, using a database with 9600 folk songs. Their program gets the user's input and transcribes it into musical notation, before a retrieval from the stored songs can be done. As the quality of the user input can vary strongly, a correction of the user input needs to be done. This article only uses the beginning of the songs for retrieval, but for most songs the characteristic part will more likely be a chorus or refrain, making local characteristics of the songs necessary.

Video

Already the Virage framework described in [90] and QBIC [71] worked on the search for video data. Many clients such as television stations have to deal with large amounts of these data and more and more of it becomes available in digital form. Many video spots are already produced in digital form with the first completely digital cinemas appearing and with DVD films being on the rise.

In [7], a simple video browsing tool for end users is described, with generating an average color of every image and allowing browsing with a color slider to find scene changes and interesting parts of a movie manually.

Videos have several advantages compared to images. They contain more information than images and especially semantic information when the spoken text of the videos can be analyzed via speech recognition and then be retrieved. Together with the image information, this can lead to better results. Also, DVDs often contain subtitles that can be used for retrieval even easier and less error–prone than speech recognition tools.

Of course, there are also important problems in video retrieval. As there is a larger amount information available than in image retrieval, it is impossible to simply index all the single images of a video (normally ~25 images per second). The information needs to be compressed to make retrieval efficient. Normally, scene changes are tried to be detected and then a representative image for every scene can be used for representation. When there is a large change in scene, the detection of scene changes is easy, but often, only the camera is moving slowly and thus it can be hard to detect where a scene really changes. This is also the case for dissolves, where one image goes slowly over into another image or other effects that are often used for scene changes. In this case, it is hard to determine where exactly the scene change occurred. A good, though slightly outdated, overview of video retrieval techniques can be found in [4]. Another review of video retrieval techniques is [150].

Features used for video retrieval can be the audio content, eventually with using speech recognition or similar features to image retrieval obtained from key–frames, such as texture, color or shape features. Some programs also use the motion changes (temporal information) in the video as retrieval features. Captions in the video are other features that can be used.

Mixed media retrieval

The general structure of the *Viper* system described in Section 3 was created to be used with a mix of media. All documents are addressed by their URLs and thus they can contain several media for retrieval. A project to combine not only these media but also to make retrieval for other file types in a general desktop environment is *Fer de Lance*[28]. A project on multi–modal interaction is *IM2*[29] (Interactive Multimodal Information Management).

So far, the main mixtures of several media are the use of textual and visual image information combined for querying [241, 294].

[28] `http://www.fer-de-lance.org/`
[29] `http://www.im2.ch/`

2.6 Summary

This chapter explained the basic technologies that are needed for and used in content–based image retrieval. A short introduction was given into human physiology and human perception of images that are the fundamentals of every CBIRS. The main features used in CBIRSs were explained and example systems for the feature sets are listed. The principal access and storage methods used in image retrieval systems were demonstrated as well. Some retrieval systems were illustrated in more detail. Especially the related issues such as MPEG–7 should be read with care as they can become a major factor in the field and open new possibilities for retrieval. Through the use of MPEG–7, a larger amount of information and especially semantic knowledge might become available that could alter the way that we search content information of multimedia data at the moment. If this information is produced at the time the image is created, a much better retrieval might become possible. At the end of this chapter, the use of content–based techniques to retrieve other media was described. For sure, the different media retrieval techniques will converge more and more to arrive at true multimedia retrieval.

Chapter 3

The *Viper* System

> Communication is the means which enables society to adjust itself to alterations of technology and education and other social changes. The scientific method can offer no grand vision, no global strategy, no panacea. It will never be possible to demonstrate that anything is absolutely right or even completely scientifically true.
>
> L. T. Wilkens, *Social Deviance*[297], Page 28. (taken from [41])

The *Viper* system (Visual Information Processing for enhanced retrieval) is the content–based image retrieval system (CBIRS) that is developed in the computer vision and multimedia laboratory at the University of Geneva. The main idea of the system is to adopt text retrieval techniques into the domain of image retrieval and to use a large feature space, just as text retrieval systems deal with a large number of words per document. Text retrieval is, compared to image retrieval, a very mature domain. Text retrieval systems such as the SMART System [226] date back to the 1960s. Many of the techniques of image retrieval are actually adopted, or we could say reinvented, from the text retrieval domain. This is the case for the use of relevance feedback, or frequency–based weights. A more formal look is thus taken at the text retrieval literature. An image retrieval framework is consequently developed that has many similarities with text retrieval systems, since text retrieval techniques have shown to be very useful in several image retrieval systems [56, 294].

Of course, there are many fundamental differences between textual and visual documents. In text retrieval, the words of the text itself are the characteristics to search for. In image retrieval we have to extract meaningful features from the images that can subsequently be used like textual features in text documents, although they might not contain the same amount of semantic information that words do.

An introduction on the use of text retrieval techniques used in *Viper* can be found in [260, 262, 264]. The book chapter [261] and [263] give a longer and more detailed description of the *Viper* image search framework. A description of the architecture of the system can be found in [184] in much more detail. A description of the *Viper/GIFT* (GNU Image Finding Tool) system in German can be found at [169].

The *Viper* system can be used as a demonstration version on the WWW[1] and a stable version can be downloaded free of charge from the GNU web page as the *GIFT* [2] under the GNU General Public License (GPL).

This chapter presents our system and already introduces several concepts for user interaction and system evaluation that are further detailed in Chapters 4 and 5, respectively. It is recommended to read these chapters for a more detailed description.

[1] `http://viper.unige.ch/`
[2] `http://www.gnu.org/software/gift/`

3.1 System architecture

The general idea when constructing *Viper* was to build a system of components where different parts like the image features or the interface could be replaced easily. The entire system relies on web technologies, such as the use of URLs to identify images and the use of the XML–based Multimedia Retrieval Markup Language (*MRML*, see Section 3.2) for logfiles and configuration files.

Further ideas about the structure of the distributed *Viper* framework can be found in [179].

3.1.1 Global structure

In Figure 3.1, we see a graphical representation of the main blocks of the *GIFT / Viper* CBIR framework. Within this framework, it is easy to adapt the system to diverse requirements and use it in various domains (medical, trademarks, consumer photographs). The replaceable components of our project such as the various algorithms for querying, multimedia collections and characteristic features are described in the following sections.

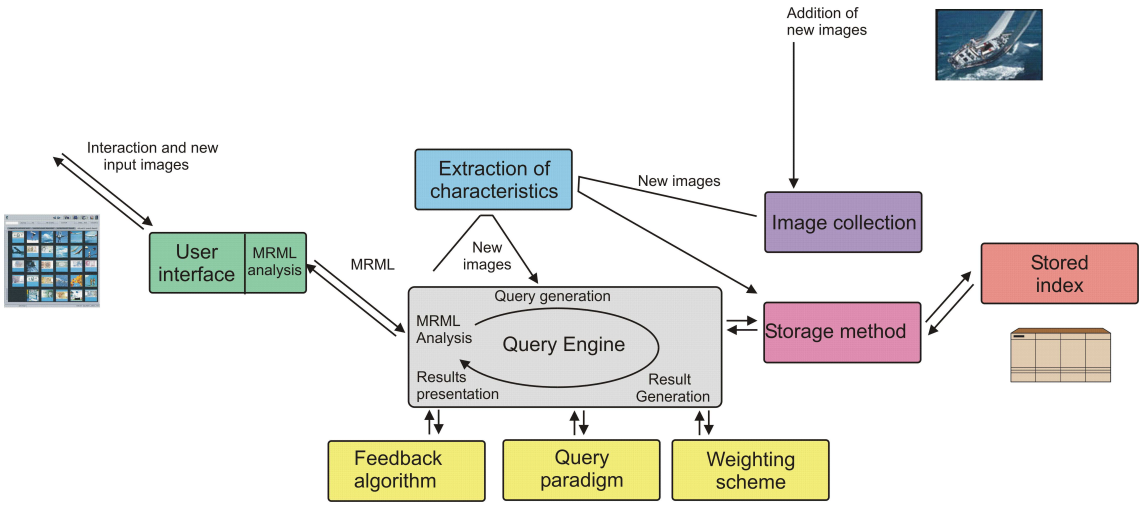


Figure 3.1: The framework of components of the *Viper* system.

The *user interface* allows the user to submit new or existing images of the database as QBE queries, which are expressed to the system in *MRML*. If the image is new, the *extraction* of the characteristics is started, otherwise the already existing characteristics are taken from the database with the indexed characteristics. The extraction of characteristics is also started when a *new image is added* to an image database. All the characteristics are finally stored into an *index* via a *storage method*, in our case normally the inverted file (see Section 3.1.3). We also developed a storage method for the features in a relational database (mySQL) with *Viper*.

The actual *query engine* evaluates the queries and demands done in *MRML*, generates a query from the input image(s) and with the data from the index calculates the results for the query. This process includes different possible *weighting schemes*, various *query paradigms* and several possible *relevance feedback* mechanism. For a more program oriented view of components of *Viper* it is recommended to read [184].

3.1.2 Distribution

Viper is built to work in a distributed environment. Thus, all the images are accessed by their URLs, so they can be at any web accessible location or on any local hard disk. The communication protocol *MRML* (Multimedia Retrieval Markup Language) was also created for this task. In the beginning, it was mainly created to separate the user interface from the actual search engine as

illustrated in Figure 3.5, and rapidly showed to be useful in other tasks such as evaluation (see Chapter 5) of retrieval systems and learning from user interaction data (see Section 4.6).

By having a Java interface (SnakeCharmer, see Section 4.4.1), the goal was from the start to have an interface usable from basically any web browser to access a search engine at any distant location.

Corba, PVM, threads

The use of an inverted file to access the features and the frequency–based weights made a parallelization of the similarity calculation fairly easy. Thus, we started with trials for parallel accesses using *PVM* (Parallel Virtual Machine) [181]. This led to good results, but the communication between the different computers still took away some of the improvements in speed.

The use of *CORBA* (see Figure 3.2, [179]) was meant to distribute the computational load, especially, when several users access large collections at the same time. Also this stayed a research prototype.

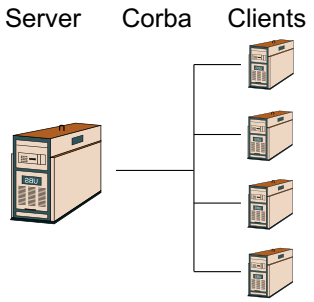


Figure 3.2: Distribution of the computing with Corba.

The actual version of *Viper* uses *threads* that are started for each feature group on evaluation. This works well on our hardware architecture (see Section 3.1.4) and is easy to maintain, easier than a PVM–based program.

3.1.3 Efficient feature access

The use of the *inverted file* allows a very efficient feature access if the frequencies of the features is roughly distributed like in a *Zipf* distribution [20, 310]. Zipf showed that this is the case for words in English texts and this could be shown for several of other fields as well (such as the sizes of cities). A schema of the Zipf distribution is illustrated in Figure 3.3. This basically means that there are a few extremely high values, in our case features and with the frequencies going down, the number of features with this frequency is rising strongly.

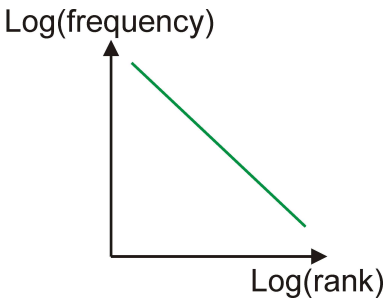


Figure 3.3: The Zipf distribution characterizes *i.e.* the distribution of words in English texts.

The *Viper* features might not be exactly distributed according to a Zipf distribution, but follow it well, with a small number of very frequent features and then a very large number of very rare

features.

In general, we can say that for a query q with n query images ($\mathcal{Q}$) we do not need to use the entire feature space $\mathcal{F}$ but we limit ourself to the features F_j present in the query images $\mathcal{Q} = Q_1 ... Q_n$.

Many efficient algorithm to construct and manage large inverted files have been proposed in the literature [216, 299].

Inverted file

In an inverted file, the access to the features is different than in many other storage methods, in which for each image a number of features for is stored and for which the distance calculation is done document by document. In contrast to that, for every feature, the images containing this feature are stored. Thus, the access is by feature and not by document. Inverted files are the preferred storage method in text retrieval and much literature on creating and searching them efficiently is available [216, 299].

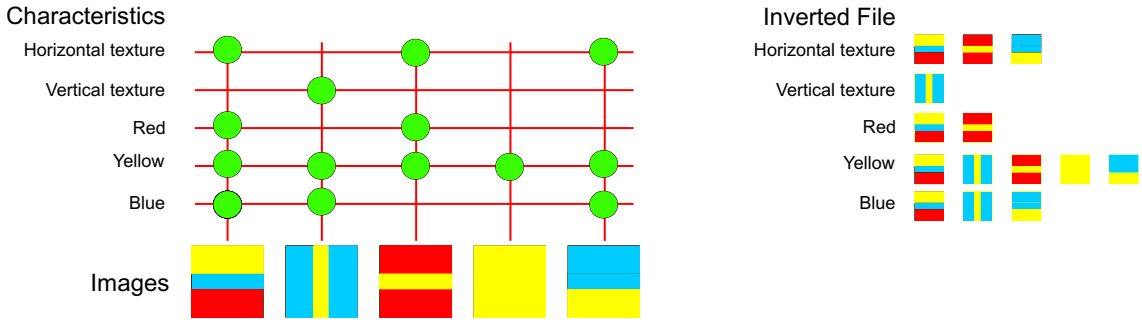


Figure 3.4: The storage structure in an inverted file.

Figure 3.4 explains this principle. One consequence of the use of the inverted file is that very rare features can be analyzed for retrieval extremely quickly. The use of a sparsely populated feature space also reduces the search to the feature subspace spanned by the features present in the query. In a way, storing our features in an inverted file is a form of data preparation for our query technique.

Besides its advantages, an inverted file also has several inconvenients. The addition of new images into the index results in a complete regeneration of the inverted file. Because of these problems, inverted files are often used in environments where new images are rarely added, or where the update of the index is done for several images at certain intervals. The regeneration of the index is very quick for short inverted files, but for large inverted files it takes a longer time.

Because of this, often, two inverted files are used in practice, a large one containing the entire index, that is updated only rarely, and a small inverted file that contains all the images that are added in between two updates of the large index. All the files from the small index can then be transferred to the large index regularly. Queries have to be made to the two inverted files in order not to miss any possibly relevant images.

We modeled this by implementing an interface to a relational database (mySQL[3]) described in [146]. The results show that implementing an inverted file–like structure based on a relational database is feasible for small indexes, but for large image databases, the performance is slower than using an inverted file. Anyway, by having a small file on the relational database and the large index as an inverted file, part of the problems with updating are solved.

Search pruning

Viper offers several methods of search pruning as explained and evaluated in Section 4.1.2 and in [181, 260]. The most important principles of pruning are:

[3] `http://www.mysql.org/`

- Stop the evaluation of *features* at a point where the first N result images might not change very much anymore. This works well when the features are sorted based on their importance to a query.
- Stop the evaluation for *images* when you know that there is only a very small chance that they can still make it into the top N result images.

It can be shown that the very frequent, and thus very costly, features can almost be seen as noise and do not significantly change the retrieval results. Section 4.1.2 also shows the difference between the block and the histogram features with respect pruning. The features of *Viper* are described in Section 3.3.

3.1.4 Hardware

The hardware system of the *Viper* project changed significantly during the time frame of the project. Therefore, the execution times stated have always to be taken as comparative measures at a given time and not as absolute values. Compared times were always taken with the same hardware and with roughly the same conditions. Varying network traffic might be responsible for small absolute differences in execution time.

Our first system was a Sun Ultra 10 with 128 MB of main memory and a 8 GB harddisk running Solaris as an operating system. This system was upgraded to 256 MB of RAM for further tests.

The newer tests were all done with a PC containing four Pentium III Xeon processors at 550 MHz and 1 GB of RAM. This system also has a RAID controller with 16 MB cache and two 8 GB harddisks. These harddisks were later upgraded to 16 GB models. Fast disk access makes a difference for our system as all the indices are stored and read from disk. The Linux–specific caching mechanisms are used and also the distribution to the processors is done by the operating system.

For measuring the execution time, we had two significantly different methods. The first time measures were taken using the processor time for calculating the results directly in the server program. This can be misleading and shows results that look better than they actually are for a real user. The newer measurements include the transfer of the query and the return of the results in *MRML* into the time measured. The time was measured on the client side. As the entire database is retrieved, this can mean extremely large lists of 1000s of URLs that are transmitted by the network. Real users will get faster replies as they normally only look at $20 - -50$ images that need to be transmitted. This new time measurement is therefore an upper bound of the real speed and real requests should be handled much faster.

The network speed thus plays an important role in the evaluation. All results are obtained in our 100 Mbit University network and no separate network part was used.

3.2 *MRML*

The Multimedia Retrieval Markup Language (*MRML* [4]) [187–189] was originally developed to separate the user interface from the actual retrieval engine, and thus, to allow the reuse of components of the retrieval system. Such a communication protocol also allows us to automatically access the retrieval system with scripts, for example, for benchmarking applications or for meta search engines. The need for such a common access method for CBIRS supporting several data types and interaction paradigms was already defined in [37]. Many of the further developments of our system such as long–term learning (see Section 4.6) and benchmarking (see Chapter 5) were strongly dependent on the use of *MRML*.

It actually is used in several applications such as benchmarking for QBE and benchmarking [172, 173] for image browsers [185, 186]. *MRML* is based on XML so that standard, freely–available parsers such as expat[5] or apache's Xerces[6] can be used. *MRML* is a multi–paradigm protocol,

[4] `http://www.mrml.net/`
[5] `http://www.jclark.com/xml/expat.html`
[6] `http://xml.apache.org/`

offering features such as query by example (QBE), choice of databases, features or algorithms to use, and property sheets for specifying algorithm–specific parameters. It is extensible, so that private tags for special features of a system can easily be specified (even SQL can be embedded, if desired). Further details on *MRML* can be found in [188]. Part of this text has been taken from the above descriptions.

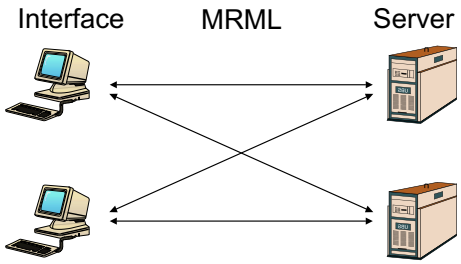


Figure 3.5: Communication with *MRML*.

A protocol like *MRML* allows the reuse of user interfaces for new CBIRSs, and permits the user to connect to a variety of disparate query engines from the same interface, as illustrated in Figure 3.5. Not only does this reduce development time for CBIRSs and learning time for the user, but it also facilitates the comparison of query engines.

3.2.1 Scope

MRML, in its current specification, supports the following features:

- request of a capability description from the server,
- selection of a data collection classified by query paradigm; it is possible to request all collections which can be queried in a certain manner,
- selection and configuration of a query processor, also classified by query paradigm; *MRML* also permits the configuration of meta queries during run time,
- formulation of QBE queries,
- transmission of user interaction data.

3.2.2 Why XML for the description?

There are important reasons for using XML rather than a communication framework such as CORBA as a basis for the implementation of *MRML*. The first is that when using XML no large communication framework is necessary, as it is for CORBA. Secondly, *MRML* offers a common human–readable format for log files. Having a simple common format for user data will make it easier for research groups to share this type of data. Together with common free image collections (such as [7]), *MRML*–compliant systems can form a tool for collecting and sharing user interaction data.

Another reason for the use of XML as a basis for *MRML* is the large number of free XML tools available such as parsers and tools to evaluate files in XML format. XML is the main description language for all kinds of meta data on the internet and is also used as content descriptors in MPEG–7[8], thus ensuring the long–term support of its specifications.

[7] `http://www.cs.washington.edu/research/imagedatabase/groundtruth/`
[8] `http://www.mpeg-7.com/`

3.2.3 Example *MRML* code

An *MRML* server listens on a port for *MRML* messages on a given TCP socket. When connecting, the client requests the basic properties of the server, and waits for an answer. The simplified *MRML* code looks like this:

```
<mrml>
  <get-server-properties />
</mrml>
```

The server then informs the client of its capabilities.

```
<mrml>
  <server-properties />
</mrml>
```

A reply to a request for all collections can, for example, look like this:

```
<mrml>
  <collection-list  >
    <collection collection-id="c-tsr500"
                collection-name="TSR500"
                cui-algorithm-id-list-id="ail-inverted-file"
                cui-base-dir="/home/henning/databases/TSR500/"
                cui-feature-description-location="InvertedFileFeatureDescription.db"
                cui-feature-file-location="url2fts.xml"
                cui-inverted-file-location="InvertedFile.db"
                cui-number-of-images="500"
                cui-offset-file-location="InvertedFileOffset.db" >
     <query-paradigm-list  >
       <query-paradigm  type="inverted-file" />
       <query-paradigm  type="perl-demo" />
     </query-paradigm-list>
   </collection>
   <collection  collection-id="c-wash"
                collection-name="Washington"
                cui-algorithm-id-list-id="ail-inverted-file"
                cui-base-dir="/home/henning/databases/washington/"
                cui-feature-description-location="InvertedFileFeatureDescription.db"
                cui-feature-file-location="url2fts.xml"
                cui-inverted-file-location="InvertedFile.db"
                cui-number-of-images="911"
                cui-offset-file-location="InvertedFileOffset.db" >
      <query-paradigm-list  >
        <query-paradigm  type="inverted-file" />
        <query-paradigm  type="perl-demo" />
      </query-paradigm-list>
    </collection>
  </collection-list>
</mrml>
```

Using similar simple messages, the client can request a list of collections available on the server, together with descriptions of the ways in which they can be queried (query paradigms).

The client can open a session on the server, and configure it according to the needs of its user (interactive client) or its own needs (*e.g.* benchmarking server). The client can also request the algorithms which can be used with a given collection:

```
<mrml>
 <get-algorithms
  collection-id="collection-1" />
</mrml>
```

This request is answered by sending the corresponding list of algorithms available for `collection--1`. This handshaking mechanism allows both interactive clients and programs (such as meta query agents or automatic benchmarkers) to obtain information describing the server.

In a similar simple manner, the client can open and close sessions for a user, and configure the algorithms chosen by the user. This enables multi–user servers and also on–the–fly learning by the query processor.

Interface configuration

The client can then request property sheet descriptions from the server. Varying algorithms will have different relevant parameters which should be user–configurable (*e.g.* feature sets, speed vs. quality). *Viper*, for example, offers several weighting functions [228] and a variety of methods for, and levels of, pruning [181]. All these parameters might be irrelevant for other search engines. Thanks to *MRML* property sheets, the interface can adapt itself to these specific parameters. At the same time, *MRML* specifies the way the interface will turn these data into XML to send them back to the server.

Here is short example of an interface configuration:

```
<property-sheet
 property-sheet-id="sheet-1"
 type="numeric"
 numeric-from="1"
 numeric-to="100"
 numeric-step="1"
 caption="\% features evaluated"
 send-type="attribute"
 send-name="cui-percentage-features" />
```

This specifies a display element which will allow the user to enter an attribute with the caption "% of features evaluated". The values the user will be able to enter are integers between 1 and 100 inclusive. The value will be sent as an attribute *e.g.* `cui-percentage-features="33"`. This mechanism allows the use of complex property sheets, which can send XML text containing multiple elements.

Query formulation

The query step is dependent on the query paradigms offered by the interface and the search engine. Here, only QBE is described, but *MRML* is also being used for browsing queries and tests are being done with region–queries in *MRML*.

A basic QBE query consists of a list of images and the corresponding relevance levels assigned to them by the user. In the following example, the user has marked two images, the image `1.jpg` positive (`user-relevance="1"`) and the image `2.jpg` negative (`user-relevance="-1"`). All query images are referred to by their URLs.

```
<mrml session-id="1" transaction-id="44">
<query-step session-id="1"
 resultsize="30"
 algorithm-id="algorithm-default">
 <user-relevance-list>
  <user-relevance-element
   image-location="http://viper.unige.ch/1.jpg"
```

```
   user-relevance="1"/>
  <user-relevance-element
   image-location="http://viper.unige.ch/2.jpg"
   user-relevance="-1"/>
 </user-relevance-list>
</query-step>
</mrml>
```

The server will then return the retrieval result as a list of image URLs along with their similarity scores.

```
<mrml session-id="1" >
 <acknowledge-session-op  session-id="1" />
 <query-result>
  <query-result-element-list>
   <query-result-element  calculated-similarity="0.9"
                          image-location="http://viper.unige.ch/3.jpg"
                          thumbnail-location="http://viper.unige.ch/3_thumb.jpg" />
   <query-result-element  calculated-similarity="0.7"
                          image-location="http://viper.unige.ch/4.jpg"
                          thumbnail-location="http://viper.unige.ch/4_thumb.jpg" />
  </query-result-element-list>
 </query-result>
</mrml>
```

Queries can be grouped into transactions. This allows the formulation and logging of complex queries. This may be applied to systems which process a single query using a variety algorithms, such as the split–screen version of *TrackingViper* [190] or the system described by [136]. It is important in these cases to preserve in the logs the knowledge that two queries are logically related one to another.

3.2.4 Applications

So far, *MRML* has already been used in several user interfaces described in Section 4.4.1 and benchmarks described in Section 5.3. Besides this, it is well suited for applications such as meta search engines.

Benchmarking

The benchmarks described in Sections 5.3, and 5.4 both use *MRML* for the communication [172, 173]. The browser benchmark [185] also uses *MRML* for the entire communication.

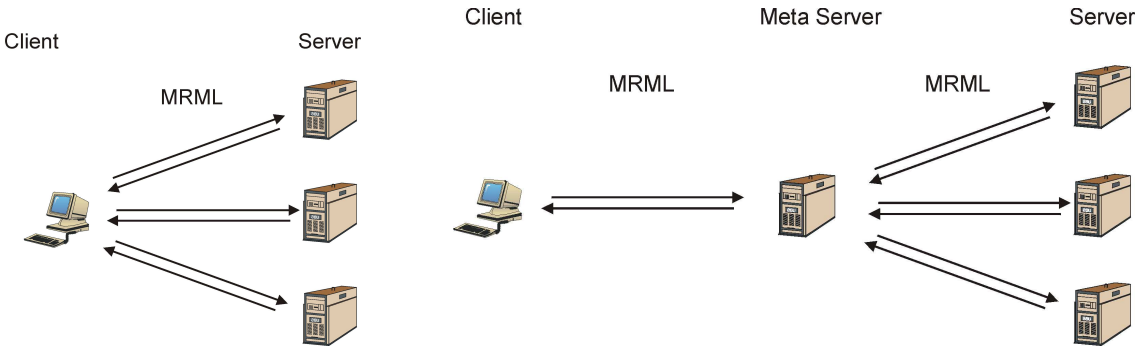
The use of *MRML* allows a complete automization of benchmarking activities without any user intervention because relevance feedback can automatically be generated from the relevance judgments (see Section 5.3.1). Thus a benchmark can produce results for the use of relevance feedback in an objective way. This makes it also very easy to control the effect of changes to the system on the retrieval accuracy on the fly and without much effort. Prototyping the evaluation of the prototypes can thus be made much quicker.

Meta search engine

Meta search engine exist not only on the WWW like Metacrawler[9] and IxQuick[10] but also in CBIR [8, 10] like MetaSEEk[11]. So far, a different interface had to be implemented for every retrieval system. The use of *MRML* makes such a number of different access interfaces unnecessary.

[9] http://www.metacrawler.com/
[10] http://www.ixquick.com/
[11] http://ana.ctr.columbia.edu/metaseek/

An interface can just send the same query to a number of search engines in the exact same format and then merge the incoming results based on whatever strategy [179].



(a) Interface contacting several search engines

(b) Meta server

Figure 3.6: Two concepts for meta search engines.

Figure 3.6 shows two generally different ways to implement meta search engines with *MRML*. The first one leaves the functionality up to the client (user interface). The user interface can contact several search engines and can then show the results on screen that seem to be the most appropriate. The second image shows something we can call a *meta server*. Here, a standard user interface is used for the *MRML*–based queries, but the contacted server then executes (sub)queries on several other image servers and then transmits the merged results to the user interface.

3.3 Features

Viper employs both local and global image color and spatial frequency features, extracted at several scales, and their frequency statistics in both images and the whole collection. The intention is to make available to the system low–level features which correspond roughly to those present in the human vision system. Most of the following two paragraphs are taken from [264].

3.3.1 Color

It is desirable that the color space used in an image retrieval system should be "perceptually" uniform, meaning that small changes in the color coordinates should correspond to small perceptual differences. The *RGB* space does not have this property. The *HSV* color space offers improved perceptual uniformity, and is easier to compute and invert than systems such as *CIE–LUV* or *CIE–LAB* [250].

Viper uses a palette of 166 colors, derived by uniformly quantizing the cylindrical *HSV* color space into 18 hues, 3 saturations, and 3 values. These are augmented by 4 grey levels. This choice of quantization means that more tolerance is given to changes in saturation and value, which is desirable since these channels can be effected by lighting conditions and viewpoint.

Two sets of features are then extracted from the quantized HSV image. The first is equivalent to a conventional color histogram, with the variation that bins containing zero pixels are discarded. There are thus 166 possible color histogram features, of which most images contain only about 40.

The second class of features represent local color properties of the image. The image is divided into square blocks at four scales, ranging from 16×16 through to 128×128. The mode color is calculated for each block. The occurrence of of a given color in a particular block is treated as a binary feature. For our 256×256 images there are thus $56,440$ possible color block features, of which each image has 340.

Figure 3.7 illustrates the partitioning of the image in blocks with several sizes and locations. All these subblocks at different scales and locations can then be used to extract the local features.

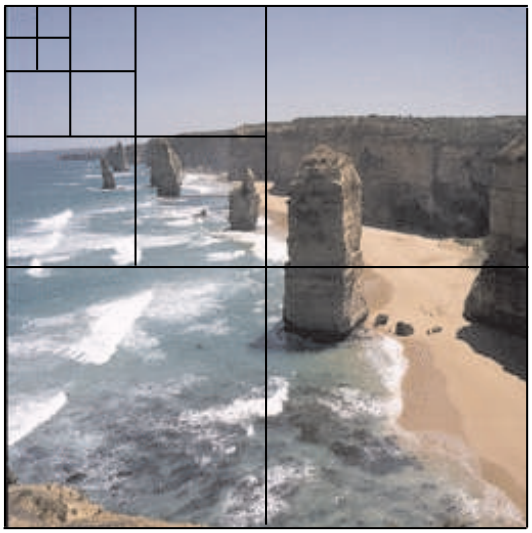


Figure 3.7: The partitioning of an image into blocks of fixed size in *Viper*.

3.3.2 Texture

Two–dimensional Gabor filters have frequently been proposed as a framework for describing and understanding the orientation– and frequency–selective properties of neurons in the visual cortex [54], and banks of Gabor filters have often been applied to texture classification and segmentation [128, 293], as well as more general vision tasks [31, 108, 148]. We employ a bank of real, circularly symmetric Gabor filters, defined in the spatial domain by

$$g_{mn}(x,y) = \frac{1}{2\pi\sigma_m^2} e^{-\frac{x^2+y^2}{2\sigma_m^2}} \cos(2\pi(u_{0_m} x\cos\theta_n + u_{0_m} y\sin\theta_n)), \tag{3.1}$$

where m indexes the scales of the filters, and n their orientations. The center frequency of the filter is specified by u_{0_m}. The half peak radial bandwidth is given by

$$B_r = \log_2 \left(\frac{2\pi\sigma_m u_{0_m} + (2\ln 2)^{1/2}}{2\pi\sigma_m u_{0_m} - (2\ln 2)^{1/2}} \right) \tag{3.2}$$

(after [108]). B_r is chosen to be 1 (*i.e.* a bandwidth of one octave), which then allows us to compute σ_m:

$$\sigma_m = \frac{3(2\ln 2)^{1/2}}{2\pi u_{0_m}}. \tag{3.3}$$

The highest center frequency is chosen as $u_{0_1} = \frac{0.5}{1+\tan(1/3)} \approx 0.5$ so that it is within the discrete frequency domain. The center frequency is halved at each change of scale, which implies that σ is doubled (Equation 3.3). The orientation of the filters varies in steps of $\pi/4$, and three scales are used. These choices result in a bank of 12 filters which gives good coverage of the frequency domain, and little overlap between filters [108]. For practical implementation, filters are truncated at 3σ, giving kernels of sizes 9×9, 17×17 and 35×35.

The use of circularly symmetric filters means that Equation 3.1 is separable. The two–dimensional convolution can thus be computed using four one–dimensional convolutions, which reduces the number of computations required for an $k \times k$ kernel by a factor of order k [108].

These filters are applied to the image, and the mean energy of each filter is computed for each 16×16 block in the image. The energy is then quantized into 10 bands, which were chosen by examining histograms of the filter energy at each pixel for 500 images of the TSR database (see Section 5.2.3). A feature is stored for each filter which has an energy in a band greater than the $[0, 2)$ band. This means that there are 27648 possible such features for a 256×256 image, of which

a given image may have at most 3072 (in practice this does not arise). Histograms of the mean filter outputs are also stored, giving a measure of the global texture characteristics of the image.

Examples for the responses of the Gabor filters we use, of the image in Figure 3.8, can be seen in Figure 3.9. The images show the responses in four directions and at three different scales. To make the actual structures visible, the images are normalized in the end.

(a) Great Ocean Road

Figure 3.8: Example image to demonstrate the response of a set of Gabor filters.

3.3.3 Annotation

Annotation has shown to be an extremely important factor for image search and there is also a project to integrate text into *Viper*. Especially when semantics are to be covered, text is often necessary. A demonstration version is available on the web[12]. More about the integration of text and a preliminary evaluation of the results can be read in [38].

3.3.4 Feature comparison

Although the performance obtained by using different feature groups can not really be compared (since they are meant to be complementary to each other), a short performance comparison using only one feature group at a time is given. In Table 3.1 and Figure 3.10 it can be seen that every feature group alone gives already reasonable results. The details of this evaluation procedure and on the measures used can be read in Chapter 5.

As hoped, the performance is best when all the feature groups are used. We can also see that on a relatively difficult image database (TSR 2500), the color histogram performs best between the single groups, especially at the beginning of the *PR graph*. The values like $P(20)$ shows that also the Gabor histogram has a surprisingly good performance (with this measure, the best). The evaluation speed is not as different for the entire feature set as it is for the single groups which can be due to the fact that large response sets need to be transmitted via *MRML* and thus, network traffic plays an important role.

[12] `http://viper.csse.monash.edu.au/~davids/cgi-bin/PerlMRMLClient/PerlMRMLClient.pl`

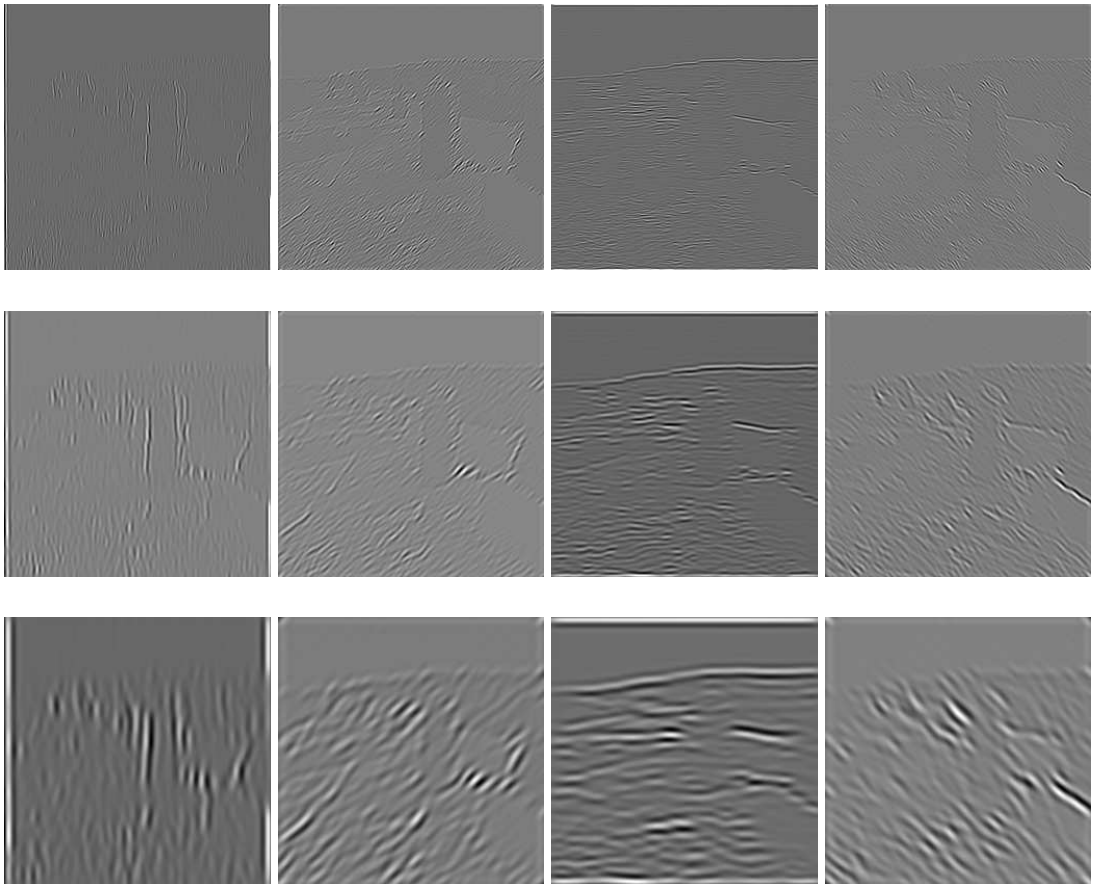


Figure 3.9: Responses of a set of Gabor filters used in *Viper* for the image in Figure 3.8 (normalized for better visibility).

3.4 Feature weighting schemes

The weighting of features (words) based on their frequencies both in the image (document) and in the entire collection is common practice in text retrieval [85, 228]. Two basic principles for weighting features are:

- Features that are frequent in an image describe this image well, which is measured by the term frequency tf.
- Features that are frequent in the collection do not distinguish images well from each other, which is measured by the collection frequency cf.

Based on these well–known facts in text retrieval, several feature weightings were constructed.

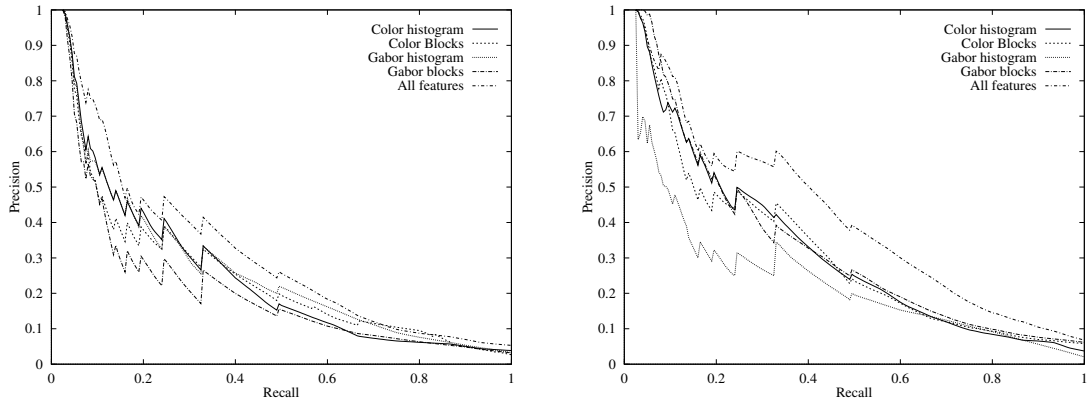
3.4.1 Basic weighting in *Viper*

In our first version, for each feature F_j of the M features in the query q containing the set of query images $\mathcal{Q}$, the list of images containing that feature are retrieved and added to the pool of candidate images. For non–histogram features, starting with $S_k = 0$, the score S_k of each image k is updated for each feature according to:

$$S_{k_{new}} = S_{k_{old}} + tf_{qj} tf_{kj} \log \frac{1}{cf_j}, \tag{3.4}$$

Measure	ColHist	ColBlocks	GaborHist	GaborBlocks	All features
r	14.5	14.5	14.5	14.5	14.5
time t	2.24 s.	1.62 s.	2.25 s.	2.50 s.	2.69 s.
$Rank_1$	44.69	73.02	102.02	153.02	38.71
$R(P(.5))$	0.2367	0.2765	0.2784	0.2038	0.2838
$\bar{Rank}$	598.59	536.78	543.93	730.56	462.37
$\widetilde{Rank}$	0.2363	0.2013	0.2145	0.2865	0.1818
$P(20)$	0.1273	0.1178	0.1417	0.09523	0.1643
$P(50)$	0.07	0.0686	0.0767	0.0510	0.0838
$P(r)$	0.2312	0.2214	0.2349	0.1981	0.2665
$R(100)$	0.3826	0.3691	0.3960	0.3088	0.4774

Table 3.1: Overview of the results for the different feature groups of *Viper* (without feedback) for the TSR2500 database.



(a) before feedback (b) after feedback

Figure 3.10: *PR graph* for the four feature groups used in the *Viper* system.

where cf_j is the frequency of the feature j in the entire database and tf if the term frequency of a feature in either the query or the image. It is worth noting that dimensionality reduction schemes such as Principal Components Analysis (PCA, [204, 210]) can have the effect of eliminating these rare dimensions which can in fact be very useful for creating a specific query. For histogram features, the score is updated according to

$$S_{k_{new}} = S_{k_{old}} + \text{sign}(df_{qj}) \min(|df_{qj}|, df_{kj}) \log cf_j^{-1}, \tag{3.5}$$

which is a weighted variant of the standard histogram intersection.

3.4.2 Comparing several feature weightings

In [262] several of these frequency–based feature weights were compared to optimize the system performance. The weightings were taken from [228].

We investigated the application of weighting functions commonly used in text retrieval to the CBIR domain. The weighting function can depend upon the tf and cf of the feature, as well as its type (block or histogram). We consider a query q containing N_Q images Q_i with relevances $Rel_{Q_i} \in [-1, 1]$. The frequency of feature j in the pseudo–image Q corresponding to the query q is

$$tf_{qj} = \frac{1}{N_Q} \sum_{i=1}^{N_Q} tf_{ij} \cdot Rel_{Q_i}. \tag{3.6}$$

The weighting functions defined in Equations 3.9–3.13 are derived from typical text retrieval term weighting functions [228]. Some modifications were necessary since the image features used can

not always be treated in the same way as words in documents, and especially because we have two different groups of features, histogram and block features.

All weighting functions make use of a base weight

$$wf_{0_{kqj}} = \begin{cases} tf_{qj} & \text{for block features} \\ \text{sgn}(tf_{qj}) \cdot \min\{|tf_{qj}|, tf_{kj}\} & \text{for histogram features} \end{cases}. \tag{3.7}$$

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

$$lf_{1j} = \begin{cases} \log(\frac{1}{cf_j}) & \text{block} \\ 1 & \text{histogram} \end{cases} \quad lf_{2j} = \begin{cases} \log(\frac{1}{cf_j} - 1 + \epsilon) & \text{block} \\ 1 & \text{histogram} \end{cases}. \tag{3.8}$$

ϵ is added to avoid overflows. The weighting functions investigated are:

$$\text{best weighted probabilistic:} wf_1 = wf_{0_{kqj}} \cdot \left(0.5 + \frac{0.5tf_{kj}}{\max_j tf_{kj}}\right) \cdot lf_{2j} \tag{3.9}$$

$$\text{classical idf:} wf_2 = wf_{0_{kqj}} \cdot (lf_{1j})^2 \tag{3.10}$$

$$\text{binary term independence:} wf_3 = wf_{0_{kqj}} \cdot lf_{2j} \cdot \tag{3.11}$$

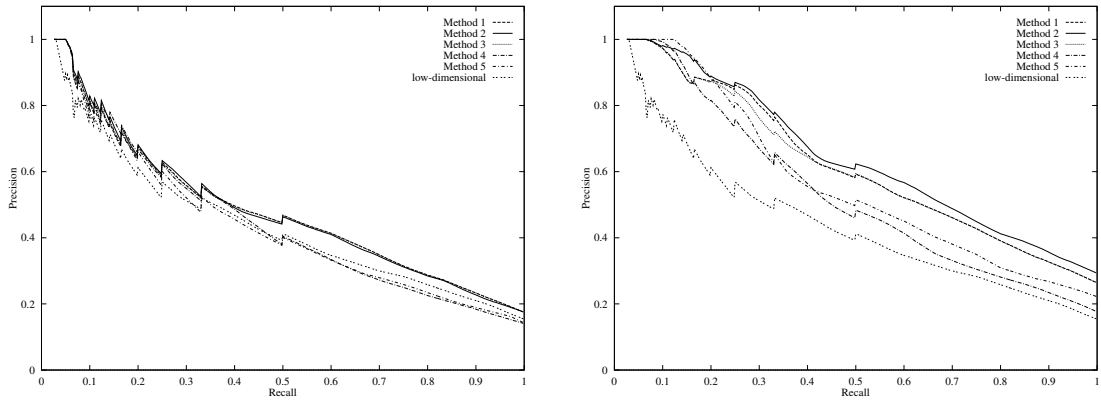
$$\text{standard tf:} wf_4 = \frac{wf_{0_{kqj}}}{\sqrt{\sum_m tf_{km}^2}} \cdot \begin{cases} tf_{kj} \cdot tf_{qj} & \text{block} \\ 1 & \text{histogram} \end{cases} \tag{3.12}$$

$$\text{coordination level: } wf_5 = wf_{0_{kqj}} \tag{3.13}$$

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

$$S^l_{kq} = \sum_j wf^l_{kqj}. \tag{3.14}$$

The images are than ranked on the scores that they obtained.



(a) before feedback (b) after feedback

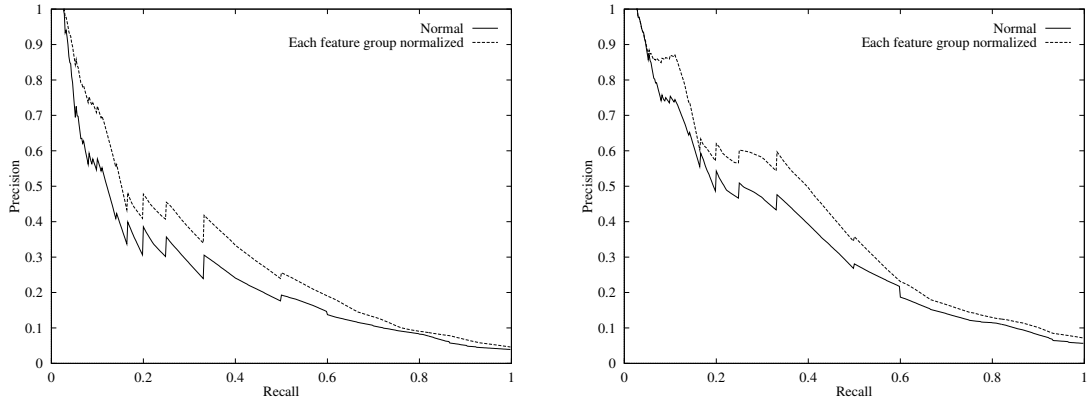
Figure 3.11: *PR graph* for several weighting methods averaged over all users and queries.

In Figure 3.11, the performance of the different weighting schemes can be seen for single–image queries and a first step of feedback. We can see that in the first query step, the results of all the methods are very close, but wf_1 and wf_2 obtain slightly better results than the other weighting schemes. With the use of feedback, we can see that wf_5 starts to be the best, but then drops, which means that it can not be used where high recall is required. In the long run, wf_2 is the

best of the methods, followed by wf_1 and wf_3. As the *classical idf* gets the best results, we use this method as the standard weighting function that is used for the following evaluations of the *Viper* system. The low dimensional system compared with these weighting functions is described in [210].

3.4.3 Separately weighting the feature groups

When doing the parallelization described in Section 3.1.2, one idea was to make a separate query for every feature group and then merge the results. This led not only to evaluate the features separately, but also to normalize the four feature groups separately and then merge the normalized results. The results are normalized by the value that the pseudo–image of the query itself would get as a result for the query.



(a) initial query step (b) with feedback

Figure 3.12: *PR graph* for the standard *Viper* version and for *Viper* using separately normalized feature groups.

Figure 3.12 shows well the improvement in performance of up to 20% when separately weighting the different feature groups. This also solved one of the problems that occurred beforehand. Images with many small–scale textures started to reappear on screen for almost every query. They contained a much larger number of features and thus it was much more likely that they had features in common with the query image(s).

These results can be explained by the fact that separate normalization better takes into account the very different number of the features in the four feature groups. The color histogram has a maximum of 166 values and the Gabor histogram a maximum of 108 values, whereas the number of Gabor blocks can reach up to $3,072$ per image and each image has 340 color blocks. This means that the block features, and especially the Gabor blocks, get a significantly higher weighting when the results are not normalized separately.

3.4.4 Separately weighting positive and negative query components (Rocchio's methodology)

Problems due to too much negative feedback in text retrieval were already addressed by Rocchio in the 1960s [218]. Based on this work, the same techniques to avoid problems with too much negative feedback were used for the *Viper* system. In this way, *Viper* adds up the features of positive and negative query images separately and then averages the positively and negatively added parts according to:

$$tf_j = \frac{\alpha}{n_1} \sum_{i=1}^{n_1} Rel_i \cdot tf_{ij} - \frac{\beta}{n_2} \sum_{i=1}^{n_2} Rel_i \cdot tf_{ij}, \tag{3.15}$$

where tf_j is the weighted tf for features j of the query, n_1 and n_2 are the numbers of positive and negative images in the query respectively, Rel_i signifies the relevances given by the user (1 for positive ones and -1 for negative ones), and α and β determine the relative weightings of the positive and negative components of the query. We use $\alpha = 0.65$ and $\beta = 0.35$ because this was one of the recommendations for text retrieval.

This means that even when many images are marked as negative feedback, a feature can only become 0 or even negative, when the average among the images marked non–relevant is higher than the average among the images marked relevant. Thus, α and β could also have other values such as $\beta = 0.5, \alpha = 0.5$. This would strengthen the influence of the images marked non–relevant, but on the other hand can easily eliminate good features. A comparison of the performance of the Rocchio feedback with other feedback techniques can be found in Section 4.5.1.

3.4.5 Additional feature weighting factors learned from user interaction

So far, the relevance feedback has always been used to create a new query and the features were solely weighted upon their frequencies in the query images. The fact that user interaction files in *MRML* are stored for every interaction with *Viper*. This makes it logical to exploit the stored user interaction data. This section only gives a short overview. In Section 4.6.1, a more profound, detailed analysis can be found that relates to [171, 175, 182].

The basic principles to learn from the user interaction data of multiple–image queries are based on the following assumptions:

- Images that are marked as relevant in the same query step have something important in common.
- If one image is marked non–relevant and another one relevant in the same query step then the similarity of the two images is not very important.
- When two images are marked as non–relevant in the same query step, it cannot be said if their similarity is important or not as they can be marked non–relevant for completely different reasons.

Figure 3.13 demonstrates the different ways that images can be marked with in the same query step. The "+" means that an image is marked as relevant and the "–" that the image is marked as non–relevant. The green ellipse thus means that the connection between the two images is a positive connection whereas the red ellipses mark negative connections. The non–relevant, non–relevant connections (marked with a dashed polygon) can not be used for our algorithm because images can be marked non–relevant for completely different reasons.

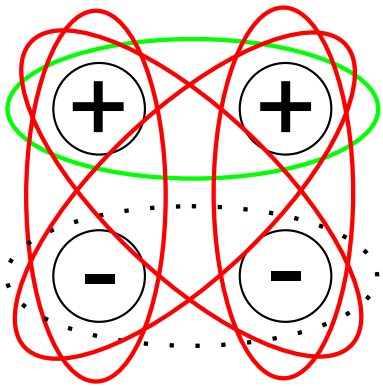


Figure 3.13: Different ways to mark images in the same feedback step and how these marks can be used for learning.

As we do not want to learn on an image, but on a feature basis, we analyze the positive and negative pairs for the features that they have in common.

Like this, each feature gets a number of positive and negative marks that can be used to calculate a simple factor that we define to be between 0 and 1, where 1 is for features who are only in positive connections and 0 for only negative connections. 0.5 should be the value for features that are as often in negative as they are in positive connections. This leads to the following simple formula for an importance factor a_j for a feature j:

$$a_j = 1 + \frac{p_j}{p_j + n_j} - \frac{n_j}{p_j + n_j}, \tag{3.16}$$

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

If we recall our standard feature weighting *idf* (see Equation 3.9) we come to the weighting:

$$h_{qj} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n}\left(tf_{ij}\cdot Rel_i\right)\cdot log^2\left(\frac{1}{cf_i}\right), & \textit{for block features} \\ \frac{1}{n}\sum_{i=1}^{n}\left(tf_{ij}\right), & \textit{for histogram features} \end{cases} \tag{3.17}$$

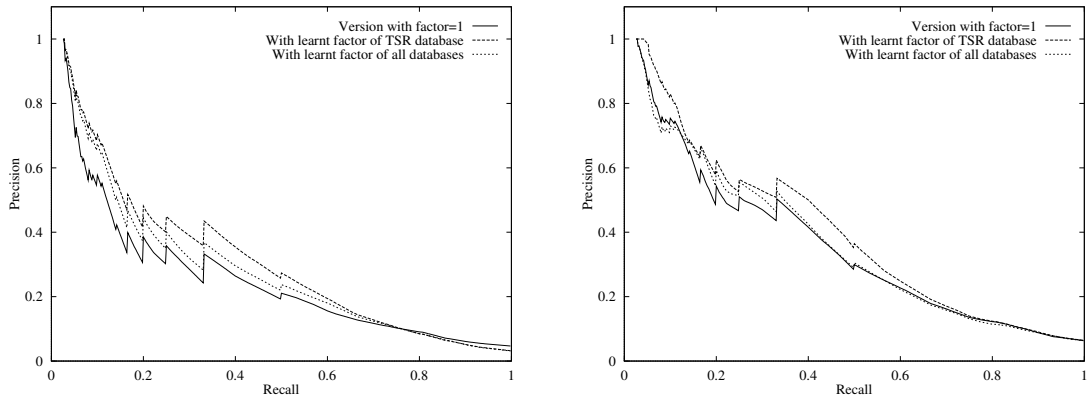
$$S_{kq} = \sum_j \left(\begin{cases} h_{qj} & \textit{for block features} \\ \text{sign}(h_{qj})\min(|h_{qj}|, df_{kj}) & \textit{for histogram features} \end{cases} \right), \tag{3.18}$$

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

To include the importance factor a_j for each of the features, we simply multiply our feature relevance h_{qj} with this factor as can be seen in:

$$h_{qj}^{new} = a_j \cdot h_{qj}. \tag{3.19}$$

The sum is then calculated in the same way as in Equation 3.18. The study described in [175] lead to the following results, when the factor was once calculated over all the queries that were done with the same database and once with the queries done with any database.



(a) without feedback (b) with feedback

Figure 3.14: *PR graph* for the results with a learned factor, once learned over the queries with the same database and once over all queries done with the system.

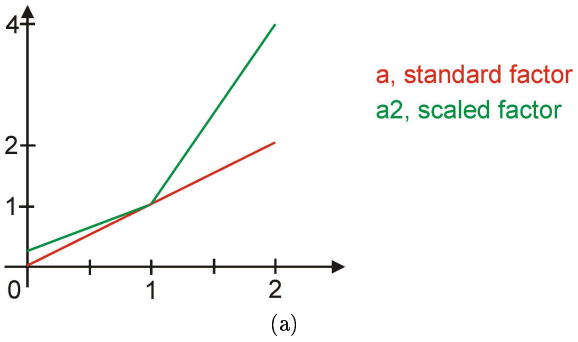
In Figure 3.14, we can see that the results improve strongly with the learned factor. This is particularly the case when the results are learned over queries with the same database where the results improve up to 15% with and without feedback.

To have a comparison, a second factor was calculated, where basically the first factor was rescaled as can be seen in Equation 3.20. The idea is to still have a factor of 0.5 for images that are as often in positive and negative connections, but to strengthen the positive part of the factor.

The negative part of the factor is slightly weakened, so negative values do not completely disappear from the calculation with negative feedback.

$$a2_j = \begin{cases} 0.25 + 0.75a_j & : \quad a_j < 1 \\ 1 & : \quad a_j = 1 \\ 1 + (a_j - 1) \cdot 3 & : \quad a_j > 1 \end{cases} \tag{3.20}$$

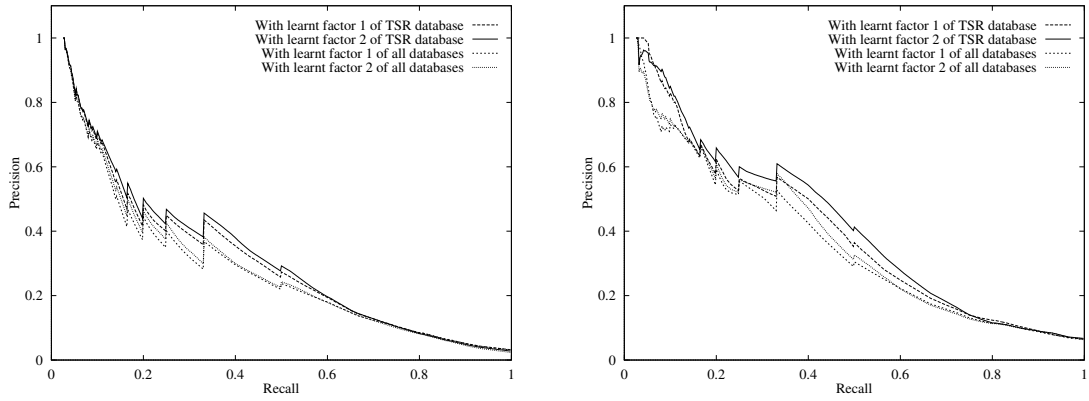
The difference in scaling of the two factors is also illustrated in Figure 3.15.



(a)

Figure 3.15: Comparison of two learned factors.

This second factor $a2_j$ is subsequently compared to the first one in Figure 3.16.



(a) without feedback (b) with feedback

Figure 3.16: *PR graph* for the results with two different, learned factors.

We can see well, that the second factor $a2_j$ delivers even better results than the first one as with the scaling the positive features get a stronger influence. The results are up to 5% better than with the first factor. A more profound study on the use of usage log files can be found in Section 4.6.1. This section make a comparison with the related field of *market basket analysis* that deals with a similar problem.

3.5 Evaluation of the *Viper* system

This section presents a performance comparison of the current *Viper* system on four different image collections with different relevance sets. A short analysis explains the resulting measurements. In Chapter 5 the performance measures and the test setup for the generation of relevance feedback is explained in detail.

It can be seen that the results depend strongly on the size $N_{\mathcal{I}}$ and the difficulty of the image collection and of the size of the relevance sets $N_{\mathcal{R}}$. For each image database we chose a set $\mathcal{E}$ of $N_{\mathcal{E}}$

evaluation images and then relevance judgments were obtained for the query images. Sometimes a single database grouping was used but for the TSR image databases, real user judgments were obtained.

3.5.1 TSR 500 database

This image collection is described in detail in Section 5.2.3. In Table 3.2 and Figure 3.17, the performance measurements for the retrieval results of the evaluation with the TSR 500 database can be seen.

Measure	no RF	RF 1	RF 2	RF 3	RF 4
r	10.56	10.56	10.56	10.56	10.56
time t	0.39 s.	1.19 s.	1.50 s.	1.76 s.	2.45 s.
$Rank_1$	13.32	9.1	7.58	7.48	7.42
$R(P(.5))$	0.3966	0.4082	0.4116	0.4007	0.4318
$\underline{Rank}$	59.61	50.53	48.72	48.87	48.33
$\widetilde{Rank}$	0.1085	0.0904	0.0867	0.0870	0.0860
$P(20)$	0.27	0.307	0.324	0.33	0.333
$P(50)$	0.1532	0.1592	0.1588	0.1592	0.1576
$P(r)$	0.4925	0.6330	0.6895	0.7013	0.7133
$R(100)$	0.7956	0.8289	0.8138	0.8128	0.8064

Table 3.2: Overview of the results for *Viper* when using the TSR 500 database.

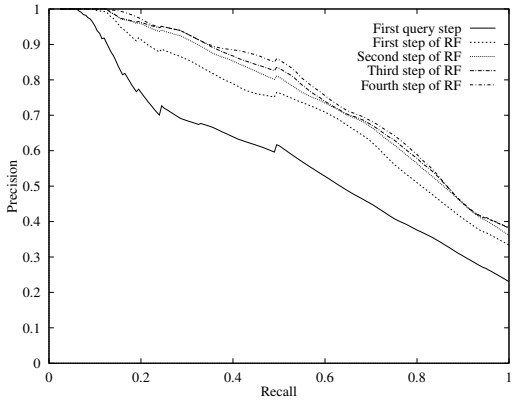


Figure 3.17: *PR graph* for the TSR 500 database.

We can see in Figure 3.17 that especially the first step of feedback results in a large improvement in the *PR graph*. Further steps still bring an improvement, but a much weaker one than in the first step. From the third to the fourth step of feedback the results seem to be saturated and do almost not improve anymore. When we take a look at Tables 3.2 we can see that here, the performance improvement in the first step seems to be much smaller as the $P(20)$ only rose from 0.27 to 0.307. Most of the improvement we could see in the *PR graphs* was thus due to the reordering in the top images, and actually, not many new relevant images appeared on screen. The response times t for one–image queries are under one second and thus very fast. Even with many query images the times are still rather low, even when transmitting the entire 500 image URLs to the user. When only transmitting 20 images (as in a "normal" user scenario), the response times will even be faster. The $P(50)$ and $R(100)$ values show that only in the first images that are actually used for feedback, the results are becoming better. Further on the results do not improve much, even with feedback.

3.5.2 TSR 2500 database

The TSR 2500 database is described in detail in Section 5.2.3. We can see in Table 3.3 and Figure 3.18 that the results for this larger database are on a different, lower level and do look much worse. Total recall is not at 40% precision as with the TSR 500 database, but rather at 10% precision. Also the query times go up significantly. This is not only due to the longer computation, but also because of the longer transmission of the 2500 query results via *MRML*.

Measure	no RF	RF 1	RF 2	RF 3	RF 4
r	14.5	14.5	14.5	14.5	14.5
time t	2.69 s.	4.54 s.	5.12 s.	9.76 s.	34.04 s.
$Rank_1$	38.71	17.26	18.40	19.33	19.33
$R(P(.5))$	0.2838	0.3810	0.3862	0.4058	0.4196
$Rank$	462.37	458.25	448.38	460.52	483.30
$\widetilde{Rank}$	0.1818	0.1802	0.1762	0.1811	0.1898
$P(20)$	0.1642	0.1976	0.2155	0.2274	0.2380
$P(50)$	0.0838	0.0976	0.1052	0.1048	0.1052
$P(r)$	0.2665	0.3699	0.3896	0.4118	0.4252
$R(100)$	0.4774	0.4938	0.5248	0.5307	0.5291

Table 3.3: Overview of the results for *Viper* when using the TSR 2500 database.

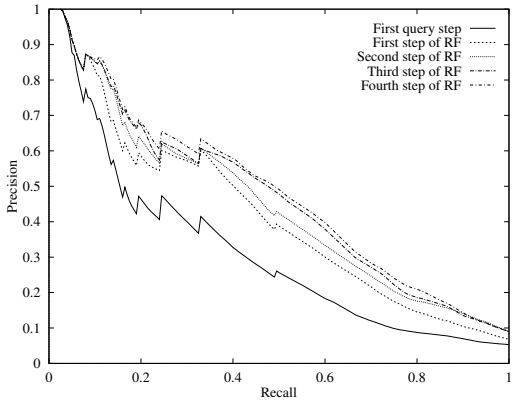


Figure 3.18: *PR graph* for the TSR 2500 database.

We can see that the improvement due to relevance feedback is stronger than with the TSR 500 database. Anyway, not for every query image can a relevant image be found in the top 20 Even after four steps of feedback it still remains like this. This can be seen from the $Rank_1$ measure that is far higher than 1. The $P(20)$ improves over every step of relevance feedback, as well as the $P(r)$. The later measure $P(50)$ and $R(100)$ improve in the first two steps of RF, but then more or less stalled. This is, in part, due to the fact that only the first 20 images are used for the generation of positive and negative feedback.

3.5.3 Washington database

This database is described in detail in Section 5.2.3. Table 3.4 and Figure 3.19 give an overview of the results with the Washington database.

We can see that the results look much better than with the TSR 2500 database, although not as good as with the TSR 500 database. The Washington database seems to be a fairly easy database. We can also see that from the start on $Rank_1$ is next to perfect and for every query relevant images could be found. This means that also positive images are available for relevance feedback, and we can see that the results with feedback get much better, especially in the first two steps of feedback.

Measure	no RF	RF 1	RF 2	RF 3	RF 4
r	65.14	65.14	65.14	65.14	65.14
time t	1.23 s.	2.18 s.	2.49 s.	2.62 s.	2.70 s.
$Rank_1$	1.5	1	1	1	1
$R(P(.5))$	.3798	.5520	.6718	.6594	.7049
$\bar{Rank}$	176.44	152.28	116.13	107.04	104.37
$\widetilde{Rank}$	.1573	.1308	.0911	.0811	.0783
$P(20)$	.5392	.7357	.8642	.8892	.9107
$P(50)$	.4057	.5271	.6085	.6328	.6257
$P(r)$	.3883	.5256	.6138	.6640	.6553
$R(100)$	.4839	.6070	.6924	.7279	.7208

Table 3.4: Overview of the results for *Viper* when using the Washington database.

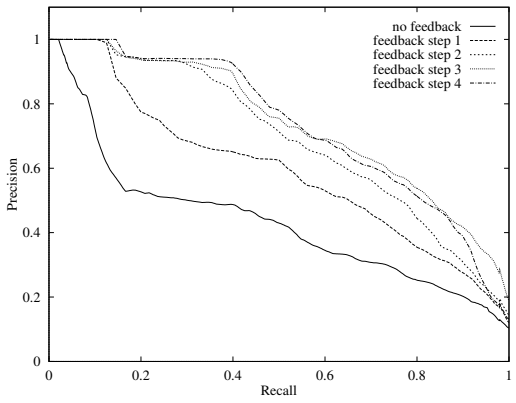


Figure 3.19: *PR graph* for the Washington database.

The precision values show how much easier this database is. $P(20)$ gets up to 90% after four steps of feedback whereas the same value for the TSR 2500 database is 23%. This is also due to the fact that the Washington database contains a much larger number of relevant images. This means that the value for $P(50)$ can theoretically almost reach the 100% mark whereas for the TSR 500 database even with perfect retrieval, the $P(50)$ can only reach a maximum of 20% because there only is an average of 10 relevant images.

3.5.4 Corel database

A detailed comparison of different methods for evaluating image sets from the Corel Photo CDs is done in Section 5.7. This shows the different problems that can arise from various evaluation methods and how hard it is to even compare the performance of systems based on the same data sets, when the query tasks and the relevance judgments change slightly.

3.5.5 Comparison of the evaluations

When comparing the performance measures of the same system on various databases it can clearly be seen that the databases have a very large span of difficulty. Although some measures such as the *generality*, defined in Equation 5.3, should give means to compare system over databases of various sizes and with differing relevance sets, it will be hard to really compare two different systems based on completely different databases. It can also be seen that on some databases, relevance feedback brings a huge performance gain over several steps whereas on other databases (like the TSR 500) the performance gain is mostly due to a reordering of the first $N = 20$ result images.

We can also see the influence of the number of images in the database and the number of relevant images on the performance measures. The use of the *generality*–based measures (see Equation 5.3) can reduce these influences, but on the other hand Section 5.7 also shows that there are many other influences that might change the performance values. This makes clear that the comparison over different image databases is very hard or even impossible and enforces the need for a standard image database with query tasks and relevance judgments to do standardized evaluation. The description of the *Benchathlon* in Section 5.2.3 describes other important factors needed for proper benchmarking.

3.6 Summary

This chapter describes and evaluates our current CBIRS *Viper* and the underlying techniques used. It shows the distributed structure of the program and especially the constitution of components that can easily be replaced within the framework. This architecture allows to adapt the program to different environments rather easily and quickly.

This chapter also relates the development of *Viper* over time and the different factors that lad to performance enhancements. An evaluation on several image databases and a critical reading of the results is done at the end. This shows how hard it is to compare a system when evaluations were not done on the same database using the same query images and the same relevance judgments. It also shows what can be said about the system performance by taking a close look at the performance measures.

Chapter 4

User Interaction in Visual Information Retrieval

> Realism is relative, determined by the system of representation standard for a given culture or person at a given time.
>
> Nelson Goodman

The interaction of a user with a visual information retrieval system contains much more than only the output of the system's response to the user and the possibility for the user to select relevant and non–relevant images using relevance feedback.

This chapter analyzes the entire interaction chain, including the influences of *response times* and other *usability issues*. It explains the problem of finding a good *query starting point*, and using appropriate *query paradigms* for the interaction with the user. Of course, *user interfaces* and the presentation of the results in general are discussed, and *relevance feedback* that can enhance the performance of retrieval systems enormously.

Relevance feedback is frequently only regarded from one query step to the next one. For every query step a new query is formulated and then executed without taking into account the entire query sessions. With relevance feedback or learning over several temporal scales, a much higher performance gain can be achieved. A *hierarchy of learning* finally regards the possibility to not only learn over all the users of a system but also to learn hierarchically on several levels. This can mean to learn on a *user* level as certain users might have repetitive search tasks or certain preferences for queries they execute. The second level can be a *user group* or a certain database where users might pursue similar tasks or at least work on similar features. The last level can be an *overall* level, where over all users and database can be learned, although this will be the most difficult level as users might have different preferences and the work can be done on very different databases.

[302] gives a good overview of relevance feedback and identifies several kinds of queries and their particular characteristics, but does not talk much about the actual interaction. Other good overview articles on relevance feedback are [221, 222, 300].

4.1 Usability in image retrieval

Nielsen [194] gives a good introduction to usability issues in software systems. Another good book on human–computer interaction (HCI) is [209]. Most of these usability issues concern the actual user interfaces but in this thesis we will mainly concentrate on the interactivity part of the usability and more specifically on interaction speed.

4.1.1 Interaction speed

When publicly presented, most CBIRSs do not state actual response times t of their systems, although it was shown in many usability studies over the past thirty years that with respect to the usability and interactivity of a system, the response time is extremely important. In [194], the following reference time values t are stated for interactive systems:

- 0.1 seconds is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.
- 1 second is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 and less than 1 second, but the user does lose the feeling of operating directly on the data.
- 10 seconds is about the time to keep the user's attention focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

These times have been verified in several tests cited in [194] and, Sharon Flank [70] cited a maximum of $t = 1 - 2$ seconds for the response time of retrieval systems on the WWW.

We can thus see that many systems that actually state their query response times ([39]=*5 s.* for 1100 images; [306]=*4 min*) are well above the limit for the user's flow of thought to stay uninterrupted and even further away from giving the user the feeling of instantaneous reaction.

While we can always wait for faster computers to solve the problem, it is important to at least stay below the 1 second limit or to leave the choice to the user whether he rather wants to have a quick reply or more accurate results. The trade–off between the quality of the results and the interaction speed has to be analyzed carefully. Big search engines on the web all use methods to prune their search (see also Section 4.1.2), knowing that this can lead to rather unstable search results at different times of the day as the load varies and pruning has a more or less strong effect. For text search engines on the internet such as Google or Altavista, often 0.5 seconds is the time that is foreseen to be used for querying and approximately the same amount of time to transmit advertisement banners.

Many authors propose methods for reducing the time needed to search the feature space, such as dimensionality reduction using *Principal Components Analysis* (PCA) [210, 242], *clustering* [109, 280], or spatial search structures such as *K–D–trees* [242, 296]. 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 limit search time rather by pruning the number of images for which distances are calculated.

As *Viper* works with inverted files and other techniques known from text retrieval, this study is limited to the implementation and evaluation of techniques known from text retrieval and sensible for our system.

4.1.2 Search pruning

Knowing the enormous importance of interaction speed for the usability of a system, it sounds logical to explore ways to improve the speed of the system to at least stay beyond the limit of *1 s*, so the user's flow of though will stay uninterrupted. More information about pruning with the *Viper* system can be found in [181, 260].

Many basic ideas for techniques for pruning with inverted file accesses have been based on techniques described in [299]. Two basic principles are retained:

- Do not evaluate all *features* present in the query.
- Do not evaluate all features for all possible response *documents* (in our case images).

We also analyzed lossless pruning methods where images were, for example, excluded from the result set when they did not have a chance to get into the top N requested images anymore. Unfortunately, these techniques did not improve the response time at all as the additional time for checking the limits was about as much time as was saved by not evaluating certain features.

Data analysis

Before explaining the techniques for search pruning, it is sensible to take a look at the data and identify which part of the retrieval system consumes most of the time in the query process. The data were collected when using the TSR 500 database containing 500 images, described in Section 5.2.3. The times referenced in this section were measured on a Sun Ultrasparc 10 with 128 MB of RAM. Later methods relied on other hardware platforms that are always mentioned in the corresponding sections. The evaluations for the TSR 2500 database in this section were done with the same hardware but upgraded to 256 MB of RAM.

The time t taken to compute a response depends on the particular query or feedback image(s) since images have a varying number of features (between $1,000$ and $2,600$ in our database). Typical response times when all features are evaluated for typical queries are *0.85 s.* for a single image query with $1,667$ features and *1.61 s.* for a three–image query with $4,224$ features. Since these response times are much longer than the "feeling of instantaneous reaction" goal discussed in Section 4.1.1, the query evaluation process was analyzed in more detail. The results appear in Table 4.1.

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 4.1: Breakdown of query evaluation times for two example queries.

It is clear that the operation for which there is the most scope for improvement is the access to the inverted file and the calculation of the scores for the feature/image combinations.

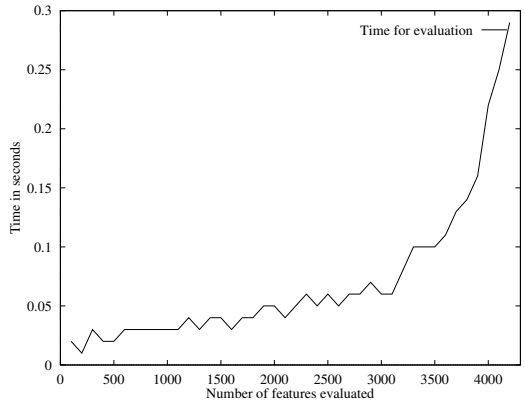
In our case, the time taken to evaluate a feature depends on its collection frequency cf as the lists of documents containing a feature gets longer with a higher cf and results in a larger number of updated similarity scores. Figure 4.1(a) illustrates the time it takes to evaluate 100 features at different stages of the query, when the features are sorted by their importance to the query (based on cf and tf). For a database with 500 images they vary typically between *0.01 s.* and *0.3 s.* and we can see that the last features with a high cf take considerably longer. The evaluation of query times on a per feature basis was not possible in out experiments due to the low resolution of library to measure the times.

Figure 4.1(b) shows well that the raise in evaluation time of Figure 4.1(a) corresponds almost exactly to the higher cf.[1]

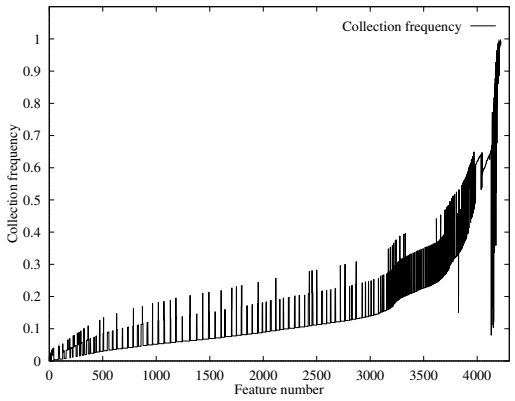
When evaluating these response times for the larger TSR 2500 database described in Section 5.2.3, the differences get even more extreme. We thus used a finer resolution and evaluated the time it takes to evaluate 50 features for a one–image query with $2,674$ features and a multiple–image query with $9,283$ features. The results can be seen in Figure 4.2.

These figures shows well that the first features are evaluated extremely fast, in around *0.01 s* for 50 features. The last features are more expensive as they are contained in a larger number of images. Here the evaluation time is up to 50 times as high for the evaluation of 50 features.

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

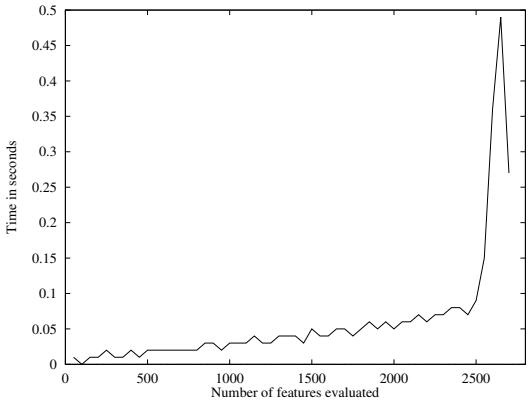


(a) time to evaluate 100 features at different stages of the query process

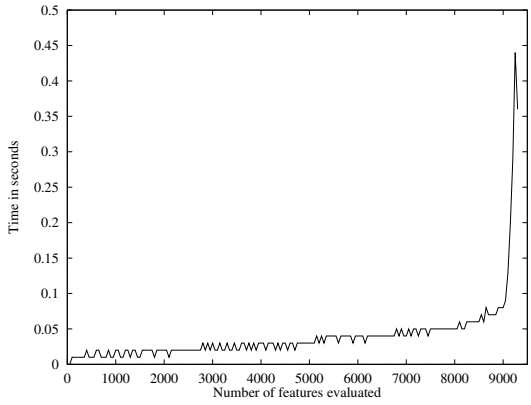


(b) cf of the features

Figure 4.1: Evaluation times for each set of 100 features for a query with three query images containing 4,224 features, and the corresponding collection frequencies of the sorted features.



(a) single image query



(b) multiple image query

Figure 4.2: Evaluation time for 50 features for a query with 2,674 features and a feedback query with 9,283 features.

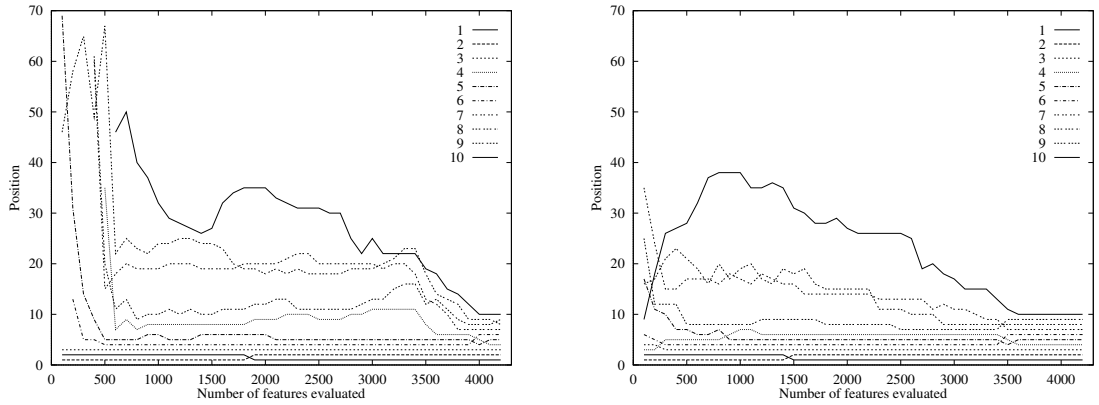
Methods for reducing the query evaluation time

The methods that are used for reducing query evaluation time fall into two philosophically different classes. Evaluation can either be stopped at any 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.

Figure 4.3(a) shows the ranks of the images that are finally ranked 1 to 10 during the evaluation of a query (using the TSR 500 database). 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.

If the features are sorted according to their weights (as in Figure 4.1(a)), the results are better, as shown Figure 4.3(b). 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.

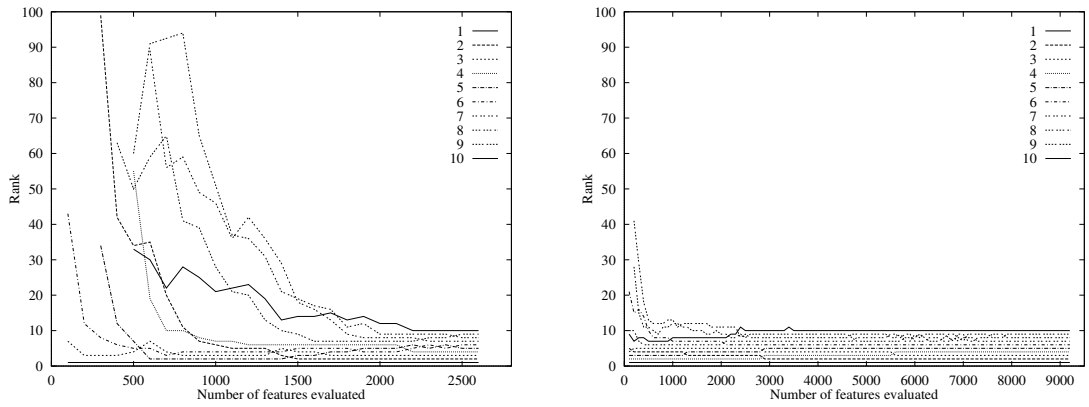
Figure 4.4 shows the same statistics for a one–image query and a multiple–image query for the TSR 2500 database. We can see well that when using a larger number of features as this is the



(a) without sorting (b) with sorting

Figure 4.3: Ranks of the final top 10 images during query evaluation, without and with features sorted by their score to a certain query (based on tf and cf).

case in feedback queries, the final order becomes clear much earlier, so the cutoff can already be after 20% of the features have been evaluated.



(a) initial query (b) with feedback

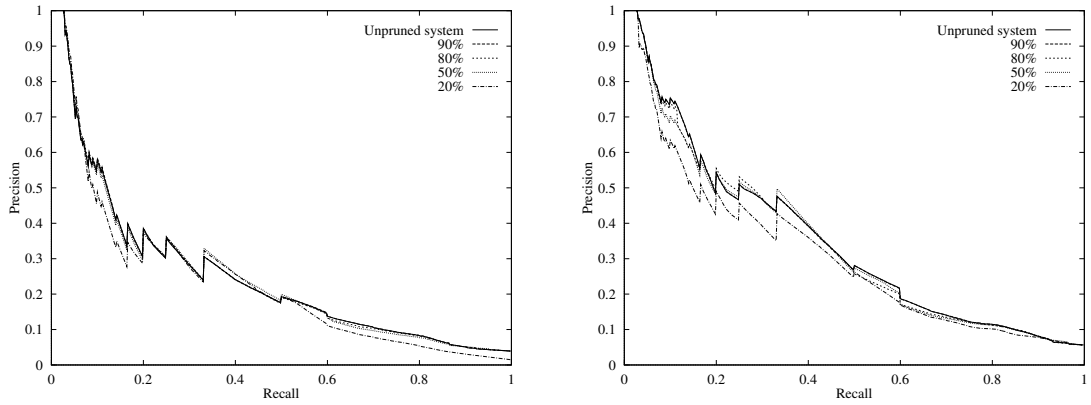
Figure 4.4: Development of the ranking of the final top ten images.

It is thus 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 cf that take the longest time to evaluate.

Feature pruning

Figure 4.4 shows that the final top ten images were in the top ten already quite early in the query evaluation. This suggests that the query evaluation could be stopped after a certain percentage of features without seriously affecting the retrieval performance. Since the low–weighted features are very computationally expensive, we can expect a better than linear reduction in computation time. To evaluate this method, performance was evaluated for cutoff points of 20%, 50%, 80% and 90%.

Figure 4.5 shows that only the curve for cutoff after 20% of features is notably worse that the unpruned case. The other cutoffs even give slightly better performance in some parts of the *PR graph*, although not significantly. This may be understood as the fact that some of the very frequent features essentially act as noise. We can also see that the use of (only positive) relevance



(a) initial query (b) with feedback

Figure 4.5: *PR graph* for cutoff after various percentages of features evaluated (feature pruning).

feedback does improve the effect that a slight pruning seems to lead to better results in the *PR graph*.

Cutoff point	Average evaluation time t	% of time of the unpruned version
20%	0.09s	4%
50%	0.28s	14%
80%	0.72s	36%
90%	0.96s	48%
unpruned	2.02s	100%

Table 4.2: Averaged evaluation times for cutoff after various percentages of features are evaluated.

Table 4.2 shows the large improvements in evaluation time obtained from this pruning technique. By reducing the number of features evaluated by only 10%, the evaluation time is more than halved. Evaluating only 20% of the features takes less than 5% of the full evaluation time, but the performance is significantly worse than that of full evaluation. This mode still might be useful if an extremely quick response is needed, or if the the user just wants to browse the images – some noise may perhaps even be desirable. For example [151] showed that quick browsing plays a very important role in image search performed by journalists. Journalists often have very strong time constraints and tend to search for an acceptable rather than a perfect result.

Pruning with an upper time limit

Pruning with an upper time limit technique is basically a subset of the feature pruning, where the evaluation is stopped after a fixed time, so the percentage of features evaluated varies depending on the query. The results depend strongly on the databases used, as the evaluations for larger databases in general take longer.

In Figure 4.6, we show results for the TSR 500 database with different time limits. This *PR graph* shows well that such a method is feasible and that the possibility can be given to a user to chose between accuracy and interaction speed without too great a loss in precision.

It can be seen that none of the limits causes a significant reduction in precision. The performance with the *0.5 s.* limit is indistinguishable from the unlimited case. One should be careful in generalizing this result as multiple image queries in very large databases might lead to a significant deterioration in quality with fixed response times.

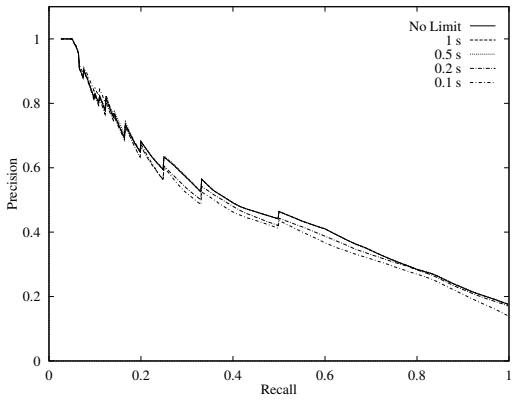


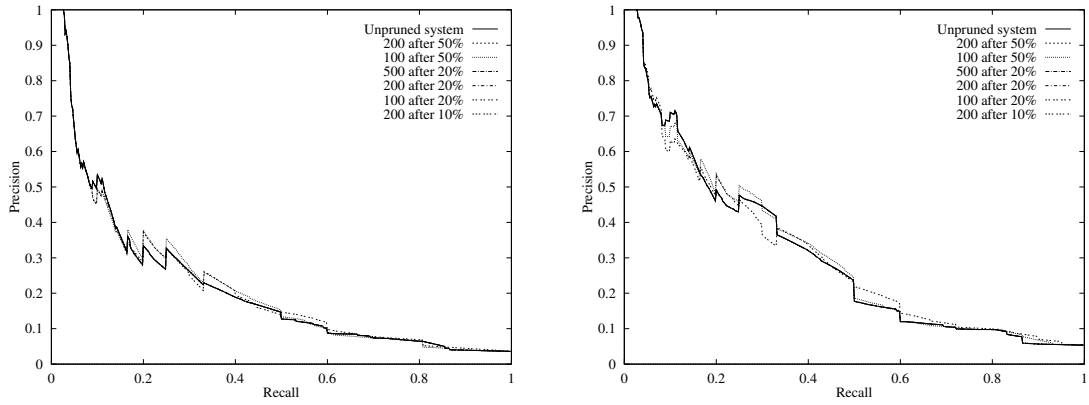
Figure 4.6: *PR graph* when the evaluation is stopped after a fixed time.

Scoreboard pruning

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

In order to be able to compare the performance of the pruned searches with that of the non–pruned version, the scores for all images must be calculated, and the resulting list then reduced to the same number as the pruned versions. The *PR graphs* are continued using the expected precision for randomly distributed responses after this point. Here, all lists were reduced to 100 images before comparison. Non–complete retrieval can make *PR graphs* unreliable, *e.g.* after around 40% recall as reported in the TREC context [91].

Figure 4.7 shows that, in most cases, the pruned versions do not perform worse than the unpruned system. Only the reduction to 100 after 10% of the features were evaluated is noticeably below the original curve.



(a) initial query (b) with feedback

Figure 4.7: *PR graph* for scoreboard pruning.

Table 4.3 shows that time savings of up to 30% can be achieved, with very little loss in accuracy. As would be expected from Figure 4.2, the evaluation time depends on the percentage of the features which are evaluated before the cutoff as well as on the number of images to which the list is reduced.

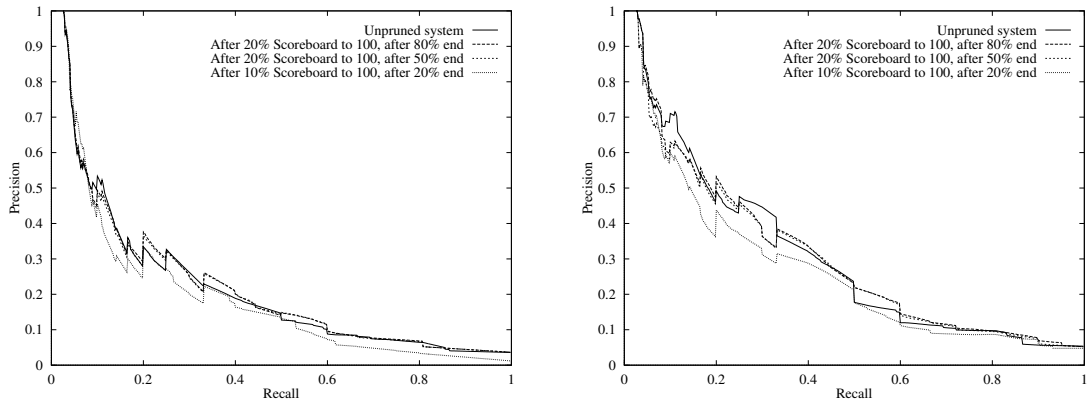
No. images retained	Cutoff point	Average evaluation time t	% time of unpruned
100	20%	1.38s	68%
100	50%	1.45s	72%
200	10%	1.41s	70%
200	20%	1.43s	71%
200	50%	1.47s	73%
500	20%	1.50s	74%
unpruned		2.02s	100%

Table 4.3: Averaged evaluation times for scoreboard pruning.

We would suggest that a reduction of the scoreboard to 100 images after 20% of the features have been evaluated as a suitable trade–off between speed and accuracy (based on the times stated in Section 4.1.1 and the numbers of pictures that users normally look at as described in Section 5).

Combining methods

The easiest combination of pruning methods to implement is that of scoreboard and feature pruning. Figure 4.8 shows that only the curve for cutoff after 20% of the features is significantly worse than that corresponding to the evaluation of all features.



(a) initial query (b) with feedback

Figure 4.8: *PR graph* for various combinations of scoreboard and feature pruning.

The differences in performance between the combination of scoreboard and feature pruning and feature pruning alone are thus very small whereas, as shown in Table 4.4, the gain in speed is between 10% and 30%, which is significant.

Pruning method	Average evaluation time t	% time of unpruned
scoreboard to 100 after 20%, cutoff after 50%	0.23s	11%
cutoff after 50%	0.28s	14%
scoreboard to 100 after 20%, cutoff after 80%	0.52s	26%
cutoff after 80%	0.72s	36%
scoreboard to 100 after 10%, cutoff after 20%	0.08s	4%
cutoff after 20%	0.09s	4%
unpruned	2.02s	100%

Table 4.4: Comparison of evaluation times for a combination of pruning methods.

The combination of the parallel access methods with the pruning methods could offer further improvements. Each feature group could be pruned separately. This has a great potential since

pruned to	10%	20%	50%	100%
r	14.5	14.5	14.5	14.5
time t	1.37 s.	1.82 s.	2.14 s.	2.69 s.
$Rank_1$	33.66	33.57	35.88	38.71
$R(P(.5))$	0.3111	0.3172	0.3233	0.2838
$\overline{Rank}$	475.11	469.13	460.21	462.37
$\widetilde{Rank}$	0.1869	0.1846	0.1810	0.1818
$P(20)$	0.1643	0.175	0.1654	0.1643
$P(50)$	0.0843	0.0857	0.0867	0.0838
$P(r)$	0.2637	0.2661	0.2664	0.2665
$R(100)$	0.4780	0.4839	0.4934	0.4774

Table 4.5: Performance measures for the actual *Viper* pruning.

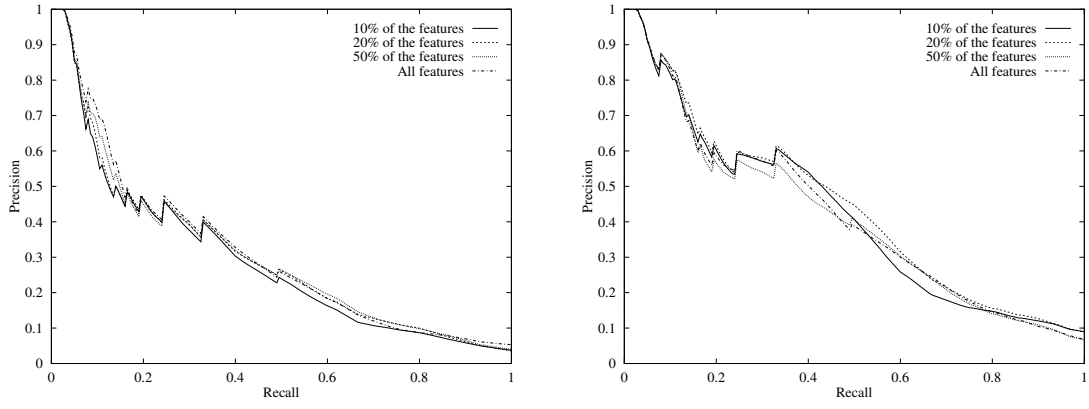
the different feature groups have very different collection and document frequency characteristics and thus should be pruned in different ways.

Scoreboard reduction is especially important for the parallel version described in [181] because it shortens the lists which need to be transmitted over the network, making the system speed less dependent on network traffic.

Other pruning ideas used in *Viper*

Viper, in its most recent version, uses a more sophisticated form of weighting the features then the first versions. The construction of the pseudo–image is more complex as positive and negative features are added separately. They are subsequently combined (Rocchio, see Section 3.4.4). Much more effect on the pruning has the fact that the four feature groups *Viper* uses (color histogram, color blocks, Gabor histogram, Gabor blocks) are executed separately and the results of these four queries are then normalized and merged afterwards.

Studies with separate pruning of the four groups showed, that pruning of the histograms led to bad results as the color and Gabor histograms contain only a small number of features and they all seem to be important. We thus use pruning on the block features only and apply the pruning separately for color and Gabor blocks. Due to the separate normalization and the merging of the result lists, this form of evaluation is not as fast as our former evaluations. For this evaluation we used our benchmark tests described in Section 5.3, so the time results t include computation time to transmit the queries and the results in *MRML* via the network.



(a) initial query (b) with feedback

Figure 4.9: *PR graph* for the actual pruning in the *Viper* system.

Figure 4.9 and Table 4.5 show how much faster the queries can be executed thanks to the pruning while preserving the quality of the results. When using 50% of the block features, the

precision measures $P(20)$ and $P(50)$ even get better, underlining the assumption that some features can essentially be regarded as noise

4.1.3 Other usability issues

Of course we cannot limit the list of usability factors to interaction speed only. The presentation of the results in a user interface (see Section 4.4.1) also plays an important role. In [194], the following dimensions of usability are mentioned and described as follows:

- *Learnability*: The system should be easy to learn so that the user can rapidly start getting some work done with the system.
- *Efficiency*: The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.
- *Memorability*: The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything allover again.
- *Errors*: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them.
- *Satisfaction*: The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

All these dimensions are also very important in the field of visual information retrieval. Especially when building real systems that are supposed to be used by non–computer scientists all these points have to be taken into account.

4.2 Query starting point

The *query starting point* is in a very close connection with the *interaction paradigm* or query refinement, detailed in Section 4.3, as the query start is already the first part of the interaction with the user. Here, we decouple the problem of a good query starting point from the refinement process or interaction paradigm (see Figure 4.10). It is, for example, well possible (and often done in image search engines on the web) that the starting point of a query is given by textual keywords or by a sketch, while subsequent interaction and query refinement is done by marking images as relevant or non–relevant and supplying the system with additional information, such as preferred features or regions of interest. The two different parts of the interaction, *starting point* and *interaction paradigm* are illustrated in Figure 4.10. Of course in both situations, only the features that are extracted for any given system can be used to start a query or for the further interaction. In general, the features used in CBIR are described in Section 2.3.2.

Many different classifications of interaction methods and query paradigms exist in the literature. In [302], a categorization of query methods is done into the following categories:

- query by spatial predicate,
- query by image predicate,
- query by group predicate.

This grouping is mainly based on what is regarded as relevant for the user in the desired result images. Another grouping is done into:

- exact queries,
- approximate queries.

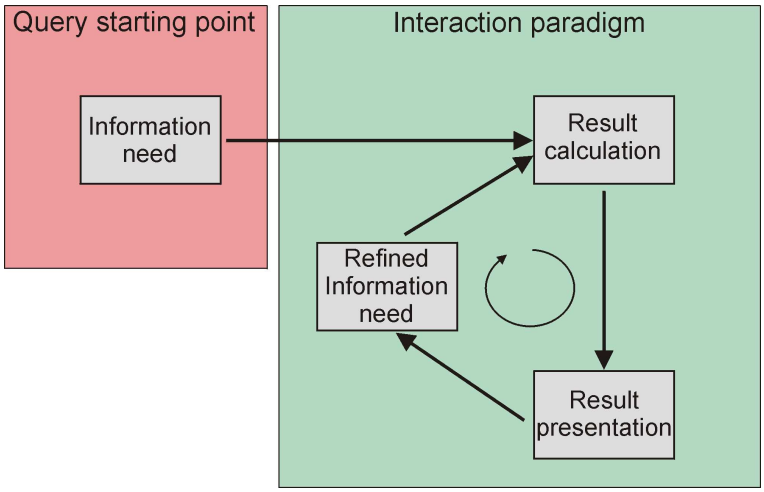


Figure 4.10: The difference between the query starting point and the actual interaction in the query process.

This mainly takes into account the goal of the query and how exact the target image(s) in the mind of the user are. For this section, we mainly take into account approximate queries that give a ranked list of results.

The problem of finding a good start image for similarity queries is also called *page zero problem* [241].

4.2.1 Text, annotation

For most image databases used in museums [240] and libraries [114], as well as for journalists [151], the users can start the queries by entering text [56, 241]. This can either be free text with approximative search or, when using databases, the text can often be entered in fields for exact matches.

Text allows us to input semantic information into the system as a starting point and thus might be a very natural way of querying as our search goals often are of semantic nature. Still, such a query scheme relies heavily on a good annotation of the images.

A very interesting system [306] starts the search with a keyword and then gathers information about images from text search engines on the WWW. The links are then followed to try to find the best images by performing a visual clustering of the found images. The search for roughly 100 result images takes some 5 minutes for the search being restricted to local web sides. Still, in our opinion it is a remarkable idea.

Many search engines on the web such as Google, Altavista or Lycos allow for image search. These search engines start with a keyword and then search for images related to this keyword by nearby text on a web page, annotation (*e.g.* the ALT tag in HTML) or by their filename.

4.2.2 Image example(s)

Most CBIRSs [22, 77, 191, 203, 264, 278] currently use image examples as a query starting point. In [281], QBE queries were divided into two groups, *query by external pictorial example* and *query by internal pictorial example* signifying whether the query image is already indexed in the database or not. Such a classification can be refined by the following two models:

- using *user* supplied example(s),
- using example(s) proposed by the *system*.

A *user supplied example* can either be image(s) from the database (internal) that the user has to find or not yet indexed images (external) that the user possesses. *System supplied example(s)* are always images from the database (internal).

System supplied examples are for examples used in systems for image browsing such as `PicHunter` [48, 49] and *TrackingViper* [190] that supply the user with images to maximize the information gain at each query step to find a certain target image. The filter image browsing described in [281] also proposes images to the user.

Many CBIRSs supply the user with a function to get a set of random images on the screen. This does not really qualify as a *system supplied example* because it is still up to the user to find image(s) as a proper starting point. In this case, proposed but unmarked images are normally being discarded for the query process. One exception is the system described in [82, 139], that supplies the user with images and uses the information of all the images that were presented to the user to calculate the next set of images to present to the user, even when the user did not mark a single image. The interface of this system has separate sides for the interaction with the user and for for the query result. The user needs to take care as non–marked images are all assumed to be non–relevant

4.2.3 Image segments or regions

Maybe the most well known system that supports query by region is "Blobworld"[2] [9, 24, 25] that is described in more detail in Section 2.4.5. Blobworld segments all the images in the database and then allows users to chose the presegmented regions, set weights for the importance of the regions and the features to use for a similarity search. Spatial relationships can also be used for searching several regions. Other systems that support region queries are [58, 112]. In [112] the importance of regions is learned from user feedback.

The imperfectness of fully automatic, strong segmentation may lead to suboptimal results and not always will the pre–segmented regions comply with what the user would like to have as a region query. Solutions can then be given by interactive segmentation or completely user defined regions to start a query.

4.2.4 Sketch

Already early system such as [71, 116] supported query by sketch as a starting point for similarity queries. Other systems that support sketches are presented in [290, 308].

The main difference between a sketch and an image to start a query is that the sketch contains incomplete data. The sketch normally concentrates on the main object in an image and mostly on the outline. The background is often discarded and also the colors are not always used and if they are used they might not be exact.

Sketches may also contain symbolic information only. This can make retrieval impossible as demonstrated in Figure 4.11 where everybody will recognize a cat. In low level features spaces, however, there will be no similarity between a cat on a real–world photo and the drawn cat.

Most often, a sketch can be used as a starting point, but then the query refinement has to be done with the returned example images as QBE. Refinement by sketch seems hard to realize, unless the user has a certain artistic drawing talent.

4.2.5 Other starting points

In [119], *query by gesture* (QBG) is explained. This is basically query by sketch, only using gesture as a mode to input the sketch data. Whatever the interaction means, what counts as a starting point is the input data.

Another possibility for a starting point is also the use of *spatial relationships* in systems such as Blobworld [24]. This can often be done in conjunction with image segments, leading to queries such as "A red round object left of a rectangular blue object". Query by spatial relationships of

[2] `http://elib.cs.berkeley.edu/photos/blobworld/`

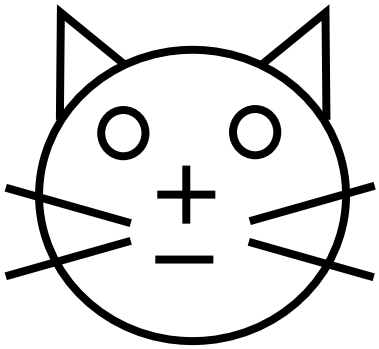


Figure 4.11: A sketched cat as a query starting point.

objects in images is described in [88, 248, 292]. Again, the use of spatial relationships seems to be well feasibly as a query starting point, but then the interaction with several images seems to be less attractive.

4.3 Interaction paradigms

This part only describes the actual interaction after the first response for image retrieval as the *query starting point* was already described in Section 4.2. In fact, it is the case that most of the classifications taken from the literature do not distinguish between the query start and the further interaction but take more into account the actual information need. Because of this, some of the information of the preceding section is also valid and retaken for this section.

A grouping given in [302] for interaction paradigms is into:

- target search,
- category search,
- associative search.

Here the target search corresponds to our paradigm of image browsing whereas the other two groups correspond to our QBE paradigm. Although category search and associative search clearly have different characteristics it is hard to say for a generic database whether the system can actually distinguish between the two. On the user side, the distinction is, of course, clear.

4.3.1 Query by textual annotation

In most domains, text has been the only query paradigm for a very long time [152, 240]. [212] contains many references to different articles on annotating images by free text or controlled vocabulary. The goal of content–based visual information retrieval is, however, to produce tools to handle image database without the need for annotations or at least to combine the visual and textual characteristics [241]. Textual characteristics are, at least so far, the only really semantic way to query image databases and the experience with expressing our wishes by text gives us easier access than the search by example images normally does. Only when we really search images similar to an example we have might query by image example be more intuitive.

Most search engines on the web such as Google, Altavista or Lycos use the text next to the images not only for the query start by also for the rest of the interaction. These search engines used to support also visual similarity search for later query steps ("find images similar to this one"), but at the moment they seem to solely use text. In [241, 294], visual and textual characteristics are combined for the search.

4.3.2 Query by example(s)

User interaction by image example(s) is by far the most commonly used interaction paradigm in image retrieval supported by most programs [71, 77, 198, 203, 264]. Several methods for relevance feedback and various relevance feedback strategies using example images are described in Section 4.5. This paradigm can be explained with asking the system to "show me some images like this one or these ones". After a starting point is found, this techniques is easy to handle and the user simply marks all the images as relevant or non–relevant, or he gives a weighting to the images as very relevant or a bit relevant as proposed in [222].

Some programs such as Blobworld [9] present interaction on parts of the image, for relevance feedback. Although this is very useful for the query start, in the interaction it might take very long to mark image regions in several images and to express their spatial relationships.

4.3.3 Image browsing or target search

The target search or browsing paradigm was introduced to image retrieval with the `PicHunter` system [48, 49]. The goal of such a browsing system is to find one certain image that the user has in mind in a large image database. When we know that an exact image existing in the database is being searched for, we can present to the user images that maximize the information gain in each feedback step, which is the principle of `PicHunter`.

When the user changes the goal of the search (a change of the image that he has in mind), a new query session has to be started. The system *TrackingViper*, described in [190], allows the user to change its mind during the query process. This is reached by deleting inconsistent judgments from the interaction process, which allows the user to move the goal of a query without restarting a new session.

A system for filter image browsing is described in [281]. This system is offering to the user a hierarchical overview of the database to be able to browse quickly through a large set of images. The user can successively choose the image he regards to be close to his target and then a new image selection is shown, taking into account the information gained over the entire query session. The limiting factor is that only one image per step can be chosen. So, when the user's goal is in between two images (s)he risks to start exploring an incorrect part of the database.

All these browsing techniques can also be extremely useful to find a query image to start a similarity search as is shown in [190] with the mixed *Viper / TrackingViper* version.

4.3.4 Other paradigms

Mixtures of image browsing and QBE above have shown much success, as reported in [190]. *TrackingViper* combined with *Viper* support both paradigms, starting with browsing to find a good starting point. Then, the QBE–based system *Viper* is used to execute a similarity query once a good starting point is found.

The system described in [32, 82] also uses all the images shown on screen even if nothing is marked as relevant by the user. This information is used to train support vector machines (SVM) and to present the user maximally discriminative images for the further training of the support vector machines. This is very similar to image browsing and it offers the property of browsing that the user does not need a good query image to start the query process.

Success has also been reported from the mixture of search by annotation and visual features [241]. [294] uses images from a newspaper as a visual information and the image caption as annotation. This can lead to significantly better results. With very good annotations, visual features can only marginally enhance the performance as can be read in [38]. The problem for measuring the performance is of course whether purely semantic groundtruthing had been done or whether visual features were taken into account for obtaining the relevance judgments.

4.4 Presentation of the search results

For now, not much formal research has been done in the area of user interfaces for CBIR. Some of the developments are very promising, though. In [92], an intelligent user interface is proposed that uses techniques such as speech recognition for querying multimedia data. In [191], the results are presented in a virtual reality manner, in a three dimensional space. This means that the axis can be used to separate images retrieved due to color similarities and images retrieved because of texture similarities. Often the queries are a mix of the two and the desired axis of a user may symbolize other criteria. It can for sure be said that retrieval interfaces have a large potential for improvement.

In [269], the performance of different user interfaces for textual information retrieval are compared and similar studies can be imagined for CBIR as well.

4.4.1 User interfaces for *Viper*

Within the *Viper* project, several user interfaces have been developed for diverse purposes. Such a user interface development is relatively easy with *Viper* as the *MRML* interface is open and described in detail. The currently available user interfaces for *Viper* are described in this section.

Java

SnakeCharmer (see Figure 4.12) is an *MRML*–compliant client application[3]. It is written in JAVA for portability and offers query by multiple positive and negative examples, query history, multiple collection and algorithm selection. A scatter plot of the results according to various aspects of similarity and a basket for user–selected images are offered as well. This interface was developed at the EPF Lausanne for the project described in [205].

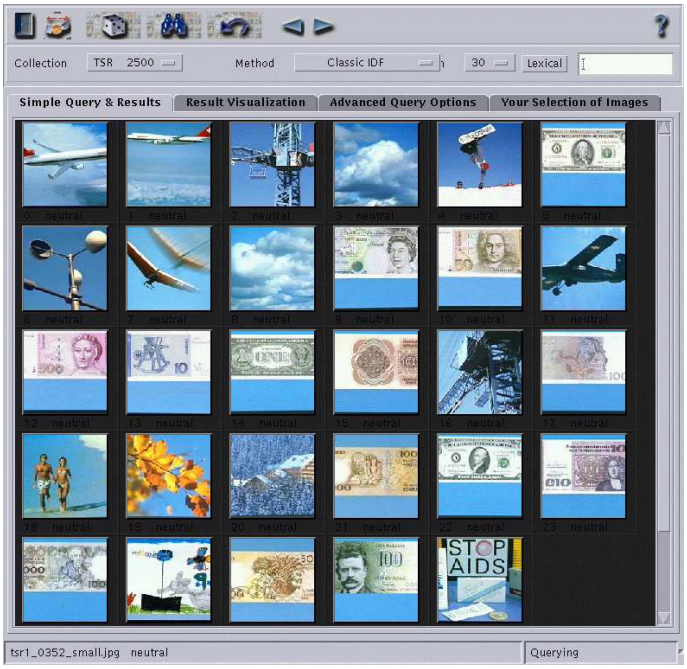


Figure 4.12: The JAVA interface SnakeCharmer.

This user interface also offers options that are not yet supported by *MRML*. It offers, for example, the possibility to group the retrieved images on two axes based on the similarity with

[3] `http://viper.unige.ch/demo/demo.html`

respect to feature groups. These two axis can for example be for similarity based on texture and based on color, respectively.

CGI

The CGI interface in Figure 4.13 allows the query based on visual or textual features[4] or on a mixture of the two groups. The weighting between visual and textual features can be chosen via the interface.

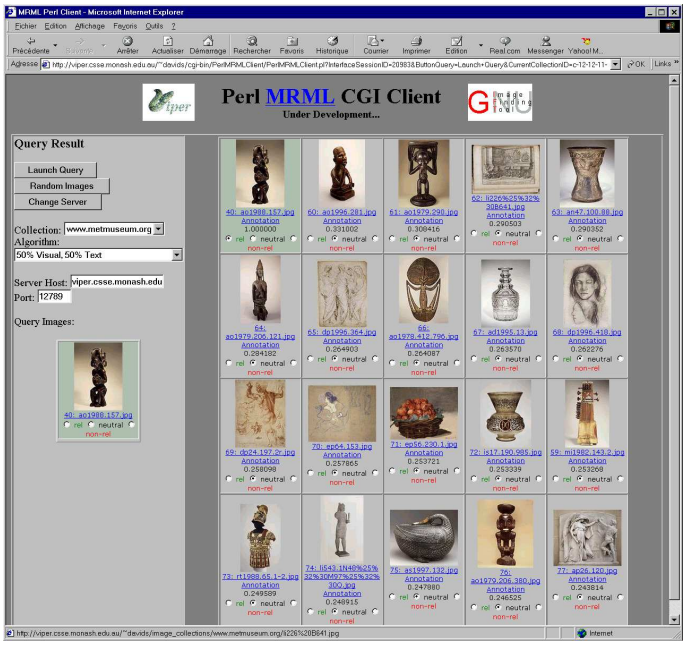


Figure 4.13: The CGI interface that supports querying based on visual and textual characteristics.

Positive and negative feedback can be used. At the moment, only two collections are available: a personal photo collection and the collection of the Metropolitan Museum of Art[5]. Like the Java interface, it uses web technologies and can thus be accessed from any place via the WWW.

PHP

As we had several problems with our Java user interface with respect to firewalls and also with respect to speed for the download, we wanted to have a lightweight and fast interface. A demonstration version is available on the web at[6]. As the code for the interface is rather short, it is very easy to adapt the interface to different needs.

As additional features, this interface includes the possibility to submit images from the local hard disk to the system or from any given URL.

Others

With the *GIFT* package, another *MRML* compliant client is delivered, the command line tool `gift-mrml-client.pl`. This tools allows to execute whatever query and stores the result in a file in HTML format. This allows easy viewing of the retrieval results via a web browser. The script is the basis for applications using the *MRML* interface such as the automated benchmark described in Section 5.3.

[4] `http://viper.csse.monash.edu/~davids/cgi-bin/PerlMRMLClient/PerlMRMLClient.pl`
[5] `http://www.metmuseum.org/`
[6] `http://viper.unige.ch/demo/php/demo.php`

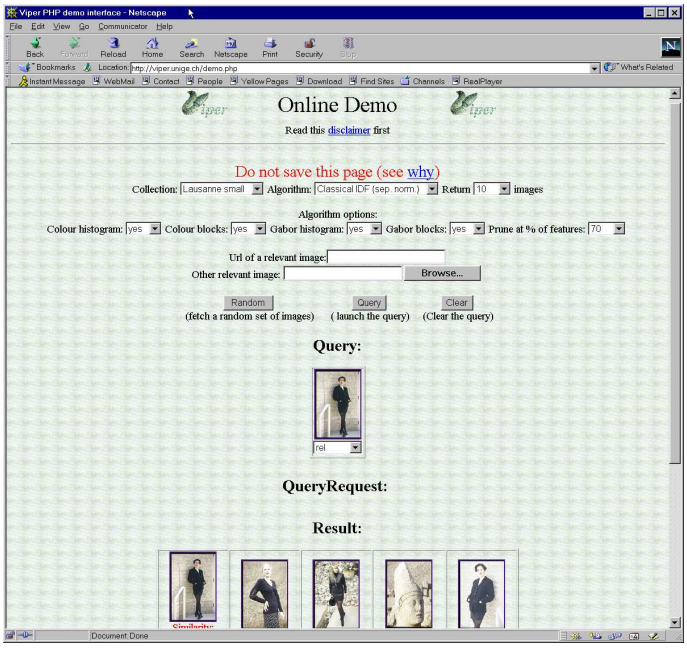


Figure 4.14: The PHP interface using *MRML*.

Another project is *kmrml*[7]. This is a plugin for browsers like the Konqueror of the KDE development to search for images directly from a browser or on the desktop. Figure 4.15 shows a screenshot of the kmrml interface.

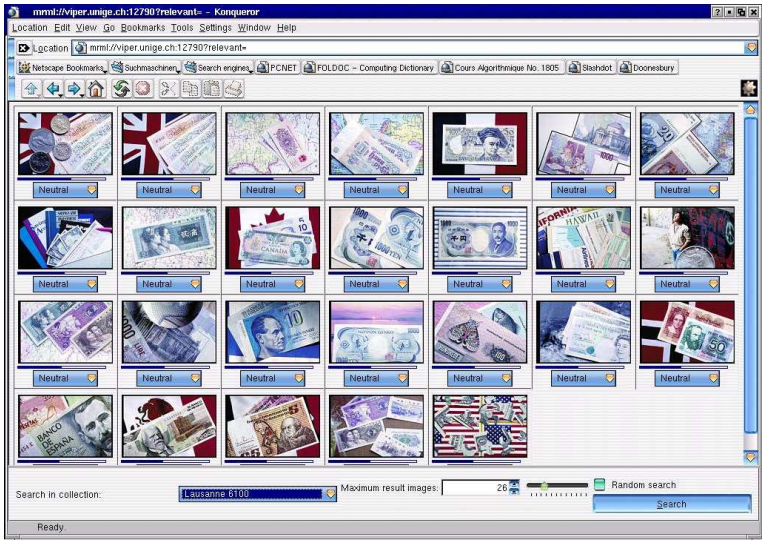


Figure 4.15: The kmrml interface for searching images.

4.5 Relevance feedback

Relevance feedback, a well known technique in text retrieval [218, 229], is regarded as one of the most powerful techniques to improve the results of content–based image retrieval systems. Much

[7] `http://devel-home.kde.org/~pfeiffer/kmrml/`

has been written about different ways to implement relevance feedback [177, 221, 222, 229, 247, 264, 302]. Most systems calculate the relevance feedback from one query step to the following query step. Like this, not the entire query session is used for calculating the next results but only the preceding step. In *FourEyes*, across–session learning is proposed [161] and also in [277, 301], it is proposed to learn over several temporal scales. [301] uses neural networks that are trained within the sessions and also in between query sessions.

Some systems like the image browsers `PicHunter` [49], *TrackingViper* [190] take several steps of user interaction into account to find a target image in a database. [281] uses a form of hierarchical browsing, the so–called filter image browsing to move within the database. In [139], an approach to learn query concepts over several interaction steps without using seed images is proposed, using support vector machines. [10] proposes to use relevance feedback also for meta–search engines.

4.5.1 Strategies for positive and negative relevance feedback

There are two main strategies to execute queries with several images as relevance feedback on a system level:

- Separate queries are executed for each feedback image and the query results are merged in the end.
- A "pseudo" query image is created by merging the features of all query images, and then a query is executed with this pseudo–image.

Both strategies have advantages and disadvantages. Executing separate queries slows down the query process, and the strategies for merging the results are not obvious. It is important to stress the importance of features that occur with similar values in all the query images. Thus the features that the query images have in common need to be found out as well. The use of a pseudo–image has an advantage with respect to the query speed, but when simply averaging over continuous feature values, much information can get lost. The use of a sparsely populated space facilitates the task of generating a pseudo–image because the features are not continuous but already quantized. Thus, averaging does not mix different measurements to an average value. Only already very similar measurements are combined for the pseudo–image.

It is also not clear whether images marked as relevant by a user in the same query step should be combined like a logical "AND" or a logical "OR" or even something in between.

Whereas the strategies on a system level might not be too complicated, the strategies for a user searching a certain set of images are much more complex. To mark positive feedback is straightforward, but to find the best negative feedback images is not an easy task. Especially unexperienced users often abstain from the use of negative feedback which can have a very negative influence on the retrieval results. In [177], several strategies for system users to mark positive and/or negative feedback are compared. This study pinpoints the importance of the use of negative feedback.

This sections describes several strategies for generating positive and/or negative feedback. The performance results of the different strategies are then compared. The results in this section have been obtained with using the TSR 2500 image collection described in Section 5.2.3 and the performance evaluation techniques described in Section 5.

Automatically generated feedback

Automated relevance feedback can be generated once relevance judgments for an image collection exist. Thus, a reproducible relevance feedback for every user can be simulated based upon the judgments and the initial query results of a system. Via this technique, the flexibility of a system with respect to the users' needs can be measured, *e.g.* by feeding back the images the user judged as relevant and which were returned in the top $N = 20$ of a query result. This technique can be used either to compare different feedback strategies or to enhance user queries by automatically creating additional negative feedback when the user only supplies positive feedback. The performance of these different strategies will be compared in the following sections.

Such an automatically generated relevance feedback is a prerequisite for the automatic benchmarks described in Section 5.3 and 5.4.

Positive feedback alone

Positive feedback is limited to preselected images that appeared in the initial query results and weights the features that these images have in common more strongly. As all highly ranked images already have many features in common, the non–relevant (non–marked) images may also be ranked highly in the next step. For this feedback, we select as relevant all the images from the initial query result which the user judged to be relevant. We chose images for feedback from the first 20 highest ranked response images, which is a reasonable number to display on screen simultaneously. 50 is regarded as the maximum number of images a user might normally browse, and 100 is simply used to show the improvements.

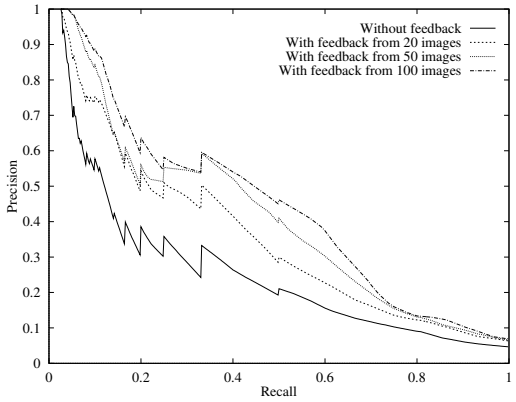


Figure 4.16: *PR graph* for only positive feedback.

The improvement in performance using relevance feedback is quite large as can be seen in Figure 4.16. When using only feedback from the first 20 result images, the *PR graph* is improved by 20% in some areas. Using 50 images for relevance feedback gives an additional improvement of about 10% in most regions of the *PR graph*. The use of 100 images improves only some parts of the graph by an additional 5%. Much of the improvement comes only from relevant images being ranked higher in the top N and not from returning new relevant images.

Positive and negative feedback

Negative feedback can improve the query result greatly, but it is important to use the right images as negative feedback so as not to inhibit any important features. Many systems have problems with too much negative feedback. This can result in positive features getting a negative weighting, which leads to images with few features ranked highly (images with very few colors and textures).

Based on these facts, we apply a variety of methods for automatic selection of negative relevance feedback. Positive images from the top 20 returned were still all selected as positive feedback. As negative feedback, we chose the first two and the last two non–relevant answer images within the top 20. Since we identified that these different images influence different parts of the *PR graph* we also combine the two strategies (see Figure 4.17).

We can see in Figure 4.17 that returning the first two images as negative feedback improves the beginning of the *PR graph* by 4 to 5%; using the last two improves the middle of the *PR graph* by up to 7%. The combination of both improves all parts of the graph by up to 9%. This shows that different negative feedback images improve different parts of the graph significantly by removing different areas of feature space from the query.

With this knowledge, a query from a user who only uses positive feedback can be improved by automatically supplying a few non–selected images as negative feedback. Our experiments have

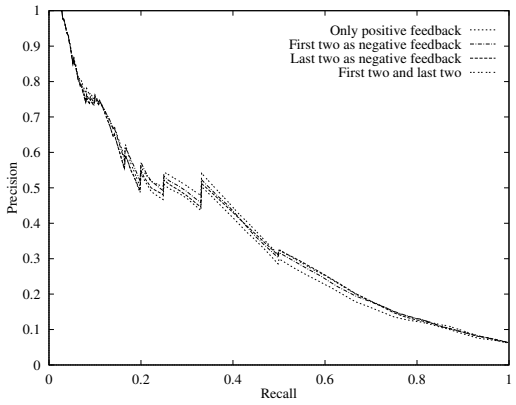


Figure 4.17: *PR graph* for several negative feedback image choices.

shown that non–experienced users tend to use solely positive feedback which leads to suboptimal results.

Different feedback weightings

As we know, different negative feedback images can improve different parts of the *PR graph* but also decrease performance when used in excess. We therefore reduce the latter effect by weighting the images with a factor other than -1, and we can feed back all neutral images as negative relevance feedback.

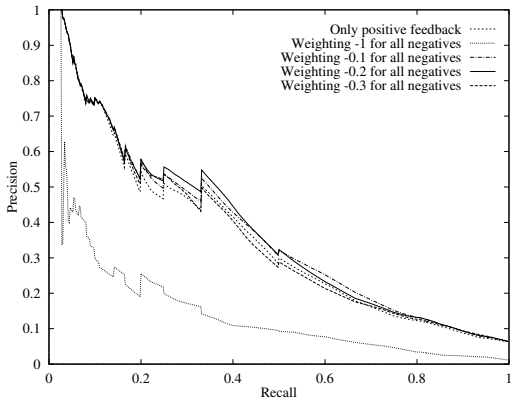


Figure 4.18: *PR graph* for various amounts of negative feedback.

In Figure 4.18, we can see that the value of -0.2 for negative feedback yields the best curve in most areas, only in the end the curve with -0.3 is better but these last parts of a *PR graph* are not as important since they only give information about images which are ranked lower and are often not shown to the user. The -0.3 curve is sometimes even worse than the curve with only positive feedback. The value of -0.2 creates improvements of up to 7% or 8%. Using higher weightings does not bring any further improvements.

A good idea might be to create negative feedback automatically with a low weighting when the user does not use any or enough negative feedback. This allows to feed back all images with a lower negative weight without having the risk of deteriorating the retrieval result by too much negative feedback. Also, the negative feedback added by the system should have a lower weighting than the negative feedback supplied actively by the user.

Separately weighted feedback

Problems due to too much negative feedback in text retrieval were already addressed by Rocchio in the 1960s [218]. Following this work, our system weights the features of positive and negative query images separately according to Equation 4.1,

$$C = \frac{\alpha}{n_1} \sum_{i=1}^{n_1} D_i - \frac{\beta}{n_2} \sum_{i=1}^{n_2} E_i, \tag{4.1}$$

where C is the set of weighted features making up the query, n_1 and n_2 are the numbers of positive and negative images in the query, respectively. D_i and E_i are the (possibly weighted) features in the positive and negative images, and α and β determine the relative weightings of the positive and negative components of the query (normally: $\alpha + \beta = 1$). We use $\alpha = 0.65$ and $\beta = 0.35$. The adaptation of this formula for the use in *Viper* can be seen in Equation 3.15.

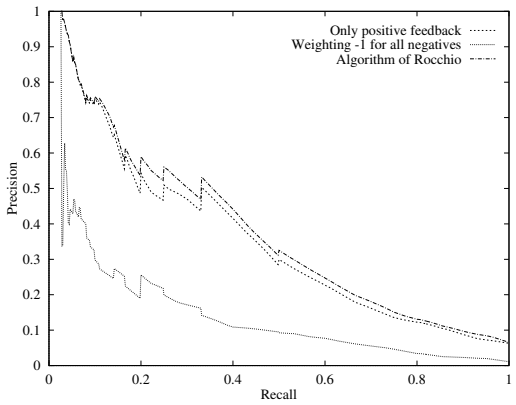


Figure 4.19: *PR graph* for negative relevance feedback with a modified Rocchio algorithm.

This technique significantly improves the query results (up to 9%). This is better than the other methods for positive and negative feedback. Clearly, we still need to test whether the weightings of 0.65 and 0.35 are as good for CBIR as they proved to be for text retrieval, but we already made the result more or less independent from the number of positive and negative feedback images. Using this method with a larger number of result images (*e.g.* 50 as in Section 4.5.1) improves the results even more.

Figure 4.19 also shows well how bad the performance with feeding back all negative images is without using the Rocchio weighting.

Several steps of feedback

To measure the interactive performance of a system, we need to consider more than one step of relevance feedback since browsing is a crucial task for CBIR [151]. We thus made experiments with several steps of relevance feedback. Only the use of several steps of feedback can show the system's ability to adapt to the users' information needs.

Figure 4.20 shows the results using two steps of only positive feedback. The major improvement occurs in the first feedback step (20%). For the second step, it is rather small (2% to 3%). The improvement with positive and negative feedback is remarkable for the first four steps where the results consistently become better. The first step already shows an improvement of about 25% and the second step an additional 10%. In the third step the results improve by about 10% in the beginning and by 8% in the middle parts. The gain for the fourth step is 5% in the middle and as well in the end. This improvement in the end means that images which were far away from the initial query have been moved closer. This improved complete recall value is very important for fields like trademark search where an extremely high recall is essential

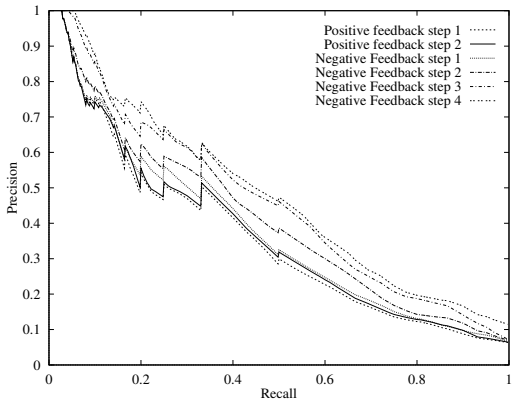


Figure 4.20: *PR graph* for several feedback steps with negative feedback.

Comparison of the results

All these results show the great importance of negative relevance feedback for the image searching process. The effect of positive feedback almost disappears after only one or two steps so the possibility to move in feature space is limited. Negative feedback offers many more options to move in feature space and to find new result images. Even hard queries can continuously be improved at every feedback step. This flexibility to navigate in feature space is perhaps the most important aspect of a CBIRS.

4.6 Learning over several temporal scales

Learning from user interaction over several temporal scales has been proposed as early as in the *FourEyes* system, where the user could mark image regions with annotations. Another proposition is described in [277], where a probabilistic architecture for across–session and within–session learning is used, based on Bayes formula.

A very early version of creating a user profile is done in [116] where a user is asked to sort a set of images into groups. Then, the weightings for the different features are set based on the sorting results.

In [136], a system is presented that groups images into clusters and changes these clusters whenever they are marked with contradicting relevances by another user. For this tool to be effective, all images of the database should have been selected by several users and should have been marked by at least one of these users. In [123], old user judgments are used to propose new images to users based on the items they have already marked. This *collaborative filtering* is applied to art images in a museum. A web demonstration is accessible at [8]. Amazon[9] also employs this technique for proposing new books and other items to users based on what they have already bought.

In [140], a method to store correlations between images is proposed that promises good results when all images in the system are marked. Very large databases will require large storage capacities for the correlations. The data to train the system is gained from automatically created usage log files.

The use of log files to discover knowledge is also very common in many other research areas in connection with the Internet. Log files are used to adapt web pages or web–accessible systems to the users needs [304]. In [11], the behavior of users within web pages is analyzed to improve the page layout.

Experimental results in [175] have shown strong improvements in CBIRS performance when using feature weights that are calculated with the help of usage log files. This study uses images

[8] `http://abyss.eurecom.fr:1111/AWM/login.html`
[9] `http://www.amazon.com/`

marked together in the same relevance feedback step for the calculation of a new feature weight. Therefore, it seems logical to take a more formal approach to exploit the usage log files of a CBIRS.

The stated problem has many similarities to the *market basket analysis* (MBA) often described in the data mining literature. Supermarkets have large files of items purchased together by a customer at the same shopping trip. One would like to know which combinations of these items occur significantly often and how association rules can be derived from these data efficiently. In [2, 3], efficient algorithms are described to solve the task. An explorative evaluation of all combinations is infeasible as there are thousands of items and several hundreds of them can be purchased together at the same shopping trip. Thus, we need algorithms to efficiently filter the data and follow promising groupings of items. Association rules can be derived from these data.

Our proposition for learning feature weights over several temporal scales can be found in [171, 175, 182] and a shorter German version of it [176].

4.6.1 Analysis of user logfiles

The evaluation is based on log files with user interaction obtained from a web demonstration version of *Viper*. The log files are in *MRML* format. This section first describes how we reduce the large amount of data to find images being marked together several times. The reduction process is analog to the reduction of data described as the *market basket analysis* in the data mining literature [2].

Comparison with the market basket analysis

As mentioned earlier, the *market basket analysis* aims at characterizing articles (in our case: images) that are often bought (in our case: marked by users) together. Although there are many similarities between finding images marked together in queries and items bought together in a supermarket, there are some differences we have to consider. The main differences are:

1. Each image can only be marked once in a query whereas in a supermarket an item can be bought several times.

2. An image can be marked in three different ways: relevant, non–relevant or neutral, whereas an item in a supermarket is either bought or not.

3. The sizes of the basket in a supermarket can vary in a much larger range compared to image retrieval, where users normally do not mark a very large number of images.

4. We have less queries with the system than we have items, whereas in a supermarket much larger datasets of items bought together are available than there are items in the shop. Thus, almost every item in a supermarket is bought every once in a while, whereas not every image in our databases is marked as regularly.

5. Items that most frequently imply other items are most important for a supermarket, whereas for us it is more important to find the images that are marked relevant after several steps of feedback. This is the case because items (images) very often marked together are frequently retrieved together anyway, whereas we also want to retrieve items that are not always retrieved in the first query step, which means that they are further away in feature space.

The additional information described in 1.) is often not used for the qualitative analysis described in [2] anyway. Therefore, we can ignore the unavailability of this information. The solution we use for coping with 2.) is that in the first step we regard an image marked "relevant" and the same image marked "non–relevant" or "neutral" as different items. In our practical setup, we did not evaluate the images marked "neutral". These images were not actively marked by the user, so the information content can be interpreted in different ways. The user might have been too lazy although they were relevant for him, or the user might have thought that they are not relevant for him but feared that marking them negatively could worsen the results, or the user may simply not have looked at them. Clearly, images marked neutral are in between the relevant and non–relevant

	relevant	neutral	non–relevant
relevant	++	?	++
neutral	?	–	–
non–relevant	++	–	–

Table 4.6: Possible combination of markings of image pairs. The ones we are using for our approach are marked with "++".

images. 3.) is not as important as we can theoretically have larger groups of images as well, and most shopping data sets will not be much larger than our image data sets. Remark 4.) is important because we cannot prune out as much information as can be done from very large log files. We have to rely more on the quality of the data, because otherwise we can not get much information from the log files. With time, we will get larger log files as well, which can be evaluated in a better way. 5.) also limits the amount of information we can filter out for further processing. Some images often marked together with different images in a different context might still contain very valuable information. On the association rule level, we do not set any minimum requirement for the probability of an association rules.

Reduction of the log files

We have several steps to reduce the data of the log files for the final analysis.

1. Reduce the log files to image sets marked together because the rest of the communication descriptors in *MRML* is not being needed.

2. Find images and then image pairs that occur more than once, so–called *large images* or *large image sets*, respectively, as they are named in [2].

3. Use only image pairs that we are really interested in as explained in Table 4.6 and described below.

4. Calculate Probabilities of association rules for all *large image sets* we found in 2.), and only use those above a certain probability threshold.

At this stage, we concentrate on association rules between pairs of images, that is sets of size 2 only. Association rules with more than two images are not evaluated for the moment.

For us, only images marked together positively in the same query step or image pairs where one image is marked positively and the other one negatively are of interest, because images can be marked negatively or left neutral in the query for several different reasons. Table 4.6 shows the combinations of image pairs we are interested in. Only the combinations marked with "++" are used is this paper. We could also use the ones marked with "?", but we decided not to as they were not actively marked by the user and so their information content is questionable. The combinations marked with "–" cannot be used. Each image in a pair of negative examples can be marked negatively for its own particular reason.

The analysis of the log files is performed completely automatically with perl scripts. The log files are in *MRML* format which is based on XML, so a generic XML parser is used to extract information about images being marked together in one query step. We then calculate the frequencies of images with a specific marking (1=relevant, −1=non–relevant). Our limit for images being called *large images* is that they have to occur at least twice. With larger log files, this limit of 2 may be increased but so far our log files are rather small and we do not want to let any information being unused. Images marked only once with a certain relevance are then removed from the query groups of images marked together in a step. Now, each query which still contains multiple images is read in and all the possible combinations of two images are created and stored with their frequency. Pairs that occur less than twice are deleted from this list.

There are further important differences between the market basket analysis problem and ours. We are not necessarily interested in the association rules that occur the most frequently. If images

are marked together very frequently, it means that they already have a high similarity and show up together frequently. We are much more interested in relevant images that show up after one or more steps of feedback. These image pairs might not have a very high frequency for the association rule but they are indeed more important than images in rules with a higher frequency, that appear each time on screen together. For this reason we do not set a limit for the probability for association rules but also evaluate rules with small probabilities.

4.6.2 New weighting for features learned from interaction log files

The data reduction explained in the above section leaves us with a list of images marked together more than once, as well as with the frequencies with which they are marked together and alone. This is used to create association rules for the connection of images. These association rules can be relevant/relevant or relevant/non–relevant combinations.

Association rules

From each of the pairs consisting of images I_x and I_y we can construct two association rules: $I_x \rightarrow I_y$ and $I_x \rightarrow I_y$. The probabilities p for these association rules can easily be calculated from the frequencies of the two images marked together and marked alone as in Equation 4.2.

$$p(I_x \rightarrow I_y) = \frac{f(I_x I_y)}{f(I_x)}, \tag{4.2}$$

where f is the frequency of the image or image pair, respectively. Each image pair thus produces two association rules and their probabilities. "*Positive association rules*" are rules where the two images in the pair were marked as *relevant* and "*negative association rules*" are rules where one of the two images is marked *non–relevant* and the other one *relevant*. *Positive* and *negative association rules* are noted $(I_x \rightarrow I_y)_+$, $(I_x \rightarrow I_y)_-$,respectively.

From images to features

With the association rules, we have a connection between images, but we want to learn on a feature basis to be more general and also learn for images not yet marked. Thus we have to make a connection between the association rules for images and the features that these images contain.

First, we have to think about what makes a feature a good or positive feature in our sense, see Equation 4.3:

$$p(F_j\, good) \quad = \quad p(F_j \textit{predicts relevance}) \wedge p(F_j \textit{does not predict irrelevance});$$

where F_j is a a feature and p the probability. This means that a feature is a good feature if it is good at predicting relevance and bad for predicting irrelevance. Prediction of relevance corresponds to positive association rules and prediction of irrelevance to negative association rules. In short, this means that the feature is often in images marked with opposed relevances.

Now, Equation 4.3 calculates the probability p that a feature predicts relevance by using the frequencies of features being in both images of an association rule or only in the first one.

$$\begin{aligned}
p(F_j \textit{predicts relevance}) &= p((F_j \in I_x) \Rightarrow (F_j \in I_y) | I_y \textit{is relevant for} I_x) \\
&= p((F_j \in I_x) \Rightarrow (F_j \in I_y) | (I_x \rightarrow I_y)_+) \\
&= p(((I_x \rightarrow I_y)_+ \wedge F_j \in I_x) \Rightarrow (F_j \in I_y)) \\
&= p(((F_j \in I_y) | (I_x \rightarrow I_y)_+ \wedge F_j \in I_x)) \\
&= \frac{f(((I_x \rightarrow I_y)_+ \wedge F_j \in I_x) \Rightarrow (F_j \in I_y))}{f((I_x \rightarrow I_y)_+ \wedge F_j \in I_x)} \\
&= \frac{|(F_j \in I_x \wedge F_j \in I_y); (I_x \rightarrow I_y)_+|}{|(F_j \in I_x); (I_x \rightarrow I_y)_+|},
\end{aligned} \tag{4.3}$$

where f is the frequency and $|C|$ is the cardinality of set C.

The same can be done for *negative association rules*. There are two possible ways for negative connections: $I_x^- \rightarrow I_y^+$ and $I_x^+ \rightarrow I_y^-$. These two cases are treated the same way. The probability for a feature predicting irrelevance is shown in Equation 4.4.

$$p(F_j predicts\ irrelevance) = \frac{|(F_j \in I_x \wedge F_j \in I_y); (I_x \rightarrow I_y)_-|}{|(F_j \in I_x); (I_x \rightarrow I_y)_-|} \tag{4.4}$$

Combining the two values we get Equation 4.5, which is the first factor a_{1j} we calculate for each feature F_j out of the feature space $\mathcal{F}$.

$$a_{1j} = p(F_j good) \quad = \quad p(F_j predicts\ relevance) \cdot p(F_j does\ not\ predict\ irrelevance).$$

When calculating this factor it becomes unfortunately clear that our low level features are frequently in one of the images of an association rule, but only rarely in the two, thus creating many extremely low values and being not very specific about the feature quality. Another problem is how to obtain a probability for features not present in any of the image pairs which should be a value between the good and the bad features. As these features seem to be in almost no image, we can discard these features and let their probability become 0.

To solve the above problem, we limit our calculations to the times a feature occurs in both images of either a positive or a negative association rule. This leads to Equation 4.5, which turns out to be much more discriminative between "good" and "bad" features.

$$a_{2j} = p(F_j good) = \frac{|(F_j \in I_x \wedge F_j \in I_y); (I_x \rightarrow I_y)_+|}{|(F_j \in I_x \wedge F_j \in I_y); (I_x \rightarrow I_y)_- \vee (I_x \rightarrow I_y)_+|} \tag{4.5}$$

It will be shown in the next section that the second factor a_{2j} also leads to better results. If a feature does not occur in any of the association rules we give it a value of 0.5 as we do not have any information to give it another probability. This is in the middle of the best possible value 1 and the worst possible value 0. Very frequent features will get a weighting close to 0.5 when they appear in almost every *positive* and *negative association rule*. In other words, 0.5 can be seen as the average weighting and corresponds to the "zero–knowledge" case.

Combining our probability factor with the *idf* weighting

To evaluate the performance of our system, enhanced with the probability of the feature that we just calculated, we also want to include the probability into our standard *idf* weight described in Equation 3.17. Therefore, we include the probability into the old feature weight calculation as an additional factor a representing the probability that the feature is a good feature, and we include this factor in the weighting as shown in Equation 4.6 to obtain a feature importance h_{qj} for a query q and a feature j.

$$h_{qj} = \frac{1}{N} \sum_{i=1}^{N} \left(a_j \cdot tf_{ij} \cdot Rel_i\right) \cdot log^2 \left(\frac{1}{cf_i}\right), \tag{4.6}$$

To calculate the score for an image we again use equation 3.18. To compare the retrieval results for a number of different factors we use both factors a_{1j} and a_{2j}, the squared (a_{1j}^2, a_{2j}^2) to check out a stronger influence of the factors, as well as a_{2j}^3 to check an even stronger influence of this factor. Because the results in [175] suggested that it might be better to keep even the negative factors with a low value and not let them become zero, we calculated a_{2j}^2 as well in a version where the lowest permitted value is 0.05, so these features are not completely discarded for negative feedback.

4.6.3 Experimental performance of the learned factor

For our experimental results, we analyzed 800 MB of user interaction data in *MRML* format from the *Viper* CBIR system. In the first reduction step we filtered the data to get all the images that were present in at least one query step. This led us to 3,258 queries with more than 80,000

images being marked in total. These images are from ten different databases, but most queries were done with three of the databases.

We decided to use the database of the University of Washington[10] for this evaluation because the database is freely available and thus the results are reproducible. This database contains 922 images in the version that we used, being separated in 13 image groups containing very different numbers of images (between 27 and 256 images per group).

In the second step, we regarded only image pairs which occurred more than once together. Over all databases, we have 61,361 such pairs with 21,727 of these pairs being from the Washington database. 7,930 of these pairs for the Washington database are positive connections (leading to *positive association rules*), 8,131 are a connection between a positive and a negative image (leading to *negative association rules*) and 5,666 are a connection between a positive and a neutrally marked image, which is not used for this evaluation. We finally ended up with 32,122 association rules for image pairs to calculate the factors.

Evaluation on the database of the University of Washington

From the 32,122 association rules, we calculated the different factors a_{1j} and a_{2j} derived in the preceding section. The retrieval results for the factors are compared for a first query step and for two feedback steps with automatically generated feedback from the query results as described in [177].

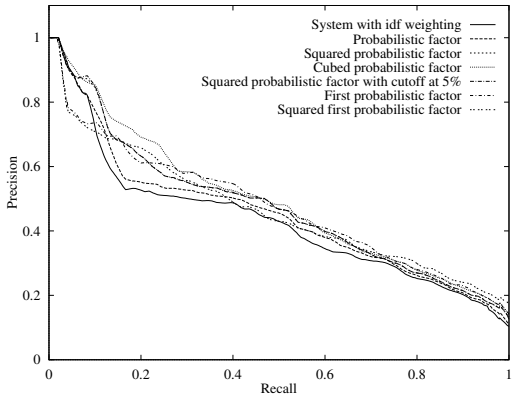


Figure 4.21: *PR graph* of different learned factors without using feedback.

Figure 4.21 shows a *Precision/Recall graph* (*PR graph*) for the first query step. We can clearly see that our first factor leads to a really bad performance in the beginning of the *PR graph* whereas the other factors all lead to similar, improved results. In the middle part of the *PR graph* especially the squared and cubed second factor give good results of up to 20% better than the original weighting. As the results for a_1^2 and a_1^2 where the features remain with a frequency of at least 0.05 are almost the same in the *PR graph*, the results omitted from the table.

In Table 4.7, we can see the other performance measures recommended in [178] for CBIR evaluation. In terms of rank measures, the first factor gives very poor results for the average rank, and the first relevant image is also found rather late compared to the second factor. Regarding the ranks, the normal *idf* system is the best, but only marginally better than the second factor. For a real user, the *precision* after 20 or 50 images, depending on how many images (s)he looks at on screen, is normally the most important measure. With respect to these measures, the systems with the additional factor look much better than the normal *idf* system. The new results are up to 7% better, which is a significant improvement. With respect to the *precisions* ($P(20)$, $P(50)$), the second factor (especially when cubed), gives the best results.

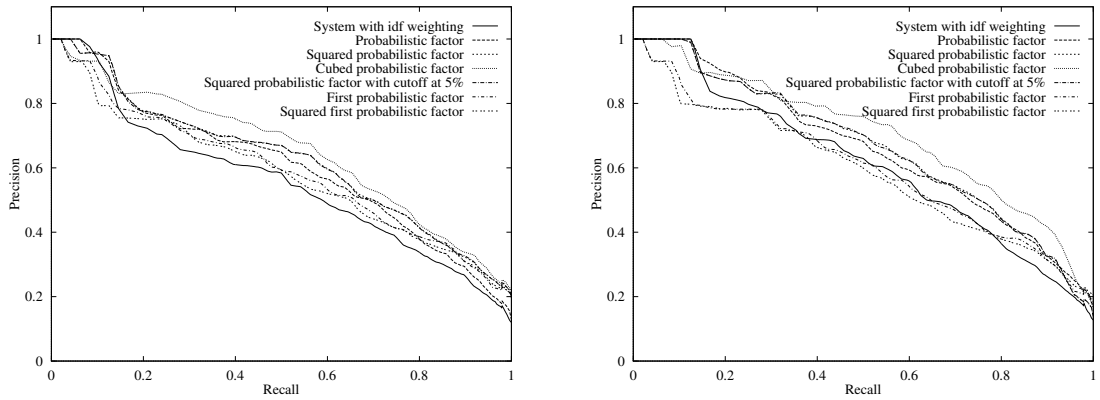
Even more interesting are the results after the first step of relevance feedback. Because the system uses old relevance feedback to improve the results for further queries, we could assume that,

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

	idf	a_2	a_2^2	a_2^3	a_1	a_1^2
r	65.14	65.14	65.14	65.14	65.14	65.14
$Rank_1$	1.5	1.5	1.79	4.79	7.93	11.43
$\underline{R(P(.5))}$	0.3799	0.4066	0.3653	0.3945	0.3791	0.2899
$\underline{\overline{Rank}}$	176.4	177.6	179.8	183.9	204.8	223.3
$\underline{\widetilde{Rank}}$	0.1583	0.1597	0.1620	0.1664	0.1895	0.2098
$P(20)$	0.5393	0.5786	0.6071	0.6	0.6071	0.5857
$P(50)$	0.4057	0.4157	0.4314	0.4371	0.4486	0.4271
$P(r)$	0.3883	0.4067	0.4214	0.4337	0.4349	0.4151
$R(100)$	0.4839	0.4936	0.5103	0.5135	0.5002	0.4782

Table 4.7: Performance measurements for different factors in the first query step.

in this case, part of the relevance feedback is already taken into account and the improvements could be less strong. Figure 4.22(a) shows that with relevance feedback, the results improve even more. Especially with the second factor (when cubed), we gain up to 15% in the middle of the graph. We can also see that the results get worse in the beginning of the graph, the higher a power of a_2 we use, but that the curve gets much better in the middle and in the end of the graph for higher powers. Similarly, the first factor a_1 is much worse in the beginning, but gets better in the middle and the end of the graph although by far not as good as the second factor.



(a) one step of feedback

(b) two steps of feedback

Figure 4.22: *PR graph* of different learned factors with feedback.

Table 4.8 underlines the results already obtained from the *PR graph*. The first factor does not give very good results for any of the measures. The *precision* values are still better than the original *idf* system, but the ranks are much worse, and the results of the squared factor get even worse than when using the *idf* weighting. a_2 though, seems to be much better, even for the early *precision* values which are up to 10% better. The rank values are also slightly better, and only $R(P(.5))$ drops below the original value for the squared and cubed version.

The second step of feedback in Figure 4.22(b) shows that the first factor a_1 gives bad results, but that the second factor a_2 delivers better results than the *idf* system from the beginning on. Only a_2^3 is slightly worse in the beginning part of the graph, but eventually becomes the best curve in the middle part, more than 10% better than the original curve of the *idf* weighting.

The measures in Table 4.9, again, underline the results from the *PR graph*. The first factor is not significantly worse for queries with multiple images whereas the second factor seems to be getting better, the more images there are in the query. The *precision* values are up to 10% better and also the ranks are better than in the *idf* system. The cubed system does not have a relevant image as top–ranked response for every query, but this can be explained with one rather bad query result, whereas all other queries are processed significantly better.

	idf	a_2	a_2^2	a_2^3	a_1	a_1^2
r	65.14	65.14	65.14	65.14	65.14	65.14
$Rank_1$	1	1	1	1.43	2.79	11
$R(P(.5))$	0.5157	0.5799	0.4483	0.4965	0.437	0.4184
$\overline{Rank}$	162.3	159.9	158.5	156.7	189.6	205.3
$\widetilde{Rank}$	0.1429	0.1402	0.1387	0.1367	0.1728	0.1901
$P(20)$	0.6857	0.7607	0.7642	0.7857	0.7071	0.7071
$P(50)$	0.5014	0.5443	0.5643	0.5771	0.5357	0.5214
$P(r)$	0.4957	0.5504	0.5680	0.5887	0.5303	0.5129
$R(100)$	0.5640	0.5935	0.6030	0.63	0.5737	0.5519

Table 4.8: Performance measurements for different factors with one step of feedback.

	idf	a_2	a_2^2	a_2^3	a_1	a_1^2
r	65.14	65.14	65.14	65.14	65.14	65.14
$Rank_1$	1	1	1	1	3.79	10
$R(P(.5))$	0.58	0.5558	0.5775	0.6284	0.5185	0.4375
$\overline{Rank}$	149.8	134.49	136	133.9	188.3	198.8
$\widetilde{Rank}$	0.1292	0.1123	0.1140	0.1116	0.1713	0.1830
$P(20)$	0.775	0.8143	0.8107	0.8464	0.7393	0.7286
$P(50)$	0.5414	0.5843	0.5828	0.6	0.5271	0.5214
$P(r)$	0.5402	0.5871	0.5933	0.6229	0.5248	0.5166
$R(100)$	0.6153	0.6594	0.6539	0.6838	0.5794	0.5692

Table 4.9: Performance measurements for different factors with two steps of feedback.

We can see clearly that the first factor a_1 was unfortunately not suitable for queries with feedback although the results for one–image queries were quite good. The second factor we calculated led to very good results. The fact that the results are even improving stronger for feedback queries is surprising as we thought that the feedback was partly already taken into account with the calculated factor. This does not seem to be the case and the results are very good. In fact, the improved results with feedback can be explained by the fact that more relevant images for positive feedback could be found and thus, the query formulation could be improved.

As always, the results depend on the quality of relevance feedback we can gain from the log files. If people use much positive and negative feedback, the results can be very good and a factor in a higher power can be used, which promises the best results. Even if the quality of the interaction stored in the log files is not clear, a simple factor can be used, and the results in our test show that even this significantly improves the performance of our system. As our feedback is taken from a web demonstration, we cannot necessarily trust the users. Users of such a web demo may try to feed contradicting queries to the system to see how it reacts. When a similar evaluation is done with the use of the system in a professional environment such as trademark search, better results can be expected as the user feedback will most likely be more consistent and of a better quality.

4.7 A hierarchy for learning

In the preceding sections, we already explained several ways for learning feature weights. Learning over several temporal scales was shown to be an important part of the learning. We can thus see that not only we can learn from the images actually in the query, but also from the entire chain of images that a user saw and eventually marked as relevant or non–relevant. User profiling is also a well known research topic in many domains. In [202], for example, new web pages are proposed based on other web pages that the user marked as interesting or non–interesting beforehand. The additional feature importance a that we calculated in the preceding section can also be calculated on a user basis as the user name is stored in the *MRML* communication used for this analysis.

Therefore, such a feature importance can be calculated on various levels as well, for example, on a user level, a database level as in the preceding section and an overall level, although learning over several databases did not produce very good results in [175].

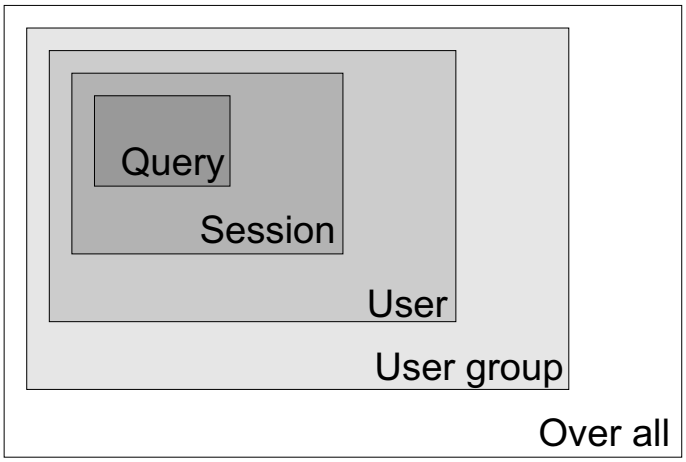


Figure 4.23: A hierarchy for learning.

Based on this study it is proposed to incorporate into retrieval systems a hierarchy of learning from user interaction as illustrated in Figure 4.23. Such a system would take into account the information that the users give at several scales. On a query basis, important features of a query can be identified and pronounced as being important for the query calculation. On the session level, we can take into account all the images a user saw and/or marked in a series of queries for a specific goal. This can already be done at the interface level where negative feedback can be kept until the user signals to start a new query task. Thus we have to give the user the possibility to tell the system when his search task has changed. A system that does in session learning is for example described in [82].

In addition to these two short–term learning techniques, we can do profiles on different bases. First of all, we can make a user profile, where we can try to identify the importance of features for a certain user. Especially repetitive search tasks or the search for similar images can be enhanced. The second level is on a user group or database level, where common interests or common features of importance can be assumed. Here, we can use as well an importance factor for features such as the one described in Section 4.6.1. This has proven to lead to a significant improvement in retrieval performance.

A learning over a number of different databases was reported in [175]. This did not lead to the strong improvements that could be reached when learning over the same image database, but still the results improved and did not get worse. A combination of all these techniques might lead to another improvement in retrieval quality.

4.8 Summary

This chapter analyzes the various aspects of user interaction in image retrieval. It is explained how important the *interaction speed* is for interactive information systems. As the study in [151] shows that, often, speed is even more important than quality, it does make sense to let the decision up to the user to define a tradeoff between speed and retrieval quality. Several methods for *search pruning* to reduce the response times of the *Viper* system are implemented and the experimental results are shown.

An overview of several techniques for *relevance feedback* in content–based visual information retrieval is given. This includes a comparison of strategies for users to feed back images to the system. A solution to the problem of excessive negative relevance feedback is proposed with the

algorithm already used by *Rocchio* for separately weighting positive and negative query components in text retrieval. The algorithm is shown to work as well in image retrieval.

Long–term learning and learning over several temporal scales from user log files is discussed in the remainder of the chapter. A comparison with the *market basket analysis* is made. This learning from a large amount of user interaction proves to be very valuable and it can also be used within a *hierarchy of learning* over several temporal and logical scales.

Chapter 5

Evaluation of Retrieval Systems

> If two scientists disagree on any issue, and the issue is within the ambit of science, then it must be possible for them to agree on a procedure which they can both accept as a critical test of their points of difference. For reasons of personality they may not be able to get together to work out such a test procedure, but it must exist as a possibility. If such a critical test cannot be imagined as possible, then the issue between them is not a scientific issue. The scientific method does not vary with the subject matter, but is the same irrespective of its results and basically the same in all the sciences.
>
> L. T. Wilkins, *Social deviance*[297], page 4. (taken from [41])

Evaluating research results has always been a very important part of the scientific method. Without a critical evaluation nothing can be said about the quality of the research. Thus, evaluation is essential to show progress and, without proper evaluation, progress is almost impossible to prove. Still, in CBIR, no benchmark or commonly accepted image database has been created, yet.

In several other domains, benchmarks have advanced the fields significantly and led to increasingly improving results. At [1] an online book about benchmarks with many examples can be found. Besides the text retrieval benchmarks (such as TREC) there are benchmarks or at least accepted document collections in the closely related fields of:

- Character recognition, with the US Postal database(used in [52]);
- transaction processing in the database field with the TPC[2];
- segmentation benchmarks with hand segmented images ([154]);
- compression benchmarks (images to compare results on available at [3], [4] and [5]);
- speech recognition (ARPA benchmark used in [74], NIST benchmarks[6]);
- for computer systems performance, SPEC[7].

It is commonly acknowledged in these areas that benchmarks did trigger significant improvements.

There are however critical voices who say benchmarks could block innovations because new techniques can only be presented if they perform better than the existing benchmark results. This certainly favors small changes in existing algorithms to optimize performance, and makes it harder for completely new techniques to emerge. New techniques, for example in speech recognition, might not have as good results from the start, but they might still have the potential to become better

[1] http://www.benchmarkresource.com/handbook
[2] http://www.tpc.org/
[3] http://www.dcs.warwick.ac.uk/nasir/work/benchmark
[4] http://silicon.terra.vein.hu/~kepaf/TestImages/
[5] http://jiu.sourceforge.net/testimages/
[6] http://www.nist.gov/speech/tests/index.htm
[7] http://www.spec.org/

when they are well studied and optimized. All this has to be taken into account when talking about benchmarking. Such an effect of favoring small changes of existing techniques can also be observed when using techniques to obtain relevance judgments by pooling, as explained in [313]. Still, the concensus is that the advantages in employing benchmarks certainly and by far outweigh the disadvantages.

5.1 Evaluation in text retrieval

Text retrieval is a very mature research field compared to image retrieval and the evaluation issue already became important in the early 1960s.

5.1.1 History

Maybe the first systematic performance evaluation efforts in information retrieval were the Cranfield tests [40, 41] in 1962 and 1966. They were certainly one of the earliest benchmarking events in text retrieval, and the collections and relevance judgments created continued to be used for many years after they were created. The Cranfield II studies [41], for example, had substantial impact on the field of information retrieval. They showed that the performance of manual indexing is comparable to automatic indexing. This fact was long neglected and it was thought that annotations of experts would always lead to better results. The *SMART* retrieval system [226], created between 1961 and 1964, is another early system that was elaborately studied and used for systematic evaluation [225].

Much effort was also put into the creation of proper test collections for text retrieval [257] (with a more detailed description in [256]). Several test collections like the CACM collection (1983) and the NPL collection (1978) were subsequently created.

Another issue is of course to have consistent series of evaluation events based on each other such as the TREC for text retrieval. A good quote, that is currently also valid for the CBIR field, was given in [255] in 1981:

> Yet the most striking feature of the test history of the past two decades is its lack of consolidation. It is true that some very broad generalizations have been endorsed by successive tests: [...] but there has been a real failure at the detailed level to build one test on another. As a result there are no explanations for these generalizations, and hence no means of knowing whether improved systems could be designed. (page 245)

An overview of performance evaluation in text retrieval is given in [227], and [225] describes several parameters for system evaluation.

Creating a document collection for evaluation purposes

Starting with the Cranfield test collections that were used for a long time, there is a history of creating document collection for evaluation purposes in text retrieval. Much more than in image retrieval, there has been a quest to supply collections for standardized evaluation. One very detailed article on the need for such a collection is [256], that also states exact values about the amount of maintenance that such a collection needs to stay up–to–date in man months and in the amount of money needed for the maintenance. A more compact version can be found in [257].

TREC basically wanted to show with its extremely large collections that systems can be evaluated on real–world–sized collections. The first TREC in 1992 already contained 2 GB of textual data take from publications like the "Wall Street Journal" and "Newswire".

5.1.2 TREC – Text REtrieval Conference

TREC was initiated by NIST[8] (National Institute for Standards), DARPA (Defense Advanced Project Research Agency) and is strongly supported by the DoD[9] (Department of Defense) to give

[8] http://www.nist.gov/
[9] http://www.dod.mil/

a boost to the text retrieval community and to really be able to compare the system performances based on large, realistically sized test collections. The first TREC was held in 1992 and it has been held every year since. The TREC benchmark has become the veritable standard in the field. More information on TREC can be found on their web pages[10]. In [91], a very good overview of TREC–1 is given and many of the goals are summarized. One citation from this overview is also particularly valid for the CBIR field although CBIR is not as old and has not yet reached this stage:

> In the 30 or 50 years of experimentation there have been two missing elements. First, although some research groups have used the same collections, there has been no concerted effort by groups to work with the same data, use the same evaluation techniques, and generally compare results across systems. The importance of this is not to show any system to be superior, but to allow comparison across a wide variety of techniques, much wider than only one research group would tackle.

Overviews of TREC–7 and TREC–9 in [287, 288] show the evolution of the benchmark. During its ten years of existence, many new topics and fields have been added to TREC, such as the interactive TREC in 1994, and the multi lingual track in 1997. A multi lingual track is very important for a multi lingual country such as Switzerland. The spoken document retrieval test in 1997 introduced the first media other than text, and with the video track in 2001 the first use of visual data was included into the TREC benchmark.

In general TREC always follows the same circle of events every year as shown in Figure 5.1, taken from[11]. Normally, the different parts of the circle are done from February to November of the same year, as for example in 2001.

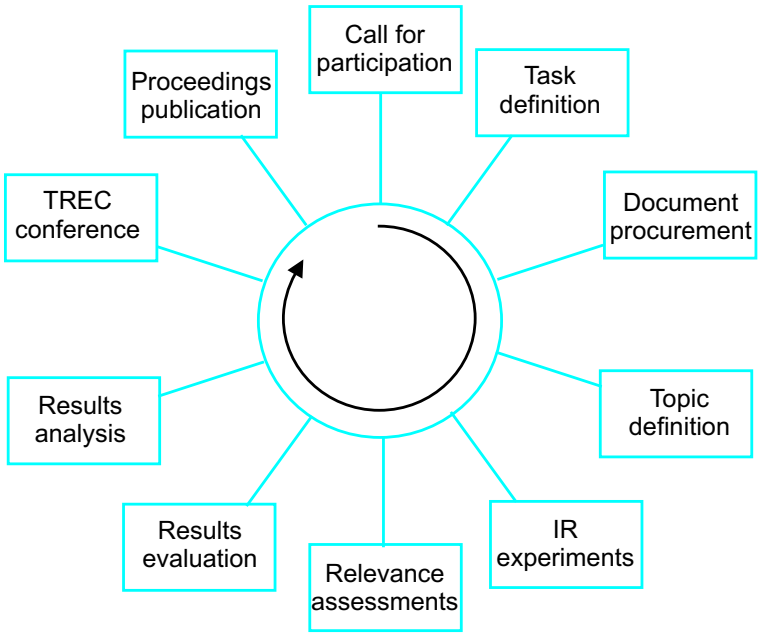


Figure 5.1: The TREC circle of events for every conference.

This means that after a *call for participation*, the *tasks* are defined, and the *documents* for the test are being gathered. After the *topics* are defined, they are send out to the participants with the documents. The participants then have a limited time (*i.e.* around five months in 2001) to submit the *retrieval results* for the different topics and tracks. After the results are submitted, the results can be used to create the pools (described in Section 5.1.2) to do the *relevance assessments* on.

[10] http://trec.nist.gov/
[11] http://trec.nist.gov/presentations/TREC10/intro/index.html

With these judgments, the systems can be *evaluated* and the results can be *analyzed*. At the actual *conference*, the participants can present their systems and together with the evaluation results a comparison of different techniques is possible. The selection of papers accepted for oral or poster presentation is not based on the actual result of the benchmark. At the end the techniques and the evaluation results are printed in the *proceedings* of the conference.

Collections

TREC creates a new collection every year that is given out to the participants a certain (limited) time before the actual conference. Old collections can as well be used for testing, but to obey copyright issues of the used data, the collections are only available to participants. In general, the collections are becoming larger in size to keep up with the demands for large databases in the field. The goal is to have realistically–sized test collections.

Already at the first conference in 1992, TREC used a collection of over 50 topics containing 2 GB of text. In general, newspaper (*i.e.* Wall Street Journal) or newswire articles are taken into account as full–text documents. No correction of spelling mistakes or formatting errors was done, to have the task as realistic as possible.

In [12] it is cited that results are only comparable when they are obtained from the same document collection. For this reason, comparing of system working on different collections is not a goal of TREC!

Topics

The topics are the query tasks of TREC. The number of the main topics rose from 50 in 1992 to 500 in 2001. In general, every topic is defined and formulated by a domain expert who is also responsible for assessing the relevance judgments for this topic. Thus, the relevance judgments are more consistent and less ambiguous than if they were created by different people for a given topic.

In 2001, there were different kinds of topics ranging from very detailed textual descriptions to very general 2 or 3 word queries. Often, constraints are added such as restricting articles to a particular subject in a certain country, or excluding a certain country.

Relevance judgments

The relevance judgments in TREC are generated after all participating systems have submitted their results for the queries (topics). This reduces the possibilities of "cheating" and it also allows for efficient pooling of the documents that have to be judged to gain the relevance judgments.

Important for judging relevance, of course, is the definition of the word *relevance* for a document in the context. Articles that present different definitions and discussions of relevance can be found in [163, 236, 237]. TREC uses the following working definition of *relevance*:

> If you were writing a report on the subject of the topic and would use the information contained in the document in the report, then the document is relevant.

In general, only binary judgments ("relevant" or "not relevant") are made, and a document is judged relevant if any piece of it is relevant (regardless of how small the piece is in relation to the rest of the document).

All the participating systems have to send in the highest scoring N documents as response set where N has gotten up from 200 to 1000 over the years. Too small a size N of the response set can make the evaluated *recall* results less correct. [91] explains this effect, where it is stated the the *recall* results start to be incorrect at roughly 40% *recall* when using only 200 documents for the calculation of the performance measures. The same effect was observed in [181] for incomplete retrieval of search results.

For not having to judge the entire set of roughly 1 million documents in the collection, a pooling method was implemented as explained in [256]. This method pools part of the top N

[12] `http://trec.nist.gov/presentations/TREC10/intro/index.html`

results (normally $N_{pool} = 100$) submitted from each system. The set containing all the top N_{pool} results of every system is then judged for relevance by an expert. Normally, each topic is judged completely by the same expert to reduce the influence of user subjectivity. Unjudged documents are assumed to be irrelevant.

Although this pooling can affect the results, it is shown in [313] that the changes are rather small and not biased. The alterations are also shown to be below the level of user subjectivity. It is also argued, that it is likely that at best 50%–70% of the relevant documents have been discovered as the database is very large and only a small part has rated for relevance. It is proposed to enlarge the number of documents that is used for the pooling.

Such a pooling can favor techniques that are variations of other systems where results have been used in the pooling. New technologies with a large number of unjudged documents can thus be disadvantaged slightly. One possibility of avoiding this is to report the number of unjudged documents taken into account for the evaluation of each system. Another possibility is the use of variable–sized pools for each query and an estimation of the expected number of relevant documents when enlarging the size of the pool.

Performance measures

TREC uses a variety of performance measures to show different aspects of evaluation. In 2000, a set of 85 performance measures, mostly based on *precision* and *recall* were computed. It is very important to state that the absolute values of the measures are of no importance [91] as the comparative evaluations are regarded to be the main goal of TREC. It is also considered as important to have a set of measures that are easy to be interpreted. Most measures are based on *precision* and *recall* defined as follows:

$$Precision\ P = \frac{rr}{N} \tag{5.1}$$

$$Recall\ R = \frac{rr}{r}, \tag{5.2}$$

where rr is the number of relevant documents retrieved, N the number of documents retrieved and r the total number of relevant documents in the database, see also Table 5.1. In this context we use the same notations as for the image retrieval part, where $\mathcal{I}$ is the set of documents (document collection) and $\mathcal{R}$ a set of relevant documents for a certain query task. Correspondingly, $N_{\mathcal{R}}$ and $N_{\mathcal{I}}$ are the numbers of documents contained in the sets.

	relevant	non–relevant	
retrieved	rr (true positive)	nrr (false positive)	$N = rr + nrr$
not–retrieved	rnr (false negative)	$nrnr$ (true negative)	$rnr + nrnr$
	$r = N_{\mathcal{R}} = rr + rnr$	$nr = nrr + nrnr$	$N_{\mathcal{I}} = rr + nrr + rnr + nrnr$

Table 5.1: Overview of relevant and non–relevant items retrieved and not retrieved.

Some actual measures used in the main TREC session are:

- $P(10), P(30), P(N_{\mathcal{R}})$ – the *precision* after the first $10, 30, N_{\mathcal{R}}$ documents are retrieved, where $N_{\mathcal{R}}$ is the number of relevant documents for this topic.
- *Mean Average Precision* $\overline{P}$ – mean (non–interpolated) average *precision*.
- $R(P(0.5))$, *recall* at .5 *precision* – *recall* at the rank where *precision* drops below .5.
- $R(1000)$ – *recall* after 1000 documents are retrieved.
- $Rank_1$,Rank first relevant – The rank of the highest–ranked relevant document.
- *Precision* vs. *recall (PR) graph*.

These key numbers offer a set of performance descriptors, so that different systems and techniques can be compared meaningfully and objectively. A list with the correlation coefficients of these measures can be found at [13].

[13] `http://trec.nist.gov/`

Video TREC

Since 2001 TREC also contained a track for visual data, the so–called video–TREC[14]. This video–track contained two different tasks to accomplish for the participating systems:

- Shot boundary detection for videos,
- Return of a ranked list of items for given query tasks.

Participating groups were allowed to use the speech part of the video, but they had to state this and they had to state results based only on speech separately, to evaluate the influence of the additional visual information.

Search tasks for 2001 included searching for specific people (for example, "Ronald Reagan") as well as searching for objects (for example the "space shuttle"), and activities (such as "Ronald Reagan reading about the space shuttle"). Also for activities in conjunction with certain objects and certain persons was searched.

Other tracks in TREC

Depending on the need and the demand, TREC adds new tracks to the conference. Additional tracks are, for example:

- Cross language information retrieval since 1997, [223],
- Web searching since 1999,
- Interactive TREC since 1994,
- Spoken Language since 1997,

These additional tracks are meant to get a larger number of research groups interested in participating in TREC and to follow current developments and hot topics.

5.1.3 Other developments

Much criticism of evaluating retrieval systems using *precision* and *recall* was used in the text retrieval domain [19] and in image retrieval [59]. Consequently, methods were developed to evaluate such things as interactive systems [19].

Also, experiments to change the relevance definition of web documents using three levels of relevance ("not relevant", "relevant", "highly relevant") [288] were presented in TREC.

5.2 Evaluation in visual information retrieval

Visual information retrieval is a much younger area of research than text retrieval. Although some applications of image databases started already in the late 1970s and early 1980s [35], the first real influential applications, according to Smeulders *et al.* [247], were not created before the early 90s, such as Kato's trademark retrieval system [116].

Thus, development, and also evaluation, is much behind when compared to the text retrieval community. Several other techniques such as relevance feedback, inverted files, and frequency–based weights were adopted from the text retrieval into the image retrieval community, so it does make sense to learn from performance evaluation in text retrieval as well as to adopt some of their techniques.

At the moment, with an exploding number of papers on CBIR at any computer vision, image processing and multimedia conference, evaluation gets more and more important as it is practically impossible to compare any two systems (even when evaluations are published in the papers). At the ACM Multimedia conference 2001 there were, for example, more than 20 papers on image

[14] `http://www-nlpir.nist.gov/projects/trecvid/`

retrieval but there was not a single database used twice, even though there were papers from the same research labs.

Only via a proper, profound, and standardized evaluation can we measure the success CBIR has made over the last years. Only with such an evaluation can we distinguish promising techniques from vain ones and advance the field into the direction of the good ones.

One common view for how a system's performance on a database of trademarks can be evaluated was given in [132]:

> There is no quantitative performance measure for similarity retrieval of trademarks for the very simple reason, that the judgment of similarity is very subjective and context dependent. We can only use guidelines given to us by the trademark office as well as common sense when evaluating the performance of our system. Using such an evaluation criteria, we have obtained some reasonably good results for our system as presented below.

The fact that many groups were thinking like this hindered the development of performance measures. Fortunately, by now, we are a bit further with respect to evaluation of retrieval systems and the importance of proper evaluation is clearly seen. The development of a benchmarking event such as the *Benchathlon* [15] expresses this growing importance.

In general, we can state the problem of evaluation on an image database that spans an image set $\mathcal{I}$ as follows: From the image set $\mathcal{I}$, we have to chose a number of query images that represents the evaluation set of images $\mathcal{E} \subset \mathcal{I}$. For every query image $\varepsilon_i \in \mathcal{E}$ we have to find l sets of relevant images $\mathcal{R}_{il}$ where the l relevance sets represent different user opinions. Often $l = 1$, when only one set of relevance judgments is available. We can then compare the answer sets of images $\mathcal{A}$ that the system returns with the $\mathcal{R}_{il}$ of the different user opinions and the performance measures should then show how well the results correspond to the different opinions. In the evaluations we do, we retrieve the entire database as a ranked set $\mathcal{A}$, so $\mathcal{A} = \mathcal{I}$. This gives us the possibility to compute global measures and also measures that assume a smaller $\mathcal{A}$ as the system's response.

5.2.1 History

The first evaluations of CBIRSs were mostly restricted to showing a few retrieval results [71] as screen–shots. A first comparison of several (three) systems is given in [308], but also in this paper only three example results were printed and the reader has to judge the results.

An important initiative to create a framework to evaluate CBIRS was the European Union (EU) project *MIRA* (Evaluation frameworks for Interactive Multimedia Information Retrieval Applications)[16]. The framework was basically created by researchers in information retrieval. Several workshops and conferences were held within this framework [63].

The first approach for a systematic evaluation of retrieval systems was given in [192] (1997). This paper gives a good overview of retrieval systems and elaborates a way in which evaluation measures could be constructed. No concrete performance measures were proposed and no evaluation was done with any example system to show the effectiveness of the measures. The use of exact rankings of the result items, as proposed in the article, can also be seen critically as it is very hard and subjective to exactly rank items based on their visual similarity with a query image.

In 1998, John R. Smith proposed to use techniques already well known from text retrieval, especially performance measures [249], to evaluate CBIRSs. He proposed using TREC as a role model. He mentioned some image collections, but no real framework was given, and no example evaluation with a system based on the proposed measures was done.

In [59] (1999) a measure to compare systems based on the rankings of relevant items is proposed. The use of a single measure might look practical to compare the performance of one system with another, but it is very hard to interpret the measure and to get more information about the system performance. A single measure cannot model all aspects of performance of a system. Especially considering the fact that images having a high rank are as strongly weighted in the measures as

[15] `http://www.benchathlon.net/`

[16] `http://www.dcs.gla.ac.uk/mira/`

images with a low rank is not very practical. Users normally want to know how many relevant images they see on their screen. The number of images a user really looks at is normally in between 20 and 50, so this part of the response set should be weighted higher.

In 2000, Clement Leung proposed constructing a benchmark and suggests sizes of databases, relevance judgments and performance measures as important considerations [137]. The evaluation of *relevance feedback* is regarded as an integral part of a system's performance. He proposes using a relatively small database of roughly 1000 images and also relatively small sets of relevance judgments for a number of selected query images. No example evaluation was done to verify the usefulness of the proposed measures.

In [185], Wolfgang Müller proposed a benchmark for image browsers to automatically evaluate the performance of image browsers such as `PicHunter` [49] or *TrackingViper* [190]. To do so, he used structured annotation and shows that with massive annotation a human's opinion can be well modeled. Like this, he can perform the *target test*, as described in [49], in a much more objective, practical way without the need for human intervention. Unfortunately such an extensive annotation takes a very long time to be generated, and so it might only be feasible for small image databases, like the one used, containing 500 images.

Also in 2000, a performance measure was proposed in [125] to compare systems. The measure basically characterizes how close together certain groups of images are retrieved. Again, a single measure is very hard to judge and neither contains much information nor is easy to interpret.

In [178] an overview of performance measures used in CBIR is given. Based on the TREC benchmark for text retrieval, a proposition is made on how to create a benchmark for image retrieval using techniques and measures similar to TREC. An example evaluation is given for one system with four queries of varying difficulty. Relevance feedback is regarded as an integral part of the evaluation

In [66, 282] an almost exhaustive list of systems was created and the techniques of the systems were described. A performance evaluation of systems where executables were available was planned, but unfortunately only a few of the commercial systems were compared with example query results and not with any other performance measures.

The *BIRDS–I* benchmark [89] was the first paper describing the actual performance measures and techniques proposed for the *Benchathlon*. A "lead measure" analogous to that used in MPEG–7 and [224] is proposed that is basically a normalized average rank measure with a penalty for "non–retrieved" relevant images (see Equation 5.5).

[172] basically uses the propositions for performance measures made in [178] but takes a completely automatic approach for evaluation. The benchmark consists of perl scripts configured by simple text files. *MRML* [17] is used to automatically execute the queries. Based on the relevance judgments and the system response, relevance feedback is generated automatically. Like this, the fully automatic evaluation of CBIRSs is possible, including the evaluation of several steps of relevance feedback. In [173], an extension to this automatic benchmark is proposed by using a web interface to evaluate the system performance. This allows quick and easy performance evaluation via the WWW. It is regarded as a complement to the *Benchathlon* event, so groups can try out the infrastructure and get used to interpreting the performance measures. A sample evaluation of one system with a freely available image database [18] is done. Images and relevance judgments used are made available.

In [98], the *generality* (see Equation 5.3), a measure that can be used to describe the difficulty of a query based on the collection size and the number of relevant images, is proposed.

$$\textit{generality } g = \frac{\textit{number of relevant items}}{\textit{database size}} = \frac{r}{N_{\mathcal{I}}} \tag{5.3}$$

To correct the *PR graph* by the influence of collection size and the size of the relevance set, it is proposed to use the plane where *precision=recall* in the three–dimensional diagram showing *precision*, *recall* and *log(generality)*. *Generality* can reduce the influence of the database size $\mathcal{I}$ and of the size of the relevance judgments $\mathcal{R}_i$ but it cannot detect changes in the query tasks $\mathcal{E}$ and thus

[17] `http://www.mrml.net/`

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

about the difficulty of the queries. The question is as well, how useful it is to compare systems over different databases. TREC, for example, regards the comparison of system over different databases as not very useful[19]. The varying difficulty of databases might have a stronger influence on the performance results than the number of relevant images r and the database size $N_{\mathcal{I}}$.

At the SPIE 2002 conference Photonics West, in the session Electronic Imaging, a special session was held on the *Benchathlon* framework with several papers proposing example queries [33], annotation software [206] and ways to gain relevance judgments from annotation [115].

5.2.2 Problems

The evaluation of CBIRSs has many problems and issues to solve, similar to those already encountered in text retrieval:

1. Create a number of *image test collections* that are freely available and that are accepted for evaluation in the field to form an image set $\mathcal{I}$.
2. Select a number of *test queries* or query tasks. In the case of QBE, query images span an evaluation set $\mathcal{E}$. In TREC, these are called *topics*.
3. Gain several *relevance judgments* for these test queries to form the relevance sets $\mathcal{R}$.
4. Choose a set of descriptive *performance measures* that are easy to interpret. These measures are supposed to compare the system's ranked answer $\mathcal{A}$ with the relevance sets $\mathcal{R}$.
5. Choose a *technical infrastructure* to automate the retrieval as much as possible.
6. Motivate several research groups *to participate* in a comparison of systems.

Whereas there are several propositions for the first five of these points (detailed below), the hardest task is to motivate research groups to participate. Normally, this should not be a big problem because everybody would profit from the fact that the systems can be compared, many new techniques can be tried out and their use can be evaluated. Unfortunately, this is at the moment not the predominant opinion and it looks more like people fear to get into a "competition" and that insufficient results might reduce the research money they obtain. One of the goals of an organizing body for the evaluation system has to be therefore that of supplying research groups with information on the benefits of evaluation standards as do other benchmarks in fields like text retrieval or transaction processing.

5.2.3 Document collections

There are hundreds of different document collections. In this section, only those collections that were used to evaluate our system are presented. A description of the test collections available for the demonstration of our system can be found at [20]. This includes the most frequently used collection in CBIR (Corel[21]) and a collection that is available without copyright and free of charge (from the University of Washington[22]).

Corel photo CDs

The Corel Photo CDs are by far the most commonly used image database in CBIR [25, 29, 112, 136, 198, 211, 277, 278, 291, 309]. There are more than 800 image sets that can be bought on CD, each containing 100 images of a certain theme. They are relatively expensive as each group has to be bought separately. Figure 5.2 shows some examples of the 61 groups we used, with a total of 6100 images.

[19] http://trec.nist.gov/presentations/TREC10/intro/index.html
[20] http://viper.unige.ch/collections/
[21] http://www.corel.com/
[22] http://www.cs.washington.edu/research/imagedatabase/groundtruth/

Figure 5.2: Sample images from the Corel database.

The images give a good overview of the themes of the database. The obvious advantage is that 100 images of a broadly same subject are grouped together and thus, some form of implicit relevance judgment $\mathcal{R}$ exists for every image in the database, namely the other images on the same Photo CD. The problem is that every research group is using a different subset of the more than 800,000 images currently available, making inter system comparison impossible.

Further, in Section 5.7 we will show how strongly dependent the apparent performance of a system is on the image sets and the query images used.

Database of the University of Washington

The database of the University of Washington[23] was created in 1999 to make available a collection of images free of charge and without copyright for benchmarking and system testing. This database was created by Professor Linda Shapiro and we helped to enlarge it with several image sets.

Figure 5.3 shows a couple of example images of the Washington database. The version we used for this thesis contains $N_{\mathcal{I}} = 911$ images of 13 different groups ("Arbor greens", "Barcelona", "Barcelona2", "Cannon beach", "Cherries", "Football", "Geneva", "Green lake", "Greenland", "San Juans", "Spring flowers", "Swiss mountains", "Yellowstone"). The groups contain between 27 and 256 images. Within the groups there are sets of very similar images, but also several groups of visually dissimilar images.

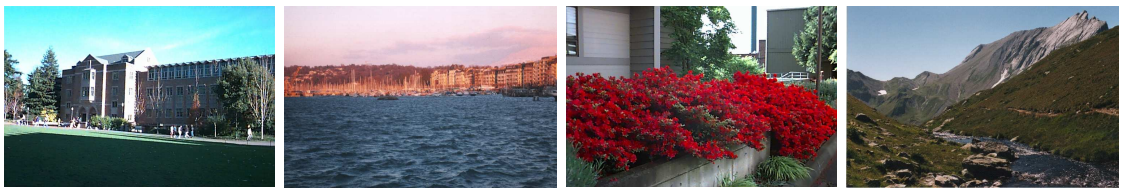


Figure 5.3: Sample images from the database of the University of Washington.

As this database is available free of charge and without copyright, we propose using it for

[23] http://www.cs.washington.edu/research/imagedatabase/groundtruth/

evaluations because the results can be reproduced by everybody. Unfortunately, the database is still relatively small, but it is in the process of being enlarged.

TSR

The first database we used was provided by the Swiss television TSR (Télévision Suisse Romande). A set of roughly 10,000 images was available on an analog video tape and had to be digitized with a frame grabber. Figure 5.4 shows 8 example images from the collection. In general, the collection rarely contains sets of visually very similar images, but it does contain images that are roughly similar in theme.

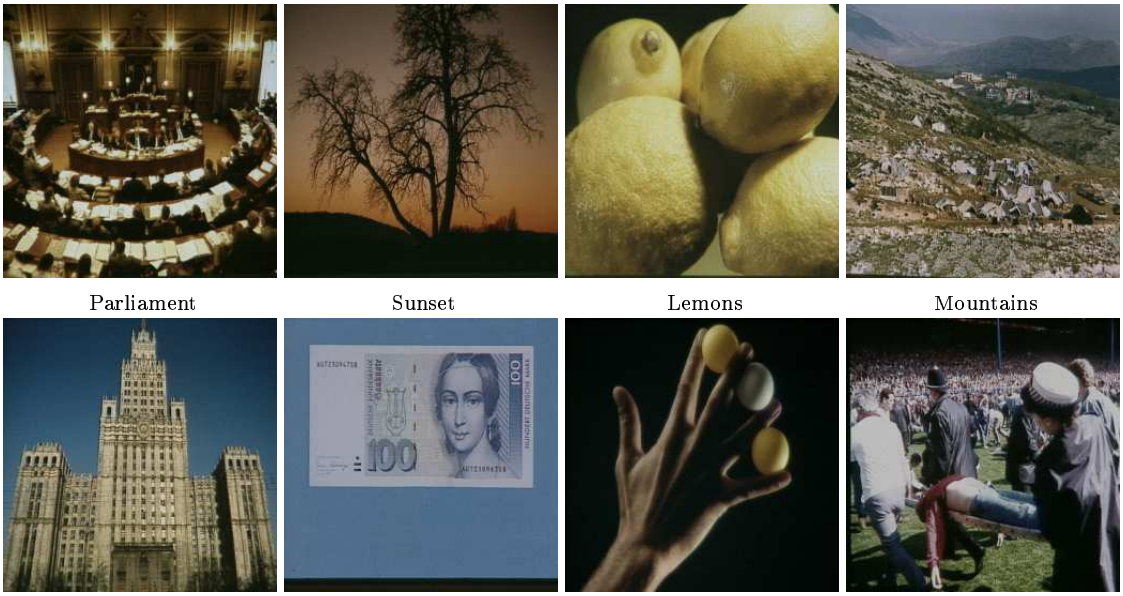


Figure 5.4: Sample images from the database of the Télévision Suisse Romande (TSR).

As the database itself does not contain any grouping, query images and relevance judgments had to be created. To do so, we created two subsets of images for user studies to get relevance judgments. Unfortunately the images are copyrighted and so we cannot distribute them.

TSR500 This subset contains $N_{\mathcal{I}} = 500$ images of the TSR collection. To evaluate our system with this database we chose 10 query images by hand to have a set of images of (subjectively) varying difficulties and with different subjects.

To obtain relevance judgments of real users we asked five test persons to look at every image in the database and sort out all the images that they regarded as being similar to the query image in question. No constraint was used on their judgment of "similarity".

The result was that the relevance sets $\mathcal{R}_{il}$ for the users varied largely in size (between 2 and 37 relevant images). Even for the same query image, the number of relevant images selected varied between 7 and 37. The study also showed that users choose very different images as being relevant for the same query image, confirming that image search is user–dependent and highlighting the need to get relevance judgments from several real users to best model user subjectivity.

TSR2500 The TSR2500 image database contains 2500 of the TSR images. Because the TSR500 images were rather dark in appearance, we normalized the images after having them grabbed from the analog video tape. Thus, they are slightly different and more colorful than the TSR500 images.

Since our first user experiment showed how different the results chosen by users are, we again obtained real user judgments for this collection. This time, we chose a set of 14 images as query

images and had three users obtain relevance judgments for these 14 images. Looking at a database of 2500 images proves to be much more work intensive than for a database of only 500 images. It took an average of 1 hour to assess the relevance judgments for one query image. In this context, time is an important argument since it can be argued as that a user who starts to get tired will choose different images than when not being tired to speed up the process.

The quality of the relevance results was very similar to the results of our first user test. The users chose different results for the same query image, and the sizes of the relevance sets varied considerably (between 2 and 36).

VisTex

The VisTex texture database is available at[24] and contains a set of different textures. Normally the large textures are cut into blocks, thus creating groups of images with roughly similar but not identical textures.

Figure 5.5 shows a few example textures of the VisTex database. The database we used contains 1670 images with 18 groups of similar subjects such as "Leaves", "Bricks", etc. Each group has between 3 and 17 subgroups with 10 images each.

Figure 5.5: Sample images from the VisTex database.

Benchathlon

The Benchathlon initiative is trying to set up an image database that is free of charge and without copyright restrictions as well. It is available from [25] and is supposed to be updated regularly.

To gain relevance judgments, there are plans to use annotation and then generate relevance judgments based on the annotation [115, 206]. So far, no relevance judgments and no set of query images exist for this collection.

[24] http://whitechapel.media.mit.edu/vismod/imagery/VisionTexture/vistex.html
[25] http://www.benchathlon.net/

Others

The *Brodatz* database is another frequently used image database of texture images, but unfortunately, it is not freely available or at least not anymore. Many groups scan the images and then perform tests on them. In general the characteristics of the images are similar to the VisTex database.

The databases for *MPEG–7*[26] [27] are also a widely used set, but unfortunately, it is not allowed to show them on the web and even not to show them in publications, although this is sometimes done.

Another database is the image database of *Columbia University* that is used in [275]. A web page with a larger number of vision test databases is available at [28].

5.2.4 Query tasks

In the case of evaluating QBE systems, the query tasks are normally images that are used for the querying, that we call $\mathcal{E}$. In TREC these query tasks are called topics, and normally the topics include a more or less detailed description of what needs to be found from the database.

In CBIR, these query images can be chosen randomly from a collection or they can be chosen based on visual characteristics to represent a wide spectrum of the database. When using database like Corel or Washington, normally one image per group is chosen as query image. This can be the first image of a group, a random image or it can be "optimized" as in Section 5.7.2. Depending on the method of choosing a representative image, the results may vary over a large span.

To evaluate images browsers, it might also make sense to propose more general concepts than just trying to find a known target image in the database. In [33] two query concepts are proposed to evaluate retrieval applications. This means that the problem of finding a good seed or starting image is also taken into account for the querying task.

5.2.5 Relevance judgments

In CBIR, there is not yet a common means of assessing relevance judgments $\mathcal{R}$ for queries. Even the inclusion of real users in the judgment process (as in IR) is not yet common.

The goal of the relevance judgments is basically to define what we would regard as being a perfect performance of the system, which can vary between different users. Thus, it is very important to properly gain relevance judgments, because they are the basis of the evaluation. A definition of "relevance" should also be clearly given, as it is done in text retrieval (Section 5.1.2).

Judgments from a collection with predefined subsets

A very common technique is to use standard image databases with sets of different topics (*e.g.* "sail boats", "tigers") such as the Corel collection. Relevance "judgments" are given by the collection itself since it contains distinct groups of (annotated) images. The choice of sets can greatly influence results, since some sets are visually distant from each other and others are visually closely related (see Section 5.7). Grouping is not always based on global visual similarity, but often on the objects contained. In some studies, images that are too visually different are removed from the collection, which certainly improves results (*e.g.* [9], Section 5.7.3)!

Judgments from experts

An alternative approach is for the collection creator or a domain expert to group images according to some criteria. The grouping is not necessarily based only on readily–perceptible visual features. Domain expert knowledge is frequently used for trademark retrieval ([67, 84]) or in medical CBIR (*e.g.* [64, 245]). This can be seen as real groundtruth, because the images are *i.e.* attached

[26] http://drogo.cselt.stet.it/mpeg/standards/mpeg-7.htm
[27] http://www.mpeg-7.com/
[28] http://www-2.cs.cmu.edu/~cil/v-images.html

to a diagnosis certified by at least one medical doctor. These groups can then be used in the same way as the subsets discussed above.

Judgments from real users

Judgments of real users are for example used in [264], but it also shows that real user judgments can be costly as explained in Section 5.2.3. Doing only part of the collection (the first n images the system returned) does not give a proper ground to evaluate the system [140], as nothing has been said about the large number of unjudged images that are assumed to be irrelevant. This does not allow to calculate proper *recall* values.

Pooling methods similar to those used in text retrieval ([256] and Section 5.1.2) can reduce the cost to gain these judgments, although other problems can arise from pooling, as explained in [313]. This study shows that pooling can influence the results slightly, especially for systems based on techniques that differ strongly from the other systems under test.

In [286] it is shown that the results when using one of several sets of relevance judgments do not significantly alter the ranking of systems, although the relevance sets are very different for every judge (only 30% to 40% overlap). This study uses the textual data from TREC–4 and TREC–6. It still has to be proven whether this is also true for the relevance assessment of images.

Judgments from annotation

In [115], ideas are given for an annotation scheme to generate relevance judgments for image collections. It is especially important to model the user subjectiveness and the fact that different users might use the same image as a query for completely different goals.

[196] already used textual annotation of a set of Corel images to judge the relevance of retrieval results of a CBIRS, but the few terms supplied by Corel might not be sufficient to produce reliable judgments. [172] shows how well extensive annotation can model the judgment of several users. The advantage of annotation is certainly that it can be reused when a database in being enlarged.

Simulating users

Some studies simulate a user by assuming that the users' image similarity judgments are modeled by the metric used in the CBIR system, plus noise [281]. Such simulations can provide very good results–indeed the quality of the results is controlled by the level of noise. Real users are very hard to model: [274] has shown that human similarity judgments seem not to obey the requirements of a metric, and they are certainly user– and task–dependent. Therefore, simulations cannot replace real user studies.

Other judgments

Some systems do not gain any user judgments before the actual query but only judge the N first images returned by a system to judge their relevance to the query [291]. This does not at all take into account unfound, possibly relevant items and users seem to be satisfied easily with what is returned when not knowing the entire database.

Summary

There are fundamental differences between these methods. The ease of obtaining relevance "judgments" is an advantage of using collections with pre–defined groups of similar images. User judgments can still be made for such collections. Domain expert knowledge should be used whenever available, such as in medicine and other specialized fields. For general CBIR tasks, we believe that the use of real users is essential (see [265], [151]). For a complete evaluation, the user with his/her expectations is a vital part of the system. The number of images a user must examine can be reduced by using pooling methods like in IR ([256]). Such a pooling does not significantly alter the results of a system because the first n relevant images of each system are in the pooling set. It is essential that the user examines a significantly large fraction of the database, and that

the relevance judgments are made in advance: users tend to be easily satisfied, even though the result may contain few, or even none, of the images selected as being relevant in advance. The characteristics of the group of users from whom the relevant judgments are obtained are also very important: CBIR system developers have different notions of image similarity from novice users.

Annotations imply the involvement of real users, which is expensive, and it might not be as easy to gain subjective opinions from annotations as it is with direct user tests. They certainly have the advantage that they can be reused to gain relevance judgments when the image database is being enlarged.

5.2.6 Performance measures

There is a wide variety of performance "measures" used in IR and CBIR. This section reviews methods used in the field of CBIR and gives references to their original definition and pinpoints their equivalence.

User comparison

User comparison is an interactive method. A group of users judges the success of a query directly after the query. It is hard to get a large number of such user comparisons as they are time–consuming.

Before–after comparison This is the easiest test method. Users are given two or more different results and are asked to choose the one which is preferred or found to be most relevant to the query. This method needs a base system or, at least, another system for comparison such as a simple histogram intersection.

Single–valued measures

Rank of the best match [14] measures whether the "most relevant" image is either in the first 50 or in the first 500 images retrieved. 50 represents the number of images returned on screen and 500 is an estimate of the maximum number of images a user might look at when browsing. The two figures then represent the facility with which the user will access the results.

Average rank of relevant images [73] use the average rank ($\overline{Rank}$) measure. It can give a good indication of system performance, although it clearly contains less information than a *PR graph*. It is vulnerable to outliers, since just one relevant image with a very high rank can adversely affect it. A simpler and more robust measure is the **rank of the first relevant image**, $Rank_1$, which is used in TREC and is very useful for CBIR as well.

In [89, 224] a normalized averaged rank $\widetilde{\overline{Rank}}$ was proposed that is defined as follows. First we define:

$$\widetilde{\overline{Rank}} = \frac{(\sum_{rr}^{i=1} Rank_i) + rnr \cdot \mu}{r} \tag{5.4}$$

where rr is the number of relevant retrieved, rnr the number of relevant non–retrieved within the N (*i.e.* $N = 50$) images retrieved, r is the total number of relevant images, $Rank_i$ is the rank of the ith relevant image and μ is a penalty for non–retrieved images. $\mu = 1.25N$ or $\mu = 2N$ are common values for the penalty. $\widetilde{Rank}$ is an average rank with a penalty for non–retrieved images. Then, we would like to have a normalized value within the interval [0..1] we can normalize by the maximum and minimum possible values $\widetilde{\overline{Rank}}_{min} = \frac{1+r}{2}$ and $\widetilde{\overline{Rank}}_{max} = \mu$, respectively, and get a normalized rank measure:

$$\widetilde{\widetilde{Rank}} = \frac{\widetilde{\overline{Rank}}}{\mu - \frac{1+r}{2}} \tag{5.5}$$

This value is very similar to the $\widetilde{Rank}$ from [178], where a complete retrieval of the database is assumed instead of retrieving only a small number of images and penalizing non–retrieved relevant items.

Precision* and *recall As already mentioned, these are standard measures in IR, which are known to give a good indication of system performance. Either value alone contains insufficient information. We can always make *recall* 1, simply by retrieving all images. Similarly, *precision* can be kept high by retrieving only a few images. Thus *precision* and *recall* should either be used together (*e.g. precision* at .5 *recall*, $P(R(0.5))$), or the number of images retrieved should be specified, (*e.g. recall* after 1000 images or *precision* after 20 images are retrieved, $R(1000)$, $P(20)$). *Precision* and *recall* are often averaged, but it is important to know the basis on which this is done. [105] use *precision* and *recall*. [9] use *Averaged precision*. [155] uses the *recognition rate* which is not defined in the text, but seems to correspond to the *precision* of a query.

Target testing The *target testing* approach differs significantly from other performance measures. Users are given a target image and the number of images which the user needs to examine before finding the target image is recorded. Starting with random images, the user marks images as either relevant or non–relevant. [49] employ this measure for the `PicHunter` system. [190] use a more elaborate version of target testing, in which the notion of moving (varying) targets is used to test the the ability of the system to track changes in user preferences during a query session.

Error rate [100] use this measure, which is common in object or face recognition. It is in fact a single *precision* value, so it is important to know where the value is measured (see above).

$$Error\ rate = \frac{Number\ of\ non\text{-}relevant\ images\ retrieved}{Total\ Number\ of\ images\ retrieved} = \frac{nrr}{N}. \tag{5.6}$$

Retrieval efficiency [183] defines *Retrieval efficiency* as in Eq. 5.7. If the number of images retrieved is lower than or equal to the number of relevant images, this value is the *precision*, otherwise it is the *recall* of a query. This definition can be misleading since it mixes two standard measures.

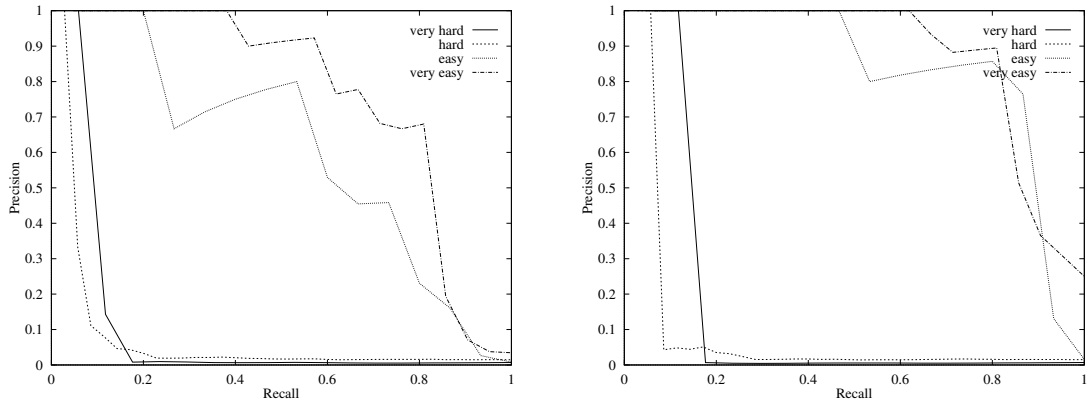
$$Retrieval\ efficiency = \begin{cases} \frac{Number\ of\ relevant\ images\ retrieved\ (rr)}{Total\ Number\ of\ images\ retrieved\ (N)} & if (rr + nrr) > r, \\ \frac{Number\ of\ relevant\ images\ retrieved\ (rr)}{Total\ Number\ of\ relevant\ images\ (r)} & otherwise. \end{cases} \tag{5.7}$$

Correct and incorrect detection [200] use these measures in an object recognition context. The numbers of correct and incorrect classifications are counted. When divided by the number of retrieved images, these measures are equivalent to *error rate* and *precision*.

Graphical representations

***Precision vs. recall* graphs** *PR graphs* are a standard evaluation method in IR and are increasingly used by the CBIR community ([264]). *PR graphs* contain much information, and their long use means that they can be read easily by many researchers. [93] use the representation with the axes changed (*i.e.* a *recall vs. precision* graph). For the sake of readability, this should be avoided. It is also common to present *partial PR graphs* (*e.g.* [93]). This can be useful in showing a region in more detail, but it can also be misleading since areas of poor performance can be omitted. Interpretation is also harder, since the scaling has to be watched carefully. A partial graph should therefore always be used in conjunction with the complete graph.

Figure 5.6 demonstrates that *PR graphs* can distinguish well between differing results. The drawback is that the PR graph depends on the number of relevant images for a given query. We can see that the plot for the "very hard" query starts later than the "hard" one and looks better, although the decrease of the curve is much faster. Practical information such as *precision* or *recall* after a given number of images have been retrieved cannot be obtained. In [98] a method to reduce



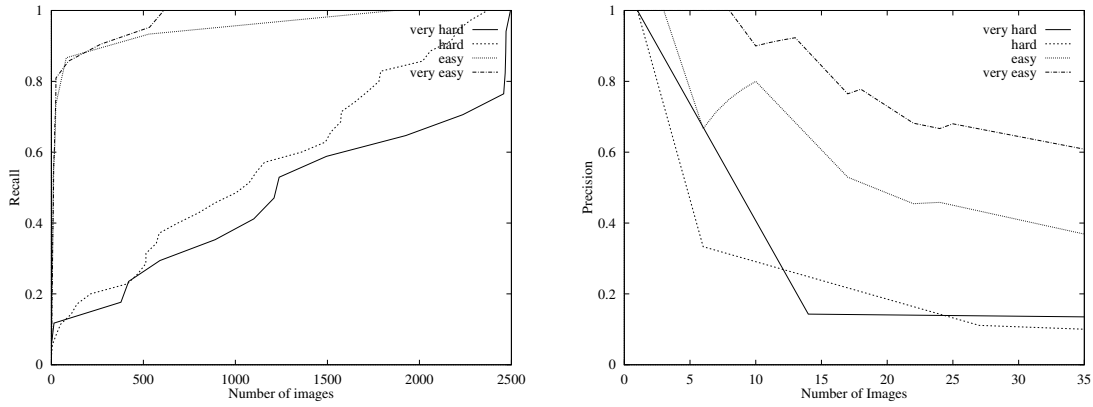
(a) without feedback (b) with feedback

Figure 5.6: *PR graph* for four different queries both without and with feedback.

the influence of collection size $N_{\mathcal{I}}$ and of the size of the relevance sets $N_{\mathcal{R}}$ on *PR graphs* is proposed using *generality* information. This corrects the graph by the theoretical difficulty of retrieval, by taking into account the performance of retrieval by chance.

Care has to be taken with averaging *PR graphs* where the single queries have a variable number of relevant items or where only a limited number of images is retrieved.

***Precision vs.* Number of images retrieved and *recall vs.* Number of images retrieved graphs** Taken separately, these graphs contain only some of the information of a PR graph. When combined, however, they contain more information and can easily be interpreted. The *recall* graph looks more positive than a PR graph, especially when a few relevant images are retrieved late ([213]). The *precision* graph is similar to a *PR graph*, but it gives a better indication of what might be a good number of images to retrieve. It is more sensitive, however, to the number of relevant images for a given query. If only part of the graph is shown it is hard to judge the performance ([5]).



(a) recall graph (b) precision graph

Figure 5.7: *Recall vs.* Number of images graph and partial *precision vs.* Number of images graph.

We can see in Figure 5.7 that the *recall graph* can distinguish well between the hard and easy queries, but not too well between the easy and very easy one. A complete *precision graph* is not easy to read in this case, as the part containing information will become very small. For this reason, we are printing a partial one. Here, we have the same problem with the different numbers of relevant images like in the *PR graph*. The result for the very hard query looks better than the result of the hard query.

Correctly retrieved vs. all retrieved graphs ([275]) contain the same information as *recall graphs*, but differently scaled. *Fraction correct vs. Number of images retrieved* graphs ([9]) are equivalent to *precision* graphs. *Average recognition rate vs. Number of images retrieved* graphs ([43]) show the average percentage of relevant images among the first N retrievals. The information displayed is equivalent to the *recall* graph.

***Retrieval accuracy vs. Noise* graphs** [97] use this measure to show the change in retrieval accuracy as noise is added. A noisy image is used as a query and the rank of the original image is observed. This model does not correspond well to many CBIR applications.

In [265] it was proposed to correct performance measures by the factor of retrieving relevant images by chance. Similarly, [98] states the importance of the influence of collection size $N_{\mathcal{I}}$ and the number of relevant images $r = N_{\mathcal{R}}$ on the retrieval results. This is very important when a comparison of measurements with differently sized databases is attempted. For now, we concentrate on the comparison of systems with the same database, and thus we can neglect this influence.

Proposing a set of standard measures

As described in [178], we propose to use a combination of measures based on the measures used in the TREC benchmark. This includes mainly measures that are very easy to calculate and to interpret. As these measures are proposed to compare systems based on the same image databases, a measure that takes into account the collection size and the number of relevant images, such as the *generality* is not proposed. If needed, the generality can be calculated from the number of items in the database and the collection size anyway.

- Number of items in the database ($N_{\mathcal{I}}$) and number of relevant items for each of the query tasks ($r = N_{\mathcal{R}}$).
- Time t it takes to execute a query.
- $Rank_1$, $\widetilde{Rank}$ and eventually $\overline{Rank}$: rank at which first relevant image is retrieved, normalized average rank of relevant images (see Equation 5.8), and the averaged rank.
- $P(20)$, $P(50)$ and $P(r)$: *precision* after 20, 50 and r images are retrieved, where r is the number of relevant images.
- $P(R(0.5))$ and $R(100)$: *recall* at *precision* 0.5 and after 100 images are retrieved.
- *PR graph.*

A simple average rank ($\overline{Rank}$) is difficult to interpret, since it depends on both the collection size $N_{\mathcal{I}}$ and the number of relevant images r for a given query. Consequently, we normalize by these numbers and propose to use the *normalized average rank*, $\widetilde{Rank}$:

$$\widetilde{Rank} = \frac{1}{N_{\mathcal{I}} r} \left(\sum_{i=1}^{r} Rank_i - \frac{r(r+1)}{2} \right) \tag{5.8}$$

where $Rank_i$ is the rank at which the ith relevant image is retrieved. This measure is 0 for perfect performance, and approaches 1 as performance worsens. For random retrieval the result would be 0.5. This rank is very close to the complement of the *normalized recall* R_n proposed by Salton in [226] that has also been used for image retrieval in [67].

These measures include the evaluation of effectiveness (accuracy) and efficiency (speed) and thus they should help to compare systems based on the same image database, using the same queries and the same groundtruth.

A larger number of measures for efficiency and about other system characteristics such as the size of the index files, the time to extract the characteristics of one image, the time to add an image to the database or to generate the entire index would also be useful. These measures would be particularly important for "real–world" use of CBIR.

Section 5.7 shows well that an evaluation based on different data such as varying query images $\mathcal{E}$ or varying ground truth data $\mathcal{R}$ cannot be easily compared even when using these performance measures. For a proper benchmark much more than performance measures are necessary.

5.2.7 Measuring the performance of relevance feedback

Relevance feedback is regarded as a major part of CBIR systems [221, 222, 264, 302] and it is also regarded to be very important for the evaluation of CBIRSs [137, 178]. We thus propose to include several, at least two, steps of relevance feedback into the evaluation process.

As it would be very time–consuming to use real users for giving the relevance feedback, we have to find ways to characterize what relevance feedback users would give. Based on the relevance judgments $\mathcal{R}$, we can safely assume that users would use all relevant images in the first $N = 20$ images of the result set $\mathcal{A}$ as positive relevance feedback. The use and the amount of negative relevance feedback normally depends on the experience that a user has with the system. Our experiments show the novice users barely use negative feedback, whereas expert users can significantly improve retrieval results with the use of selected negative relevance feedback [177].

Several possible strategies of generating relevance feedback based on the users relevance judgments and the system's query response are compared in [177]. This comparison quantifies the improvement brought by the use of negative relevance feedback and shows that it does even make sense to feed back non–marked images as negative relevance feedback to help novice users. Thus, we propose to negatively feed back all the non–relevant images in the first $N = 20$ images of the system response $\mathcal{A}$ for the evaluation of feedback, if possible. Systems that do not support negative feedback (such as [22]) can be compared based on positive feedback only.

5.3 A fully automated visual information retrieval benchmark

This section describes in details the techniques used in the automated benchmark [172], as well as ways to modify it to adapt it to other systems and databases, or to add additional performance measures.

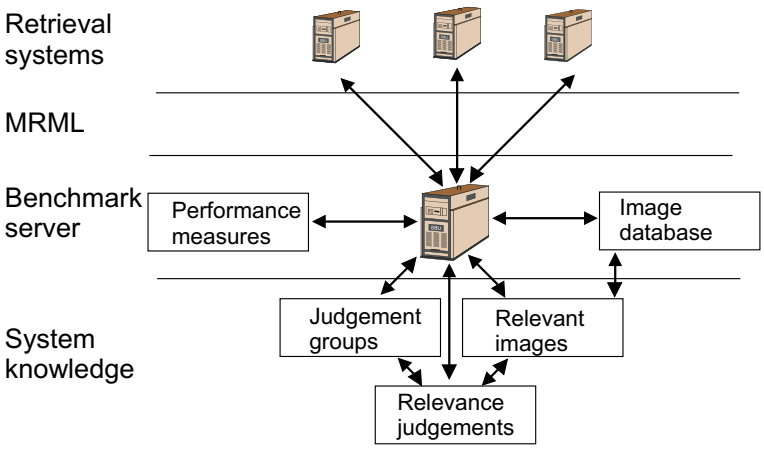


Figure 5.8: Structure of the automated benchmark.

Figure 5.8 shows the basic structure of the benchmark. *MRML* serves as the communication layer between the evaluated systems and the benchmark server. The image database and the performance measures are known to all the systems. The relevance judgments, however, should not be known since the responses can easily be optimized with this knowledge. Initially, the relevance judgments will also be made available.

The fact that TREC receives all the results offline would make "cheating" easy, because the systems can send in any results also hand selected results. This is assumed to not happen. The techniques are normally explained at the conference and in such a friendly forum to compare techniques, cheating would make no sense and it could discovered simply by reproducing a system using the same techniques as described.

5.3.1 Benchmarking software

The benchmark is implemented as a Perl object. The only parameters that need to be set to run the benchmark are the *host name* and *port number* of the query engine to be evaluated. For a new image database, the location of the query *relevance judgments*, *relevance groups* and *relevant images* must be set.

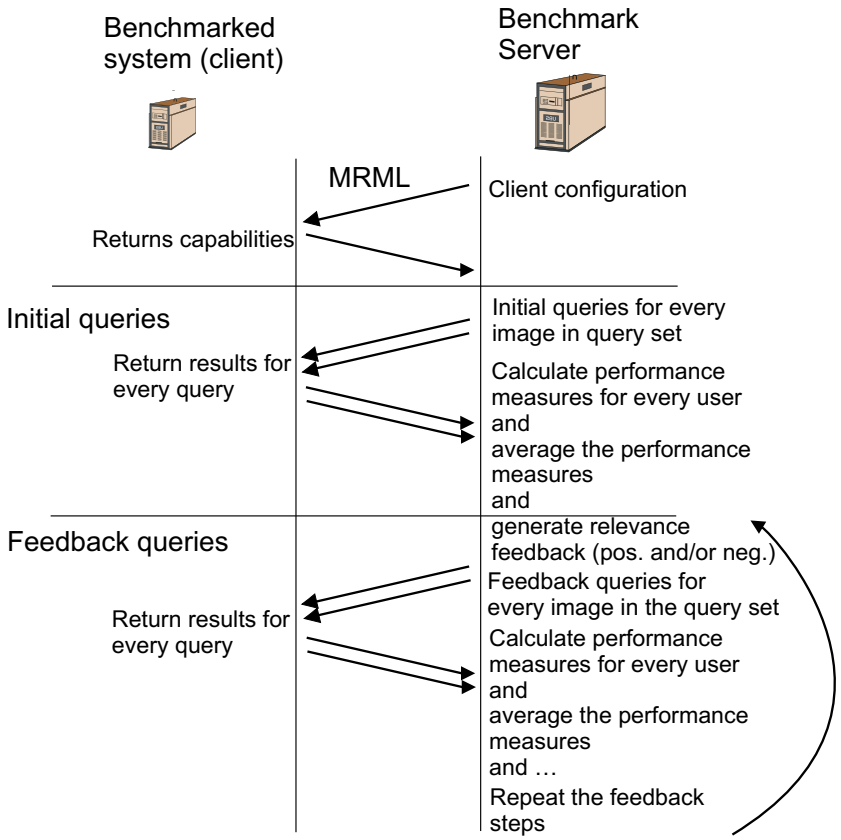


Figure 5.9: Communication flow for the automated benchmark.

Figure 5.9 shows the basic communication scenario when running the benchmark. First, the server to be evaluated must be configured for the benchmark scripts. Then the capabilities of the server to be evaluated are checked out via *MRML*. This checks whether the correct query paradigm is supported and whether the database selected for evaluation is available on the server.

Then, a query is executed for each image in the standard query set $\mathcal{E}$. The returned results are evaluated on the basis of the relevance judgments of each judge $\mathcal{R}$. Positive and negative relevance feedback is generated, based on these relevance judgments, separately for each judge. Using the generated relevance feedback, another series of queries is executed and the results are evaluated. This feedback step can be repeated several times to refine the query, in order to evaluate the effectiveness of relevance feedback for the evaluated system.

Reading the base data for the evaluation

Three inter–related data sets are required for the benchmark to run properly:

- The list of *query images* (or query tasks) for the first query step, $\mathcal{E}$;
- the list of *relevance judges* (one for each person l who made judgments, or only one set if a predefined database grouping is used);
- a relevance judgment file $\mathcal{R}_{il}$ for every image $\varepsilon_i \in \mathcal{E}$/relevance judge l combination containing a list of all images regarded as relevant for a certain query image.

These files are currently in plain text, but it is planned to use XML to be consistent with *MRML* as well as with the other structures in the *Viper* framework.

Generation of relevance feedback

Relevance feedback is generated from the relevance judgment files and the system response. We assume that the user would mark all relevant images positively and all non–relevant images negatively. We also need to assume the number of images the user would view. We choose 20 as a typical number of images displayed by a CBIRS. Thus, all images from the first 20 images of the system response which are in the relevant set for the query image are fed back positively and all those not in the relevant set are fed back negatively (as explained in [177]).

It is possible to change the number of images that are used for the generation of relevance feedback. This is very useful when it is for example known that there is a large number of relevant images, if we can assume that the user might be willing to look at a large number of images on screen. By default, the system remembers negative feedback over an entire query session, as non–relevant images are assumed to stay irrelevant, but this remembering can be turned off. Negative feedback can also be turned off if a system does not support it.

Evaluation

We perform an evaluation for each *query image/relevance judge* combination for the initial image and for each step of relevance feedback. The results are subsequently averaged over all queries (*query images* x *relevance judges*), with the aim of obtaining robust and meaningful results.

5.3.2 Configuring the benchmark for other image databases

It is very easy to configure the benchmark for a new image database. It is only necessary to create the *query image file*, the *relevance judges file*, and, for each query image/relevance judge combination, a *relevance judgment file*. In the simplest case, when the database is organized into groups, one image from each group is used as a query, the relevance judges file has only one entry, and the database grouping is used to construct the relevance judgment file.

5.3.3 An example evaluation

In this section the results of the automated benchmark are given, based on the database of the University of Washington. Table 5.2 shows the results for the initial query and four steps of relevance feedback for the *Viper* system. The benchmark configuration was constructed from the database organization (see Section 5.3.2). The results shown are averaged over the 13 query images. Characteristics of the image database can be found in Section 5.2.3.

Figure 5.10 shows the average *PR graphs* for the queries. This is the performance measure with the highest information content. The behavior of the system without relevance feedback and the strong improvements with relevance feedback can easily be seen in the fact that the curves "move up". The ideal performance would lead to $precision = 1$ and recall corresponding to an horizontal line in the upper part. The fourth relevance feedback step gives only a minor performance gain.

Measure	no RF	RF 1	RF 2	RF 3	RF 4
r	65.14	65.14	65.14	65.14	65.14
time t	1.23 s.	2.18 s.	2.49 s.	2.62 s.	2.70 s.
$Rank_1$	1.5	1	1	1	1
$R(P(.5))$	0.3798	0.5520	0.6718	0.6594	0.7049
$\overline{Rank}$	176.44	152.28	116.13	107.04	104.37
$\widetilde{Rank}$	0.1573	0.1308	0.0911	0.0811	0.0783
$P(20)$	0.5392	0.7357	0.8642	0.8892	0.9107
$P(50)$	0.4057	0.5271	0.6085	0.6328	0.6257
$P(r)$	0.3883	0.5256	0.6138	0.6640	0.6553
$R(100)$	0.4839	0.6070	0.6924	0.7279	0.7208

Table 5.2: Overview of the results for *Viper* when using the Washington database.

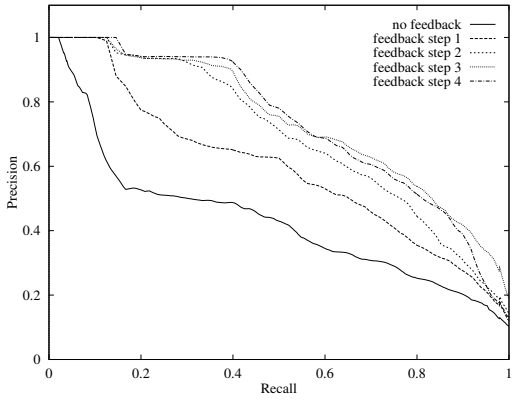


Figure 5.10: *PR graph* without and with relevance feedback using the Washington database.

The performance measures in Table 5.2 are meant to complement the *PR graph*. For a user, the *precision* of the images shown on screen is most important. We assume that the user looks at 20 to 50 images, so the *precision* at these points is very important. A significant improvement in each of the first three relevance feedback steps can be seen.

5.3.4 From images to multimedia

Although this benchmark has so far only been tested and used for images, in general, the techniques used can also be applied to retrieve and evaluate other data such as audio or more general multimedia. The extracted features might be different, but the technical infrastructure can essentially be used as all documents are accessed by their URLs and thus can contain several and varying media.

5.4 A web–based benchmarking system

A description of the web–based benchmark [173] is available from [29]. This page contains the prerequisites for the execution of the benchmark and links to a number of benchmark resources. The basic technologies are the same as for the benchmark described in Section 5.3. An example system using *MRML* can be downloaded at[30].

[29] `http://viper.unige.ch/evaluation/`
[30] `http://www.gnu.org/software/gift/`

5.4.1 Overview

Figure 5.8 shows the general structure of the benchmark. The communication between the benchmark server and the benchmarked systems is done in *MRML*. The benchmarked systems basically only need to know the URLs of the images in the database. The performance measures are openly visible as well.

For the web–based benchmark the image groups are for now not hidden. Any system could thus cheat and get good results. However, this web–based benchmark is so far more a research tool than an official benchmark, so there is no need to hide the ground truth data. For an official version this could be different.

5.4.2 Communication framework

We can see in Figure 5.11 that only the first step in the communication, the configuration of the benchmark and the last step of the communication are not done in *MRML*. This is the basic augmentation of the functionality compared with the communication framework shown in Figure 5.9. The configuration is done via a CGI interface, and the results are displayed on any web browser after the execution of the benchmark. Otherwise, the communication is the same as explained in Section 5.3.

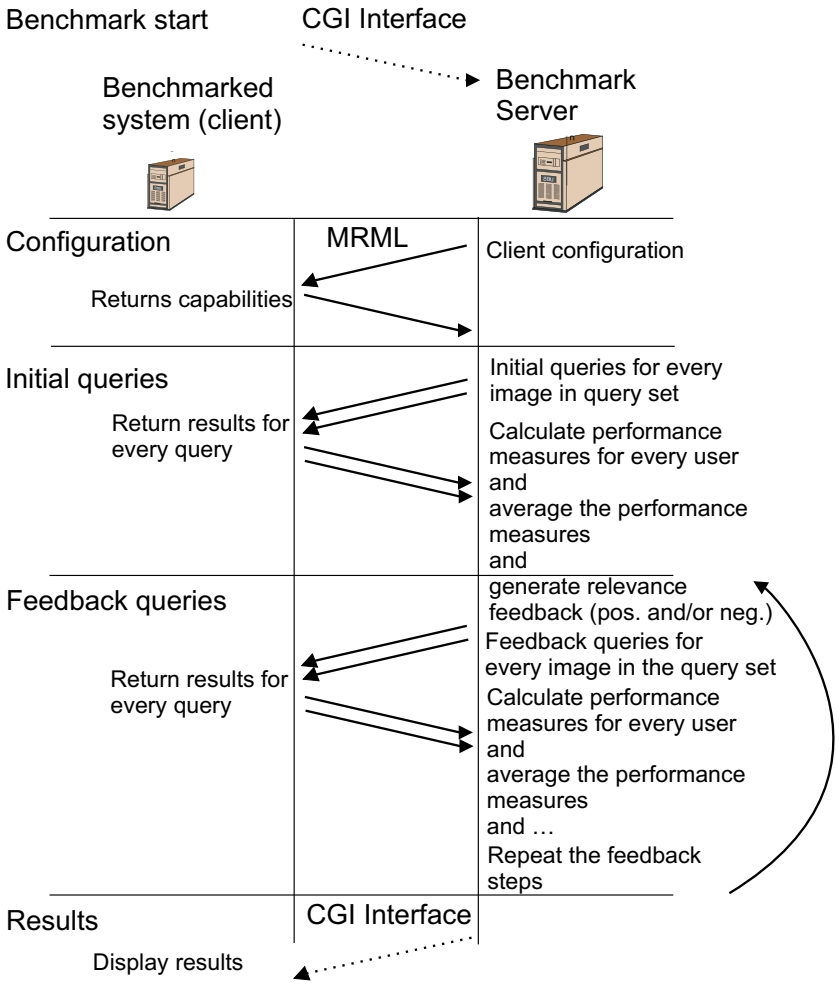


Figure 5.11: Steps of the automated benchmark for the communication.

With the first step of communication in *MRML* the benchmark server contacts the system to

benchmark and verifies that the chosen database is available and that the system speaks *MRML*. If this is the case, all queries are done as single image queries (QBE). After the first query step, all performance measures are calculated for the available relevance judgments and they are averaged over all the relevance judgments and queries. Now positive and negative relevance feedback is generated for every relevance judgment group (respectively every user) and every query. Based on this generated feedback, a new set of queries is executed and again the performance measures are calculated based on the relevance judgments, and the performance measures are averaged in the end. This step can be repeated as often as necessary. At the very end, all the averaged performance measures are displayed on the web browser.

5.4.3 Configuring the benchmark

The CGI Interface shown in Figure 5.12 allows the user to enter a number of parameters that the system needs to execute the benchmark.

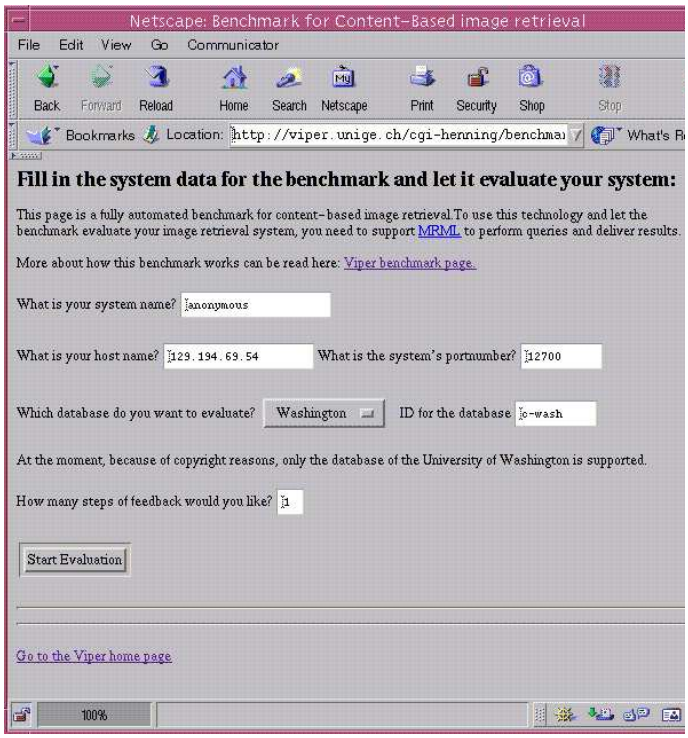


Figure 5.12: A screenshot of the web–based benchmark.

The *system name* is only meant as an identification of the benchmarked system to the server, it can be left at anonymous if the developers want their system to stay unknown. Important for the communication are the *host name* and the *port number* of the system to benchmark. These two parameters are absolutely needed to start the *MRML* communication on this socket. The choice of a *database* determines the queries and the relevance judgments the web–based benchmark will use. The *database ID* is important for the benchmark server to chose the database via *MRML*. For flexibility reason we allow to use any database ID and do not link it directly with a database name. The number of *feedback steps* finally determines the number of query steps that are done with the system. The first step is in this context the step with only one query image and no feedback.

5.4.4 Results

The results of the web–based benchmark are the same as with the previous, automated benchmark, only the presentation of the results differs significantly as they are shown directly on screen

in a web browser.

The big advantage is that this allows to make available image collections with groundtruth on the WWW and to have evaluations based on this data on the fly. Communications are kept light, thanks to *MRML*. It also allows the testing of the infrastructure for a possible benchmarking event, where system technologies and their results can be compared. This means that it can be used as a complement for a real benchmarking event.

5.5 The *Benchathlon*

The *Benchathlon* initiative basically started after discussion at SPIE Photonics West conference in 2000. The need for a real benchmarking event was identified. In 2001, a first session with the BIRDS–I benchmark [89] was held and a database containing about 3000 images was made available. 2002 saw several papers with propositions for a benchmarking architecture, examples queries [33], ways to annotate images [206], and ways to gain relevance judgments from annotations [115]. Still, no comparison of several systems was done as it was hard to get acceptance for a technical platform.

5.5.1 What can make a benchmark successful?

When we take a look at successful benchmarks like the TPC or the TREC, we should be able to learn from them to create a successful and accepted benchmark for the field of image retrieval. Both these benchmarks have a strong and independent governing body and are absolutely accepted in the field. The advantage that these two benchmarks have is that the area of research is already commercially successful and thus, there is a larger commercial interest. This might prove to be true for CBIR in the near future as well.

Both tests work with large, realistically–sized databases. For an image retrieval benchmark is will also be crucial to get these large image collections to do realistic evaluations.

It can also be seen that both tests did not start without any preparation. In text retrieval, many years of evaluation with standard collections were done before the TREC basically united the field to a real benchmarking event. The TPC was basically created from a benchmark and a data set that existed several years beforehand, but without a strong governing body. Results with these databases were often reported to be used for "benchmarketing" as no real standard procedures existed. The creation of a strong, independent body thus started to create the success for TPC.

For the *Benchathlon* to be successful we thus need several factors:

- Gather realistically–sized image test collections to do the evaluation on.
- Create a discussion in the field to gain acceptance for the method and performance evaluations used.
- Have a strong and independent governing body that is not using the benchmark for its own goals.
- Make it a regular event that people are getting used to, and where the participating groups see the utility of benchmarking for their research.

Although this might still take a few years to become accepted in the field, it is well worth to work for it. I think that it is important that such a benchmarking meeting becomes a friendly comparison event, where techniques are discussed and compared to each other, similar to the spirit of TREC.

5.6 What else can be benchmarked?

There are many different functions that can be tested with a benchmark and a proper benchmark in CBIR certainly needs to incorporate not only one, but several of these tests. All the

proposals for evaluation like [59, 137, 172, 249] deal only with the QBE paradigm, but [185] gives an example for evaluation on browsing and in [89] several methods are proposed to measure the efficiency of systems. It is important to have a combination of measures for efficiency and accuracy.

System developers can then decide to participate at one or several of the benchmarking fields, so specialized systems (*i.e.* for medical image retrieval) can be tested only in their specialized field.

For an evaluation of efficiency, it is important to state a description of the computer system used for the tests, as processor speed and the amount of memory can strongly influence the efficency results.

5.6.1 Looking for a specific image

This part is basically taken from [89]. Systems have to demonstrate their capability to extract features from a given input image and search for a corresponding image in the image database indexed beforehand. When looking for the exact same image basically the response time is important, whereas the search for an altered input image has to show that it is accurate as well.

- Search for an exact image from the database.
- Search with a cropped part of an image from the database.
- Search with a geometrically altered image from the database, such as rotated, scaled or shifted.
- Search with an image where a part is occluded.
- Search with a compressed image of the database, *i.e.* strong JPEG compression.

This can test the invariances of a retrieval system and especially the retrieval speed. Producing altered versions of images is very easy and the relevance set contains only the original image.

5.6.2 Looking for a number of similar images

The search for a number of similar images to a given query image is the standard QBE evaluation. Propositions for this are made in Section 5.2.

5.6.3 Looking for a sketch of an image

This test is very similar to QBE, but the information is in general incomplete as somebody drawing a sketch is normally concentrating on the object and not drawing the background, and normally the time you take to draw a sketch is limited. Otherwise the same measures for efficiency and accuracy can be used.

5.6.4 Target search (or called image browsing)

Image browsing was first proposed by [49] with the `PicHunter` system. The goal is to find a given image in the database and the performance is measured by counting the number of images that a user has to look at before finding the target. A benchmark for image browsers is presented in [185].

5.6.5 Practical application tests

This test models practical functions of a system that are routinely used. An index of an image database can be generated and the time for this is measured. Then, a number of images are added into the database and a query with this image is executed directly afterwards. Performance measures have to measure the efficiency of the system with respect to a given task.

- Feature extraction and index generation.

- Inserting an image into the database.
- Inserting an image into the database and find a known image similar to this one.

Other functions, depending on the application field, can be added to this part for completeness.

5.6.6 Measure the scalability of a CBIR system

For many applications, it is important that a CBIR system can deal with very large databases in an efficient manner. To show the scalability of a system, the time for several actions like feature extraction, index generation, and image querying can be measured for several collection sizes, for example with 10,000, 100,000 and 1,000,000 images. This gives means to interpolate the response time for even larger image databases.

Care needs to be taken with respect to the hardware included in forming the system, for example to determine if operations can or cannot be completely cached.

5.6.7 Tests for special application areas

Images from different application fields have different characteristics and specialized programs should be tested accordingly. *Medical images* are, for example, often in black and white and also *satellite images* might need color characteristics different from the ones used for *stock photography*. Many of them can be tested with the same performance measures, but fields such as *trademark retrieval* will need different performance measures because for trademark retrieval *recall* is the essential point and trademark researchers are normally used to looking at a large number of images.

The process of getting relevance judgments can be different because in certain fields, real ground truth is available and not as subjective as for photographs.

5.6.8 Evaluation of CBIR interfaces

It might not be possible to measure CBIR interface completely automatic, but users can for sure determine how well the information is presented and how easy it is to give feedback or find groups of similar images. Measures for the quality of interfaces have to be developed. The developments made with three–dimensional presentation of query results such as [191] show that interfaces for CBIRSs can be studied much more than this is the case at the moment.

Experiments like the ones performed in [269] for text retrieval might help to develop methods for evaluating user interfaces for interactive systems also in CBIR.

5.7 The truth about Corel

The Corel Photo CDs have been used in many publications to demonstrate the performance of content–based image retrieval systems (CBIRSs) and has become a de facto standard in the field [25, 29, 112, 136, 198, 211, 277, 278, 291, 309]. Sometimes, the Corel images are mixed with other images [82]. Unfortunately, Corel[31] does not only offer one set of images but a collection of more than 800 Photo CDs containing 100 images each of a certain theme but also differently compiled collections of smaller–sized images with sets of several thousand images grouped into same–subject groups (*i.e.* used in [309]). Even people of the same research lab presenting at the same conference often use completely different sets of images from the Corel collection to evaluate their systems [112, 140, 211]. This makes it very hard to compare the performance of systems as different groups of images can be either very easy or hard to separate from one another, depending on the images they contain. When having a large number of image sets, it makes sense to compile a collection of images with groups where each group contains a distinct feature (sunsets, planes on a blue sky, divers, deserts images, ...), so they can be separated easily and the performance of a system looks better .

[31] `http://www.corel.com/`

Once a collection is compiled, a decision has to be taken as to which images to chose as query images. Here, it is important to chose images that represent a group well and not images that differ from all other images within the group. Sometimes, the first image is taken [172] but mostly nothing is said about the process of selecting a query image [198, 278]. Sometimes, it is selected by hand [62, 136] or randomly [112]. Both these options leave much room for choosing the images as query images that perform best for a given system, which, as we will show, can improve the performance enormously (Section 5.7.3). Often, more than one image of a class is chosen as query image [62, 136].

Although the Corel sets contain images of the same subject, many groups contain a few images that are visually dissimilar. It may therefore be sensible to keep these images out of the relevance sets for a query as they will make the performance look bad, although it was only a few visually dissimilar pictures that were not found. Thus, often a subset of the images is chosen to be in the relevance group [62, 278]. Sometimes the entire grouping of Corel is discarded and a new grouping is done [309] which basically means that the developers can decide upon the performance of the system under evaluation. [198] does a new grouping based on the annotation delivered with the Corel Photo CDs. This can also be done by choosing easy–to–separate groups. Basically all researchers use a small subset of the images. Choosing the easiest subsets can drastically improve the apparent performance as shown in Sections 5.7.4 and 5.7.5. In [291], the system developers judge the results themselves after every query step, which makes all results irreproducible.

One of the problems with using the groupings of the Corel collection as relevance judgments (RJs) is that the grouping does not always comply with what a human might chose visually as a good result for this group. What we really would like to evaluate from a CBIRS is how well its response fits with what a human would expect to be returned as a result. Humans are subjective and have different opinions [265]. Thus, it might be sensible to use real user judgments as a basis for evaluation, as done in [264].

The problem induced by the fact that retrieving images from small sets is easier than from large collections has already been mentioned in [265] where a correction for discounting the chance factor was proposed. Similarly, the use of *generality* in [98] leads to appropriate measures to compare the retrieval quality when the database and relevance sets have a varying size. The *normalized averaged rank* proposed in [178] shows to be effective for comparing the performance when removing unused images (Section 5.7.5).

By this section we will demonstrate, using our system *Viper*, that apparent performance can be changed massively by choosing the test setup.

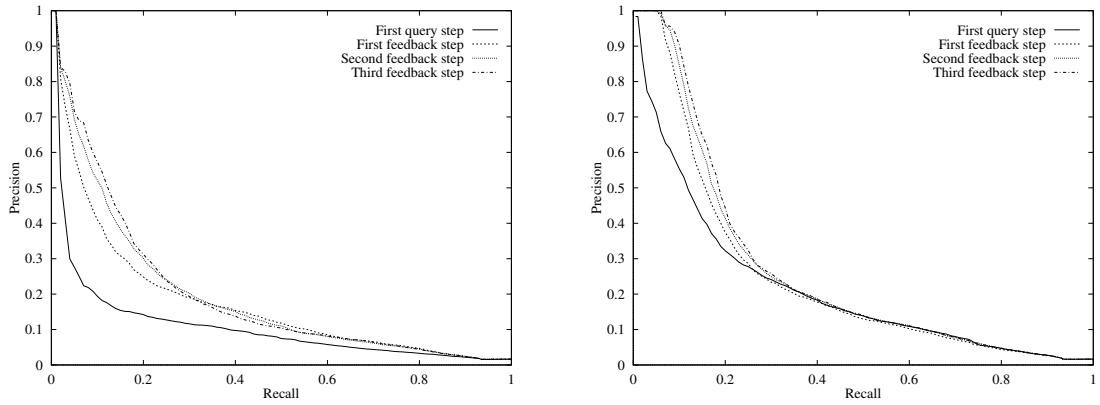
Unfortunately, in our research group, we only have access to a small subset of images from the Corel Photo CDs, containing 61 different groups of 100 images each. They do not contain the groups most often used by other systems and also relatively easy to query such as "sunsets", "tigers", "eagles on the sky". We are thus limited with the upper performance we can reach, but in the following sections it will become clear that even with a small subset, the apparent performance can be boosted.

5.7.1 Basic performance

Our initial evaluation procedure using databases with fixed grouping consists in using the first image of a group as a query image in $\mathcal{E}$ and the entire set as relevance judgment for the query $\mathcal{R}$. The entire database of 6100 images formed $\mathcal{I}$.

Figure 5.13(a) and Table 5.3 show that the results do not look very good, although the use of relevance feedback does improve sensibly the *precision* in the first $N = 20$ images, $P(20)$. Having an average of 5 relevant images in the first 20 returned does not sound too bad. However, we can see that $Rank_1$ is well above 1, which means that there are image sets where no relevant image was retrieved in the top 20.

When taking a look at the sets with the worst results we can see that sometimes the first image is visually very different from other images in the same group. An example is a query for images of "Paris" where the first image of the group was the Eiffel tower at night, which perfectly helped retrieving another image of the Eiffel tower and other buildings at night, with none of the buildings



(a) first image as query

(b) optimized query image

Figure 5.13: *PR graph* using the first image as query image and an optimized query image.

	1st step	RF 1	RF 2	RF 3
r	100	100	100	100
time t	8.53	11.69	12.62	13.19
$Rank_1$	28.79	21.13	24.30	23.56
$R(P(.5))$	0.0552	0.1336	0.1487	0.1525
$\overline{Rank}$	1713.4	1602.8	1561.0	1543.5
$\widetilde{Rank}$	0.2535	0.2366	0.2301	0.2276
$P(20)$	0.2508	0.4082	0.4795	0.5164
$P(50)$	0.1711	0.2603	0.2846	0.2895
$P(r)$	0.1267	0.1821	0.1857	0.19
$R(100)$	0.1267	0.1821	0.1857	0.19

Table 5.3: Performance using the first image of a group as query image.

being from "Paris". Therefore, it sounds logical to chose a query image that better represents the class.

5.7.2 Optimizing the query image

For not having to chose a "best" query image by hand, we wanted to find the image that best represents the class from our system's point of view. We thus evaluated a query for every image of every class and choose the image with the highest $P(20)$. If there were several images with the same $P(20)$, then $P(50)$ gave the decision. Thus, all we modified was the set of query images for the evaluation $\mathcal{E}$, the relevance sets $\mathcal{R}_i$ and the image set $\mathcal{I}$ stayed the same.

Figure 5.13(b) and Table 5.4 illustrate how strongly the performance of our system improved, although the same system *Viper* with the same relevance sets $\mathcal{R}_i$ and the same database $\mathcal{I}$ was used. For every group, we have a relevant image in the top 20 after one step of feedback, we even have an average of 11 relevant images in the top 20 and after three steps of relevance feedback even 15 relevant images in the top 20.

Comparing the two data sets in Table 5.3, we can see clearly that all the performance measures improve strongly. $P(20)$ more than doubles and the rank–based measures improve substantially.

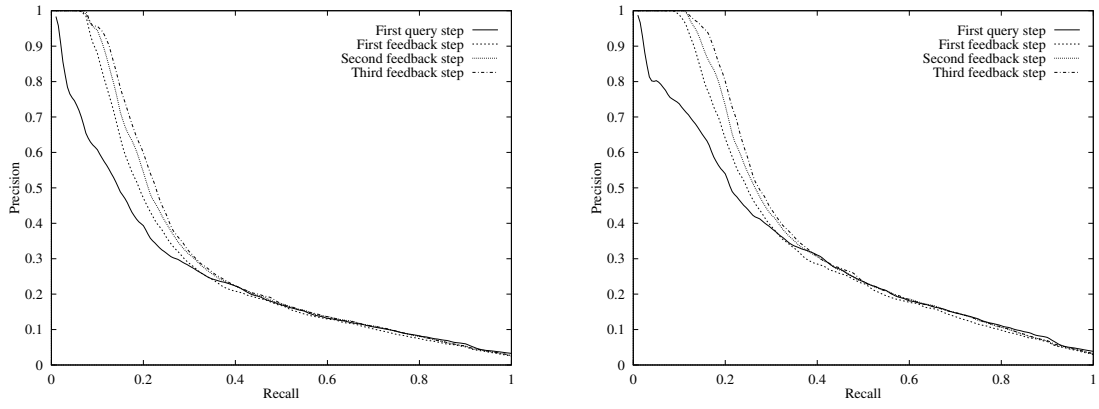
5.7.3 Removing bad images

We said earlier that there are images in each group that are visually dissimilar from several images in the same group. Although these images are not used as query images anymore, they still worsen the measured quality of retrieval. We do not want to cut too many images and set

	1st step	RF 1	RF 2	RF 3
r	100	100	100	100
time t	8.35	11.60	13.08	12.89
$Rank_1$	1.49	1	1	1
$\overline{R(P(.5))}$	0.1590	0.1904	0.2074	.2136
$\overline{Rank}$	1223.9	1205.7	1185.9	1176.1
$\widetilde{Rank}$	0.1787	0.1759	0.1729	0.1714
$P(20)$	0.5508	0.6352	0.6959	.7344
$P(50)$	0.3531	0.3750	0.3892	.4006
$P(r)$	0.2452	0.2449	0.2566	0.2549
$R(100)$	0.2452	0.2449	0.2566	0.2549

Table 5.4: Performance using an optimized query image.

an upper limit of 20 images to be removed from each set $\mathcal{R}_i$ of the database. We assume that an image is really bad for the query and can never be retrieved in one of the top ranks if the image is ranked below one third of the entire database (we used $Rank > 2000$). The query set of images $\mathcal{E}$ and the image set $\mathcal{I}$ used are not changed.



(a) no bad images

(b) no bad groups

Figure 5.14: *PR graph* when removing bad images from the relevance judgments and removing bad groups.

As changing this does not at all change the ranking of the images and neither the *precision* after the first images are retrieved, we can only hope for an improvement in the measures based on *recall*. Figure 5.14(a) shows that the entire *PR graph* looks improved as the curve is slightly lifted up. This makes the performance look quite a bit better although the *precision* measures in Table 5.5 show that the performance in the first 2000 result images did not change at all. As expected, the average rank measures and the $R(100)$ do improve significantly as images retrieved late are left out from the evaluation.

5.7.4 Removing bad classes

The easiest way of improving the performance is of course to only have a set of easily distinguishable groups of images. With the Corel Photo CDs containing more than 80,000 images it is easy to choose a small subset of groups that work well. Unfortunately, we do not have access to many of the image groups commonly mentioned in citations, but we can already see that removing the 21 worst relevance sets from the evaluation (leaving us with 40 query images in $\mathcal{E}$) still leads to a massive improvement in performance as can be seen in Figure 5.14(b) and Table 5.6.

Thus, we changed the size of $\mathcal{E}$ and consequently as well the number of relevance sets $\mathcal{R}_i$ for each of the query images ε_i. The image database $\mathcal{I}$ is not changed. And this with still using the

	1st step	RF 1	RF 2	RF 3
r	82.41	82.41	82.41	82.41
time t	8.39	11.56	12.33	12.77
$Rank_1$	1.49	1	1	1
$R(P(.5))$	0.1871	0.2268	0.2465	0.2544
$\overline{Rank}$	850.3	883.3	860.0	850.39
$\widetilde{Rank}$	0.1325	0.1379	0.1341	0.1325
$P(20)$	0.5508	0.6352	0.6959	0.7344
$P(50)$	0.3531	0.3751	0.3891	0.400
$P(r)$	0.2717	0.2729	0.2829	0.2882
$R(100)$	0.2931	0.2933	0.3077	0.3055

Table 5.5: Performance when removing bad images from the relevance judgments.

	1st step	RF 1	RF 2	RF 3
r	83.68	83.68	83.68	83.68
time t	8.66	11.90	12.43	12.46
$Rank_1$	1.43	1	1	1
$R(P(.5))$	0.2519	0.2839	0.3058	0.3124
$\overline{Rank}$	600.5	651.3	628.2	616.2
$\widetilde{Rank}$	0.0915	0.0997	0.0960	0.0940
$P(20)$	0.6463	0.7412	0.8088	0.8463
$P(50)$	0.4365	0.4495	0.4685	0.4845
$P(r)$	0.3368	0.3290	0.3395	0.3456
$R(100)$	0.3609	0.3532	0.3648	0.3650

Table 5.6: Performance when removing bad groups from the relevance judgments.

exact same system, exact same database with the exactly same measures for evaluation. We now have average of 13 relevant images in the top 20 ($P(20) = 0.65$) and 22 in the top 50 ($P(50) = 0.44$) in the first query step. This is 13 images more than what we had with our first evaluation. All performance values basically doubled.

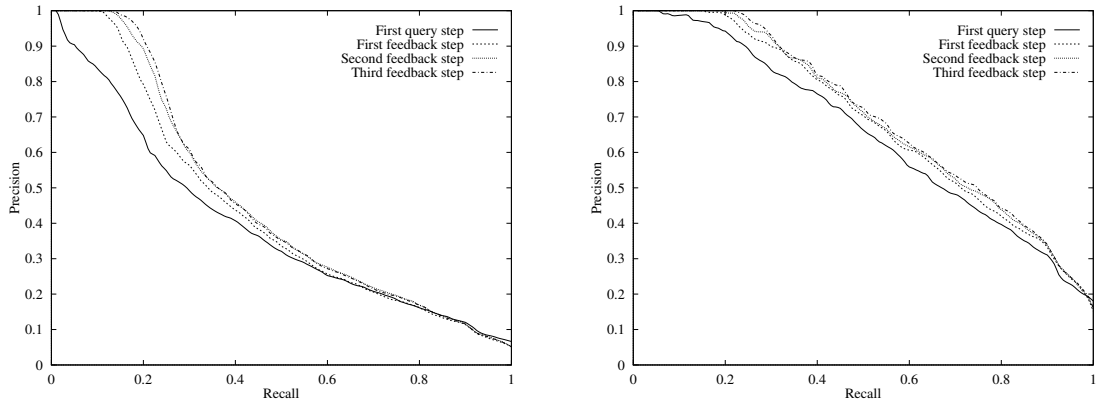
5.7.5 Creating a new, smaller inverted file

Until now, we always used the exact same database $\mathcal{I}$ for the queries. Just like we remove image sets or single images from the evaluation process (from $\mathcal{E}$ and $\mathcal{R}_i$), it makes sense to remove all these unused images completely from the database. Like this, we create a new image database $\mathcal{I}$ where every image is part of exactly one relevance group. This database now contains 40 image groups (with one query image each) and a total of 3500 images. $\mathcal{E}$ and $\mathcal{R}$ are not changed anymore.

We can see clearly in Figure 5.15(a) and Table 5.7 that this again improves the performance strongly, especially the *average rank* drops to roughly half the value. The query time is strongly reduced and all *precision* and *recall* values go up. Interestingly, the *normalized average rank* stays almost exactly the same as before, which shows that the actual performance of the system did not improve and that it simply looks better because of the smaller database size. As the *normalized average rank* is normalized by the database size, it characterizes well the ranks of relevant images.

5.7.6 A "perfect" performance

Now, we use the same technique as before but we generate a new, even smaller inverted file. This time, we do not take the best 40 groups, but only the best 15 groups, with a total of 1285 images. The new size of $\mathcal{I}$ corresponds well to common sizes used for evaluation of CBIRSs at the



(a) 40 groups (b) 15 groups

Figure 5.15: *PR graph* when creating an inverted file with only the good images and groups for 40 groups and 15 groups.

	1st step	RF 1	RF 2	RF 3
r	83.68	83.68	83.68	83.68
time t	4.71	5.80	6.51	6.52
$Rank_1$	1.18	1	1	1
$R(P(0.5))$	0.3228	0.3654	0.3863	0.3923
$\overline{Rank}$	350.3	364.3	347.88	351.89
$\widetilde{Rank}$	0.0907	0.0948	0.0900	0.0911
$P(20)$	0.7212	0.8250	0.8813	0.9163
$P(50)$	0.504	0.5345	0.5585	0.5650
$P(r)$	0.3848	0.4013	0.4094	0.4092
$R(100)$	0.4162	0.4268	0.4387	0.4391

Table 5.7: Performance when creating an inverted file with only the easy images and easy groups for 40 groups.

moment [139, 277]. As before, we downsized $\mathcal{E}$ and the $\mathcal{R}_i$, and then we created a smaller image set $\mathcal{I}$ to do the evaluation on.

In Figure 5.15(b) and Table 5.8, we can see that this time the improvement is really extreme and the performance can almost be called "perfect". Still, we have to remember, that we use the **same system** *Viper* with the **same features** $\mathcal{F}$ as before, we only made the query task to solve much easier.

From the first query step on, we have an average (!) $P(20)$ that is above 0.9, up from a meager 0.25 in the first evaluation (see Table 5.3). After 3 steps of relevance feedback we have a $P(20)$ of basically 1. The same applies for the other measures. The $R(100)$ shows that only an average of 20 relevant images per query were not found within the first 100 results.

We can also see that relevance feedback improves the results only slightly. This can be explained with the fact, that almost no negative feedback is given to better explore the feature space. The feedback is generated only from the first 20 images returned and most of them are already correctly retrieved items. If we chose a larger number of images for the generation of relevance feedback, the results of the subsequent relevance feedback steps will most likely become much better, as already reported in [177].

5.7.7 Extensive learning of feature weights

We might think that the "perfect" performance that a system can reach has already been achieved with the results shown above. We might err as we can take into account feedback infor-

	1st step	RF 1	RF 2	RF 3
r	85.67	85.67	85.67	85.67
time t	2.86	2.41	2.51	2.38
$Rank_1$	1	1	1	1
$R(P(0.5))$	0.635	0.678	0.6943	0.702
$\overline{Rank}$	119.1	115.5	113.1	110.66
$\widetilde{Rank}$	0.0546	0.0521	0.0503	0.0486
$P(20)$	0.9167	0.9767	0.9867	0.9967
$P(50)$	0.752	0.7707	0.7813	0.7906
$P(r)$	0.5824	0.6041	0.6065	0.6085
$R(100)$	0.6232	0.6464	0.6517	0.6580

Table 5.8: Performance when creating an inverted file with only the easy images and easy groups for 15 groups.

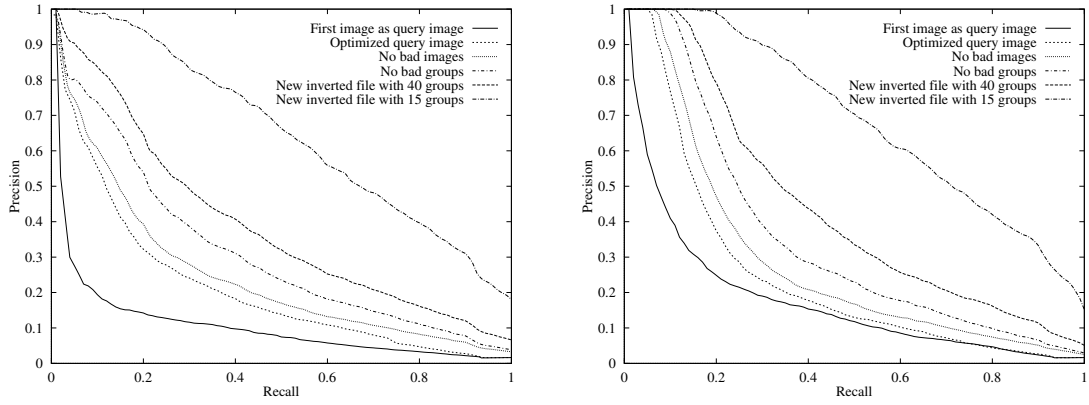
mation from user logfiles as explained in Section 4.6.1.

To prove the validity of our learning algorithm, we use the method explained in [112] to learn the importance of regions in an image for the generation of our usage log files. We make a query for every image in the database, take the top 100 resulting images and check them for relevance or non–relevance, respectively. From these 100 images we create relevant–relevant and relevant–non–relevant images pairs and use them to calculate the additional weighting factors.

We can see that this can still reach an improvement in the results. The performance improvement when using the same algorithm for a collection that was not optimized for our system can even be better than the improvement we showed here. Still, these are artificial conditions and no real–world logfiles would have such a good information to learn from.

5.7.8 Comparison

In Figure 5.16 the *PR graphs* of the different methods presented can be compared. This shows, how strong the gain in apparent performance really is.



(a) no relevance feedback

(b) with relevance feedback

Figure 5.16: Comparison of the different evaluation methods for the Corel database.

All the methods used can also be called "cheating" as the system, the techniques and features used and the performance measures stay exactly the same. We only make the retrieval task easier for our system, we basically optimize the query task for the use with our system.

All this highlights the importance of proper performance evaluation and especially of a commonly accepted and available image collection $\mathcal{I}$ with query tasks $\mathcal{E}$ and relevance judgments $\mathcal{R}_i$ for these tasks. Without such a set ($\mathcal{I}$,$\mathcal{E}$,$\mathcal{R}$), it is nearly impossible to compare any two CBIR systems, even when they use the "same" image collection such as the Corel Photo CDs.

$P(50)$	group number	content
0.92	211000	oil paintings
0.66	153000	swimming Canada
0.9	156000	divers and diving
0.6	524000	works of art: engravings
0.9	99000	religious stained glass
0.54	161000	land of the pyramids
0.72	373000	everyday objects
0.54	502000	works of art: landscapes
0.7	125000	coins and currency
0.54	96000	candy backgrounds

Table 5.9: $P(50)$ for the image groups that are easiest to query.

To have an idea about the performance differences of single images sets, Table 5.9 lists the images groups with the highest $P(50)$, characterizing an easy group. We can see how much easier the very easy image sets are. By having the entire set of more than 800 groups it should be possible to get a subset with even higher $P(50)$.

Based on this study, it would be interesting to create a difficulty measure for every Corel image set, but unfortunately the difficulty of a group depends on both the other groups present in an image database and the CBIRS used. Not only the intra–group similarity but also inter–group similarities are important to measure the difficulty of an image collection.

5.8 Summary

This chapter describes the evaluation of retrieval systems. Starting with the evaluation of text retrieval systems and especially with TREC, analogies with the CBIR field are shown. A review of techniques and measures currently used for the evaluation of CBIRSs is followed by propositions for evaluation measures to be used in image retrieval based on the measures of TREC.

An automatic benchmark based on the proposed measures and the communication protocol *MRML* is constructed and results are shown. The same infrastructure is also used to make the benchmark web–accessible.

The *Benchathlon* evaluation initiative is proposed as a possible governing body for a benchmark for CBIR. Several fields of evaluation that will be interesting for such an evaluation forum are proposed.

To underline the importance of such an independent benchmark with a standardized image database $\mathcal{I}$, including query tasks $\mathcal{E}$ and relevance judgments $\mathcal{R}$, an evaluation using the Corel Photo CDs is done. This shows that based on the "same" Corel collection the same CBIRS can score completely different results, depending on the images chosen for the database, as query images and as relevance judgments. This basic test shows that so far, it is impossible to compare any two systems that are not assessed on the same image set and based on the same query images and relevance judgments.

Chapter 6

Real World Applications : Trademark and Medical Image Retrieval

> If I have some six photos, I may put them side by side on the table and leave them there for the rest of the evening [...] then it just turns out that this is the one. In a way they drop out automatically.
>
> A journalist, taken from [151].

The above statement shows that we can learn much from working with real users in real applications. Search behavior cannot be discovered in a purely theoretical research environment. Applications in real working environments are needed to pinpoint problems and to be able to create a retrieval structure that is adapted to the users' needs.

No complete evaluation of *Viper* was done for any of the two application fields described in this Chapter. Furthermore, no adaptation of the *Viper* system was possible within the timeframe of this thesis. However, the user test at the Institute for Intellectual Property (IIP) in Bern described in Section 6.1 showed us how much domain knowledge is needed to make a properly working application based on a research retrieval system.

The importance of real–world applications cannot be underestimated. It is essential not to simply use images from a certain domain and then evaluate some interesting algorithms under laboratory conditions, but to really make user tests and let users evaluate the system based on practical cases.

The field of content–based image retrieval is growing up and has an enormous potential, but is it important to show what seriously works and what does not work. Many application areas have been identified for the use of content–based retrieval algorithms, from the management of private *photograph* collections to image collections used by *journalists* for their information needs [151, 294]. Other images indexed commonly are *satellite images* [5, 35, 117] or images of *GIS* (Geographical Information Systems) [68]. CBIRSs have also been applied for *face recognition* (*e.g.* forensic) [155, 203], to detect *cars* [200] or *buildings* [105], for *character recognition* [52] and on *museum collections* [240].

More and more retrieval systems are also used in the *medical domain* [13, 44, 117, 195, 271] for the most diverse tasks from the detection of tumor shapes [124], melanomas [134] to texture analysis of pulmonary images [21, 141, 244, 245] or simply for a classification of the anatomical structure on images [78].

One of the most active areas within the field of image retrieval has definitely been that of *trademark retrieval* [30, 39, 67, 110, 116, 129, 303]. Trademark management fits well with the QBE paradigm as, normally, there is a query image that needs to be compared with other images which are already registered as marks. Thus, there is no *page–zero problem* to find a starting point for a query. Trademarks are also relatively simply structured and their information is mostly contained

in the shape. In practice, colors do not play an important role because it limits the registration to a single color. Black and white trademarks in contrast reserve the form in all colors. Unfortunately, most retrieval systems for trademarks have not been developed in cooperation with trademark offices but simply use the images that are available (for example from the MPEG–7 datasets) to show invariance characteristics of shape features [34, 39, 46, 110, 118, 254, 284]. Only a few research groups really cooperate with a patent office and actually evaluate their system with real–world query results [65, 67, 84]. Worse, the search for trademarks in patent offices is almost completely semantics–based (or more exactly: object–based) so that most of the low–level features are not expected to work well. Rather, a combination of semantic (textual) and visual characteristics will lead to success and it cannot be expected that visual characteristics will completely replace the use of text for the trademark query process as was predicted in [116, 132, 303]. Rather, a combination of text and visual characteristics will lead to success.

6.1 Trademarks

Trademark retrieval is one of the most common fields used to demonstrate CBIRSs. Besides the use in trademark offices to identify conflicts between existing marks and marks that are to be registered, there is also an application for automatic mail sorting in companies [60, 111, 268]. Based on logos, letters can be distributed directly to the responsible persons. Another application is the recognition of logos on drug tablets [75]. A demo for the recognition of ancient watermarks can be found at [1].

6.1.1 History of trademarks

Trademarks existed in one or the other form since a much longer time than one might think. This chapter gives a little timeline about the history of trademarks. More about trademark history can be found at [2].

The first symbols thought of to be marks were on cave drawings created in *5000 b.c.*. Several symbols of the flanks of animals are though to be ownership marks. In Mesopotamia, marks started as early as *3500 b.c.* with marks on commodities with cylindrical seals. Only little later, bricks, pottery and quarry stones were marked in Egypt *3000 b.c.*. In Corinth *2000 b.c.* and Greece *600–300 b.c.*, marks were used on ceramics and pottery. Extensive use of trademarks was made by the Romans between *500 b.c.* and *500 a.d.* where bricks were marked with the maker (brand), date of production and place of production. The maker of bricks could then be held responsible for eventual failure of the bricks while building. Not much is known about the use of marks between the end of the Roman empire and the beginning of the renaissance.

In the *12th century* trade guilds started to use symbols for their members and in the *13th century* the first *watermarks* were used for paper in Italy. Marks were use for cutlers to maintain their monopoly and for traders to identify pirated goods. The year *1618* marked the first reference to an infringement, where a clothier was using the mark of another clothier manufacturing higher quality clothes. Trademark laws were enacted in France (*1857*) and England (*1862*) and Bass was the first mark to be registered in England in *1876*. In *1883*, a first international agreement was reached at the Paris convention for the protection of industrial property. In *1887*, Coca–Cola used its first trademark for a tonic beverage. In *1964*, Mc Donald's reserved its logo as 3D mark for buildings and arches. Now, not only 2D and 3D marks can be registered, but also sounds, groups of sounds, colors (*e.g.* violet for the Milka brand of chocolate) and even scents.

Still, by now, most of the registered marks are flat 2D images. Owens Corning reserved the color pink *1987* and in *1991*, a scent for a sewing thread was registered. In *1997*, the trademark office considered to register domain names. American trademarks can all be viewed and searched on the Internet publicly[3].

[1] `http://vision.unige.ch/demos.html`
[2] `http://www-old.lib.utexas.edu/Libs/ENG/timeline/tmindex.html`
[3] `http://www.uspto.gov/`

In the US, around 1 million marks are registered, in Switzerland there are around 300.000 registered trademarks.

6.1.2 Workflow for trademark search and goals of the retrieval process

When a new trademark is registered at a trademark office, the owner of the mark usually wants to make sure that there are no conflicts with other, already registered marks, for example, because the two marks are too similar in appearance. In Switzerland, the responsibility to avoid conflicts is left to the registering party, whereas at the European trademark office[4], all marks that are newly registered are automatically checked for conflicts against other already existing marks.

The service to check the database of existing marks for possible conflicts is a service offered by the trademark office (IIP, Institute for Intellectual Property) for a fee.

An example of a query image (submitted trademark) is shown in Figure 6.1. This trademark is in the Nice classes 35, 36, 37, 42 (for Nice classes see Section 6.1.3 below). The two parts of the trademark, visual and textual, are highlighted with red boxes.

Figure 6.1: An example for a trademark that has been used for retrieval at the IIP.

When such a searching service is asked for, the image for the new mark normally arrives either by mail or by fax, which can significantly reduce the quality of the image, especially for automatic image analysis. Then, the image is encoded with the Vienna code, the Nice groups for the product or service are selected, and the image is scanned into the computer for further digital treatment such as generating a report with the mark. A query with the Vienna codes and the Nice groups is done to the query system *Ascepto* that manages the entire trademark database, a system supplied by the company *SWORD*[5]. The query system then returns all the images that are in at least one of the Nice groups for the new image and that have at least one of the Vienna codes in common with the query mark. This list of answer images can have a largely varying size, from 100 result images up to 10,000 result images.

All these returned images are now analyzed manually with around 10 images shown on screen at the same time. This process of observing all the marks can take more than a day. All the possibly conflicting marks are extracted manually and, eventually, a document is prepared containing conflicting marks and other very similar marks. All trademarks that have been entered into the database but that are not yet associated with their Vienna coding are also returned to the observing image researcher. It is extremely important to have 100% recall for trademark retrieval because a single missed image that might create a conflict can have serious legal consequences.

An example for a subset of the trademarks returned as response to a query (using Vienna Code and manual inspection) with the trademark in Figure 6.1 can be seen in Figure 6.2. In the trademarks, the important graphical elements are highlighted in red to underline why these images were chosen. The result demonstrates well that global visual similarity features or fixed local features can not be expected to deliver good results. For a human being, these graphical elements seem to be similar, but automatic algorithms have to perform a grouping of the elements before automatic features for retrieval can be extracted. Mostly, only a single component is assumed in

[4] http://oami.eu.int/

[5] http://www.sword.fr/eng/offerings/compnt_acsepto.html

the trademark images which in reality is only rarely the case. Most trademarks are composed of several components

Figure 6.2: Query results for a similarity search based on Vienna code and Nice code, manually filtered results.

As soon as an image is sent in for registration, it is included into the search process to avoid conflicts with images that are not yet activated but in the process of registration.

6.1.3 Codes and classifications for trademarks

There are many different ways to group and classify trademarks. Some of the classifiers are used for legal reasons like the *Nice classifiers*, that group items into a number of similar services and goods where conflicts might arise. Other classifications are created to facilitate the search for similar trademarks like the *Vienna Code*. More information about trademark grouping can also be found at [6]. A simple definition of *trademark* and *service mark* is given as [110]:

> A *trademark* is either a word, phrase, symbol, design, or combination of words, phrases, symbols or designs, which identifies and distinguishes the source of goods or services of one party from those of others. A service mark is the same as a trademark except that it identifies and distinguishes the source of a service rather than a product.

Kind of mark

The easiest grouping of trademarks is into the three different *kinds of marks* that exist:

- individual mark;
- collective mark;
- guaranty mark.

[6] `http://www.ipi.ch/`

In this context, an *individual mark* is the most common mark. Such a trademark is for an individual company to distinguish its services or goods. A *collective mark* is for an entire group of companies, services or goods. A *guaranty mark*, finally, is a mark that guarantees certain standards for the products that bear the mark. This can, for example, be a food stamp which signifies that meat comes from a certain region or country or that it was produced according to certain criteria such as biological or environmental standards.

Form of the mark

Besides the *kind of mark*, it is very important to know in which *form* the marks are presented. We can distinguish between the following different representations:

- 3D mark;
- 2D mark;
- acoustical mark;
- scent mark.

A *3D mark* is for example the Mercedes star which is being explained in a three–dimensional fashion. *Scents* and *acoustical marks* need to be represented on paper for example with notes or formulas, but they are rare and thus there are not (yet) many cases where conflicts might occur. *2D marks* are the most common marks, such as simple brand names (*Coca–Cola*) or graphical symbols for marks such as the *Nike* flash.

2D marks can further be subdivided, depending on the components they contain, into:

- word mark;
- image mark;
- combined mark.

A *word mark* is a mark that only consists of characters such as “Coca–Cola”, whereas an *image mark* does not contain any characters at all such as the Nike flash seen in Figure 6.3.

Figure 6.3: An example for an image mark.

Many trademarks combine characters and graphical symbols and thus are called *combined marks* (see Figure 6.1 for an example). The search for these marks has to be different in several ways, as for marks containing text, not only the visual characteristics are important. It is also important that the chain of characters is not too similar and as well that the pronunciation of the marks cannot be mixed up with the competitors’ marks.

In [110], another distinction is introduced between:

- *trademarks*, where a product is protected;
- *service marks*, where a service rather than a product is protected.

The principal factors to differentiate marks are the similarity of the marks and the commercial relationships between the goods and services identified by the marks.

Nice Code

The Nice code classifies trademarks into a number of groups of similar goods and services. In total there are 42 groups, and a trademark similarity search is always restricted to one or to a few of these groups. A description of the Nice codes can be found at [7]. Some examples are shown here to demonstrate the Nice classes:

Goods

- Class 1 (Chemicals). Chemicals used in industry, science and photography, as well as in agriculture, horticulture and forestry; unprocessed artificial resins, unprocessed plastics; manures; fire extinguishing compositions; tempering and soldering preparations; chemical substances for preserving foodstuffs; tanning substances; adhesives used in industry.
- Class 2 (Paints). Paints, varnishes, lacquers; preservatives against rust and against deterioration of wood; colorants; mordants; raw natural resins; metals in foil and powder form for painters, decorators, printers and artists.
- Class 3 (Cosmetics and cleaning preparations). Bleaching preparations and other substances for laundry use; cleaning, polishing, scouring and abrasive preparations; soaps; perfumery, essential oils, cosmetics, hair lotions; dentifrices.
- Class 4 (Lubricants and fuels). Industrial oils and greases; lubricants; dust absorbing, wetting and binding compositions; fuels (including motor spirit) and illuminants; candles, wicks.

  ...
- Class 33 (Wines and spirits). Alcoholic beverages (except beers).
- Class 34 (Smokers' articles). Tobacco; smokers' articles; matches.

Services

- Class 35 (Advertising and business services). Advertising; business management; business administration; office functions.
- Class 36 (Insurance and financial services). Insurance; financial affairs; monetary affairs; real estate affairs.
- Class 37 (Construction and repair services). Building construction; repair; installation services.

  ...
- Class 42 (Miscellaneous services). Providing of food and drink; temporary accommodation; medical, hygienic and beauty care; veterinary and agricultural services; legal services; scientific and industrial research; computer programming; services that cannot be placed in other classes.

A trademark normally belongs to several of the groups and the membership to a group is determined in the widest sense, so that a mark is rather included in too many groups than in too few. When performing trademark retrieval it is extremely important not to miss a single relevant mark, whereas false positives can more easily be accepted because all marks are controlled manually anyways and false positives can easily be ruled out.

Vienna Code

Vienna codes form the most commonly used description language for trademarks. It is a hierarchical description scheme that includes mostly descriptors of objects and as well classes to describe abstract forms. The printed version of the Vienna code can be obtained from the WIPO[8] (World Intellectual Property Organization) and it is also accessible on the web at [9].

Some of the categories of the Vienna code are shown here to demonstrate the hierarchical structure and the detail of this classification standard:

- Category 1 Celestial bodies, natural phenomena, geographical maps
- Category 2 Human beings
- Category 3 Animals

[7] `http://classification.wipo.int/fulltext/nice/enmain.htm`
[8] `http://www.wipo.org`
[9] `http://classification.wipo.int/fulltext/vienna/enmain0.htm`

- Category 4 Supernatural, fabulous, fantastic or unidentifiable beings
- Category 5 Plants
- Category 6 Landscapes
- Category 7 Constructions, structures for advertisements, gates or barriers
- Category 8 Foodstuffs
- Category 9 Textiles, clothing, sewing accessories, headwear, footwear
- Category 10 Tobacco, smokers' requisites, matches, travel goods, fans, toilet articles
- Category 11 Household utensils

  ...
- Category 28 Inscriptions in various characters
- Category 29 Colors
- 11.1 KNIVES, FORKS AND SPOONS, KITCHEN UTENSILS AND MACHINES
- 11.1.1 Knives, forks and spoons
- 11.1.10 Kitchen utensils and machines
- 11.1.2 Spoons
- 11.1.3 Knives
- 11.1.4 Forks
- 11.1.5 Sets consisting of knife(ves), fork(s) and/or spoon(s)
- 11.1.6 Chopsticks

  ...
- 11.7 OTHER HOUSEHOLD UTENSILS
- 11.7.1 Household utensils not classified in divisions 11.1 or 11.3
- 11.7.3 Coat hangers
- 11.7.4 Clothes–pegs, clothes–pins
- 11.7.5 Brushes, sponges, steel wool (Not including brushes for toilet purposes (10.5.1 and 10.5.25).
- 11.7.7 Brooms, window cleaning instruments

At the IIP, every trademark is characterized with its Vienna code upon inclusion into the database or on query arrival. The coding with Vienna code is done by two different persons separately, to minimize human subjectivity. If the opinions differ, a third person verifies the results. To code an entire trademark database containing more than $100,000$ images is very labor intensive and expensive in this way. However, on the other hand, it delivers good query results, especially when queries are done on an object level (semantic) as it is mostly the case. Problems exist only for very abstract forms, where the visual descriptors might not always be sufficient. Another problem can be the extremely large number of returned, possibly relevant objects, where queries might return up to $10,000$ images that need to be controlled for relevance to the query manually by an image researcher. For trademark retrieval it is essential to have 100% recall, as a single missed but relevant trademark can have legal consequences.

6.1.4 Features and techniques commonly used for trademark retrieval

An extremely large variety of techniques is used to retrieve trademarks. Most of them are of course using *shape features* because color is not very important for trademarks. Color only limits the applicability of a mark as the form is only reserved in this one color, whereas black and white marks reserve the shape in all colors. Most of the classical shape features are described in more detail in Section 2.3.2.

In [116], variant features based on *pixel values* and edges in image regions are used in maybe the earliest visual trademark retrieval system. [46] uses *chain codes* to describe the shapes and then *string–matching* techniques as a similarity function. The STAR system (System for Trademark

Archival and Retrieval) [132, 303] uses a variety of features ranging from *textual* features (with phonetic similarity search) to shape features based on *moments* and *Fourier descriptors*.

Fractals are used in [284], *Wavelet* features in [111], *Hidden Markov Models* (HMMs) in [34], *genetic algorithms* in [30] and *relaxation matching* in [129]. Many systems use *invariant moments* or *Zernike moments* moments as descriptors [39, 110, 254].

The *Artisan* system described in [65, 67, 84] is the only system that uses perceptual grouping to analyze parts of the shapes that belong together. The grouping is based on the theories of Biedermann [16] (see also Section 2.2.3, perceptual grouping). The system then uses a number of *single–valued measures* (*aspect ratio*, *circularity*, etc.) to describe the grouped components (boundary families) of the shape. This system is the only trademark retrieval system that has been evaluated against example queries of a trademark office, although the database used is rather small (10,000 images) for real–world trademark retrieval. Since single Nice classes contain only a few thousand images and as queries are always restricted to a number of these Nice classes, such a database size might be sufficient for demonstration purposes.

6.1.5 Using *Viper* for trademark retrieval

Although we knew from the start that the *Viper* system (described in Chapter 3) was not especially adapted for trademark retrieval, we wanted to perform the user tests to show the feasibility of a trademark retrieval system based on *Viper*. *Viper* does not contain any shape features, neither does it contain invariant features besides the color histogram. Color showed to be unimportant for retrieval and thus, *Viper* only has the Gabor filters as local features and an orientation histogram–based on the Gabor filters as global features.

An example result for the same query image shown in Figure 6.1 with the *Viper* system is shown in Figure 6.4. This (disappointing) result demonstrates well the problems that low–level features can have with the retrieval of trademarks. The results that were manually selected based on search with the Vienna code (shown in Figure 6.2) show how many of the relevant images could not be found by *Viper*. Without a grouping of the image components and a feature extraction locally on the components, no good results can be expected for trademark retrieval.

Goals

The need for user tests existed from the two sides, from the Computer Vision and Multimedia Lab at the University of Geneva and from the Institute for Intellectual Property in Bern. The main goals from the Computer Vision and Multimedia Lab at the University of Geneva were:

- to show the usability and stability of *Viper* in a real application environment;
- to obtain real user interaction data;
- to test the *Viper* system with a large, real–world image database;
- to find out how hard it can be to adapt our system to a specific application field.

The main goals from the Institute for Intellectual Property in Bern for the user test were:

- to get acceptance for new technologies among the users of their retrieval system;
- to find a complement to the textual search by Vienna code:
  - to find out whether some marks were missed through the textual search because of bad Vienna code;
  - to find out if images were missed because they were abstract graphical symbols that were hard to code;
  - to control the quality of the Vienna code;
- to find out what content–based search can offer and what not.

It is clear that most of these goals were of qualitative nature and thus it is more a pilot study than anything else. For quantitative results, a more profound study needs to be undertaken.

Figure 6.4: Query results for a similarity search with *Viper* based on local and global Gabor filter characteristics.

Problems encountered

All along the user tests, we encountered a number of problems that slowed down the development and might also be responsible for part of the unsatisfying results encountered in the user test. First, with the start of the field test (July 1, 2001), we only had a *partial database* containing 50,000 of the 100,000 image marks. This definitely biased the results of the comparison between the *Viper* system and the text–based trademark search. Also, we did not have the *classification* of the marks with the Nice code, so all our searches were performed on the entire database and not on the smaller part relevant for the query. This also changes the performance of the system and especially makes a comparison of our system with the retrieval by Vienna code basically impossible.

When we received the entire database with 100,000 images and the corresponding Nice codes (November 1, 2001), we instantly indexed the new database and performed all the queries on the entire database but we did not have the time to perform queries restricted to the Nice codes given, because this would have meant changes in the user interface and in the query processing.

We also did not have any control on the *questionnaire* that we requested to be filled out after every query. The results show that several queries were executed without filling out the questionnaire, leaving us with a much smaller number of results for the evaluation. There were also several problems related to the questionnaire, where the sheets were not filled out completely or where the data were obviously put into the wrong fields.

The total number of queries performed with the system was lower than the estimated size of 50 per month, so that we had a much smaller number of results.

With respect to the *performance evaluation* of our system in the questionnaire, it was clear that the visual search was being compared subjectively to the search by Vienna code. As most of the searches are for semantics (objects), it was clear that this comparison would lead to imperfect results for our system as only low–level features are used.

We did not get any *updates* for marks either, and thus, images that were inserted into the database shortly prior to the query would not be found with our system. Such a case was reported

from one of the image researchers with a query for the new Swiss airline SWISS. When a retrieval for a similar mark was done, the Vienna code–based system brought up all the newly registered symbols whereas *Viper* did not find anything.

The user test

A dedicated *Viper* server was installed for the IIP and access to the images was restricted to the IP addresses of the IIP and to the University of Geneva. A German version of the PHP interface was created and new images could be submitted via the PHP interface. A *tutorial*[10] that explains the use of the interface was also developed in German and in English and placed on the web to ease the use of the system.

The user test where all new trademark queries of the Institute for Intellectual property were supposed to be executed with the *Viper* system in parallel to the text–based search was started on July 1, 2001. The test data for this thesis includes the interaction data up to April 1, 2002.

Within this time frame, a total of 208 new trademark images were used for queries and a total of 122 evaluation forms were filled out. We recorded 4,895 queries with the web interface of our system which includes test queries with the system done for demonstrations on our side and on the side of the IIP. 1467 queries contained newly submitted images, which means that they are queries of real cases from the IIP. A total of 83,582 images were submitted to *Viper*.

The questionnaire The following questions were included in our evaluation form that was supposed to be filled out by the image researcher after every query with a new trademark image had been performed:

- How do you rate the quality of the system's response? (very good, good, average, bad, very bad)
- How many relevant images were found with *Viper*?
- How many relevant images were found with the Vienna code in total?
- Did you find any images with the *Viper* system that could not be found with the Vienna code?
- How much time did the entire query process with *Viper* take you?

It is clear that some questions have to lead to subjective results such as the question for the quality of the results. The query with our system was performed after the query with the Vienna code was finished, and so it is clear for semantic (object) queries that low–level features will not lead to perfect results. Internally, the ratings were coded as: very good=0, good=1, average=2, bad=3, very bad=4. The default value was set to average=2.

The query for images that were found with *Viper* but not with the Vienna code also contains only limited informations as we did not take the Nice classes into account for the searching as does the search via Vienna code. This may lead to images found to be relevant that are in a different class of goods and subsequently do not conflict with the query image.

Unfortunately, the number of images found with *Viper* was often not filled in. With the query logs we were able to find out how many images were marked as being relevant for the corresponding queries, but this means that relevant images found in the last query step (and consequently not send to be relevant via *MRML*) might not be taken into account. This might lead to overly bad results for the number of images found with *Viper*.

Usage of the system The usage of the system was logged in several ways. The questionnaires were stored in a file with date and time, all the newly submitted images were as well stored with date and time. The third logging is in the *Viper* log file, where all interaction is stored. These logfiles can be very large. The files for the 9–month user test have a size of 69 MBytes in *MRML*

[10] `http://viper.unige.ch/tutorial/`

format (see Section 3.2 for details about *MRML*). This large quantity of information can make an evaluation rather hard and manual inspection impossible.

Most important for the evaluation were of course the questionnaires because they can give us the users' opinions about the system. Unfortunately, they were often not used or only partially filled in. An analysis of the replies filled into the questionnaires can be found in Table 6.1.

	images	questionnaires	rating	rel. images	*Viper*	new images	time
7/2001	5	3	3	23.67	0.33	0.33 %	10.33
8/2001	40	16	2.56	29.75	4.94	0.25 %	15.81
9/2001	33	18	2.61	22.83	3.5	0.06 %	20.69
10/2001	24	5	2.8	23.8	2.6	0.2 %	19.8
11/2001	26	16	2.69	15.25	2.31	0.06 %	16.87
12/2001	30	20	2.9	25.15	2.5	0.25 %	26.35
1/2002	7	5	3	20	0.6	0 %	20
2/2002	20	19	2.84	26.85	1.85	0.47 %	31.15
3/2002	23	20	2.7	20.6	3.2	0.35 %	31.45
total	208	122	2.72	23.05	2.86	0.24 %	23.3

Table 6.1: Result of the questionnaire for the usage of the *Viper* system for trademark retrieval.

The longer time taken to execute queries in the end of the test period is most likely due to the use of more relevance feedback. In meetings before this period, the use of relevance feedback for successful querying was stressed. Still, negative feedback was not used selectively to refine the query, but rather all images were marked as non–relevant even when there was not a single relevant image. Even when using feedback that separately weights positive and negative parts of a query, this will lead to suboptimal results. This misusage of negative feedback might explain a large number of queries where not a single relevant image has been found. Once one or more relevant images were found, the feedback was generally effective and led to further hits, as was reported by several image researchers.

Conclusion of the user test

We can see in the evaluation of the questionnaires that some of the initial goals were not reached by the user test. This can partly be explained with the following problems:

- Lack of the complete dataset and Nice codes due to technical problems at the start of the user test.
- Lack of time to implement Nice codes when they were finally received, so the queries were always executed on the entire database.
- Lack of training for the users of our system, so several mistakes were done, especially with the use of too much negative relevance feedback, which led to unsatisfying results.
- The feature set of *Viper* does not contain any characteristics that are especially effective for trademark retrieval. For proper trademark retrieval the image needs to be grouped into components, and then shape features need to be extracted from the components to allow effective retrieval.

On the other hand, some of our beforehand stated goals with the user test were reached:

- Several of the queries (24%) brought up images that were not found with the text–based system, although this is partly due to positive images from other Nice codes that are thus not relevant to the query.
- The system showed to be stable when used with large databases and with several users at the same time (the *Viper* server went down only three times within the test period of nine months, once due to a crashed harddisk).

- The web–based infrastructure proved to be easy to maintain and easy to use.
- Times for the query process were rather low, so they did not have a negative influence on the rest of the workflow. 20 minutes for the entire query process including several steps of relevance feedback were significantly faster than queries with the query system based on Vienna code.

For a subsequent user test much more work would need to be done, especially in the system development of *Viper* to specialize the system for trademark retrieval. This includes the use of the Nice codes and the implementation of specialized features that have shown to be effective with the use of trademark images.

6.1.6 A framework for trademark retrieval

A complete framework for trademark retrieval would be an *integrated* system that uses visual characteristics of the trademarks as well as textual coding in Vienna code. Also, the Nice codes need to be taken into account for the querying process, so queries are restricted to the important groups of goods and services.

The visual features need to be extracted on a *component* basis as global features or features gained from partitioning showed to be rather ineffective. Learning from the user interaction can become a major part for the visual feature system. Learning can prove to be very effective, especially when done on an object basis.

The combination of the *visual and textual* features can be done in a way that the visual features are used to help with the *ranking* of the result images. All images that have some similarity in the Vienna code still needs to be observed on screen, because 100% recall is essential in trademark retrieval. Still, a better ranking of the images might lead to better overall results. When an image observer has to control 10,000 images, for example, his attention will go down and the error rate might go up. With the visually similar images being ranked highly, it can be expected that less important images are retrieved late, at the time the image researchers might get tired. The visual features can also bring up new images that might not have been found otherwise. This can also be seen in the results of the questionnaire in Table 6.1.

Visual features can also be used for *quality control* of coding in the Vienna code. With a number of known examples it can be attempted to find images with possibly missing marks. In databases of more than 100,000 elements it could be expected to have a number of errors in the system.

Speed of an interactive system is also an important factor. It was shown that even with databases of 100,000 images, fast query responses can be reached. With *Viper*, an entire query session took an average of 20 minutes including several steps of feedback and image download. Even with the entire Vienna code being integrated into *Viper*, a query should not take longer than 10 second, and most likely even a response time of under 1 second is achievable when implementing good pruning techniques.

6.1.7 Conclusion

Part of the cooperation has shown to be a failure, but part of it has proven to be a success. It was experienced that the *GIFT / Viper* framework is stable enough to be used in a professional environment every day. It has also shown to be fast enough for the use with large databases. Much experience could be gained in the field of trademark retrieval and the limitations of low–level features sets became apparent. The tests showed that when real trademark retrieval is aspired, much more than rotation–, scale– and shift–invariant features are needed. Only few of the trademarks submitted as queries to our system contained only one component. Most of the images contained several components and thus some perceptual grouping seems to be essential before features can be extracted from the components.

Trademark retrieval based solely on automatically extracted visual features seems to be, at least for now, a utopia. Visual features are rather needed as a complement to the textual features coded in Vienna code. The visual features can help to find hard–to–code trademarks or marks

where there were errors in entering the Vienna code. Visual features can also be used to control the Vienna codes of groups of images with similar symbols to make the codes more consistent, or simply help in coding the images.

6.2 Medical Images

Medical image retrieval has started to become more and more popular within the field of content–based retrieval. After only a small number of publications in the early years, the figures have been going up in the last few years.

Viper has so far not been used for medical image retrieval and the project is in a very early stage. Therefore, this section can merely be seen as part of the "Future work" (Section 7.3).

6.2.1 Application fields within the medical domain

Content–based images retrieval can be of help in the medical domain wherever images are analyzed manually, based on their color or texture, or based on certain structures.

Naturally, this can be the case in *Pathology* where cuts of tissues are analyzed by the pathologists. Many pathologic cases are standard work, but when the physician is unsure, (s)he might consult books with representative cuts of the same tissue and compare the structure. A CBIRS in connection with a database of standard cases can be of great help in this process and provide images that are similar with respect to certain visual aspects very fast.

Dermatology is another department where images are sometimes compared with standard examples [134]. Especially the observation of lesions over time can reveal the malignancy of a mark. Texture and color play and important role, and thus proper calibrations need to be done so images that are compared over time are really taken under the same conditions.

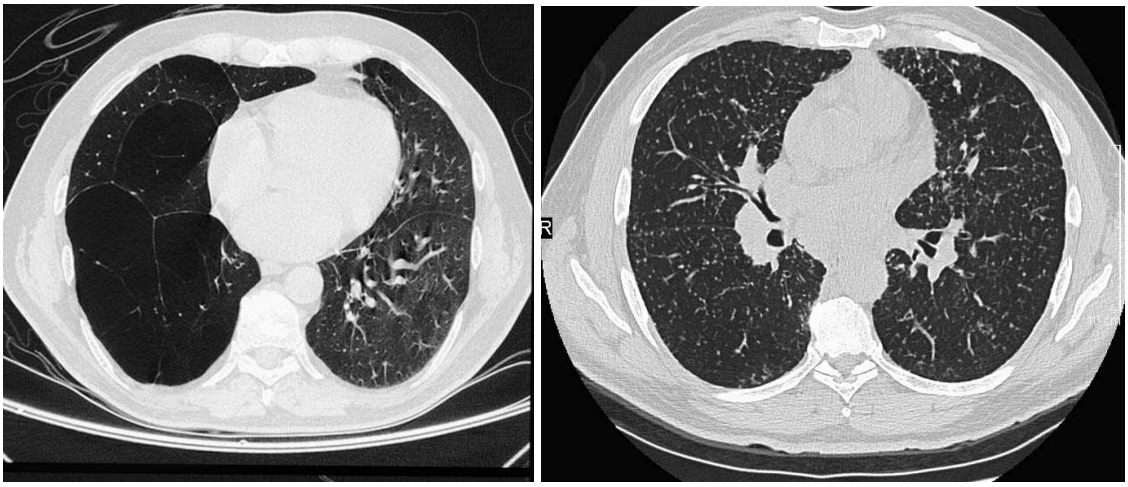
Hematology is a very old field for computer vision applications. In blood smears, the red and white blood cells need to be counted and several automatic algorithms for segmentation and cell counting exist [42]. To our knowledge, no implementation of content–based query techniques has been proposed for hematology as of yet. Such an application could make sense in not only comparing patterns with respect to numbers of red and white cells, but especially with analyzing untypical white cells that can be very characteristic for certain diseases.

Most experience with images definitely exists in *Radiology* where there are large numbers of diverse images produced and where many computer–based techniques for the analysis of images have been developed. Images produced in radiology are very diverse and many applications can be imagined, for example the one described in Section 6.2.2 for lung images. Other applications are the search for tumors, based on shape, gray–level and texture. In many image archives content–based access methods can well complement the established exact retrieval techniques. This can be very useful for teaching as well as for scientific studies where certain specialized cases might be needed. Also in this field, a mixture of visual and textual features can be expected to lead to the best results as it does for trademark retrieval.

6.2.2 A project for lung image retrieval

The goal of this project is to create a database with lung images for a number of lung diseases as a reference database. Based on such a reference image database, image features derived from cooccurence matrices or other texture descriptors can be applied and their performances on the database can be compared to develop a set of descriptors well suited for the retrieval task. A few examples images of the lung taken with a HRCT (High Resolution Computed Tomography) are shown in Figure 6.5.

Applications for such a system can be as well in *diagnostics* as in *teaching*. To evaluate the retrieval system, new cases need to be classified and then queried with the system. Like this, the texture descriptors can become more and more optimized for the query tasks based on the examples used for querying and the user feedback on these examples.



(a) Emphysema (b) Micro nodules

Figure 6.5: Examples for HRCT images of the lung.

The indexing of new images will strongly rely on *manual interventions* on several levels. The *slices* of the lung volume that are characteristic for the disease in the lung are chosen by an MD who also marks the *regions* (PBR, Pathology Bearing Region) in the slices that characterize the disease in the best way. Other informations stored with the image are also used for the retrieval such as the thickness of the slices. These factors can significantly alter the appearance of an image. The diagnosis also needs to be stored with the image so that retrieval can be performed based on the diagnosis as well.

The query process will thus include features extracted at possibly several regions of an image or even from several slices containing several regions. The combination of visual features with technical features such as slice thickness is also important to compare images based on the same grounds.

It is planned to incorporate and compare a variety of texture descriptors. A pilot application is supposed to work with the Gabor filters of the *Viper* system but these features need to be replaced as soon as possible by features better suited for this sort of texture analysis such as features based on *cooccurence matrices*, *statistical* features and *gray level distributions*. Small nodules can be analyzed using methods of *mathematical morphology* to measure the average size of the nodules and their roundness. A number of texture descriptors has been used for lung image retrieval in [21, 141, 244, 245].

6.2.3 Conclusions

Although there has not yet been an application of the *Viper* system in the medical domain, there are several interesting and challenging problems in medical image retrieval. Images of lung diseases rely on the analysis of the lung texture and thus they seem to be well suited for the use with an image retrieval system. Applications can be in the diagnostics as well as in teaching. It can help well to bring up examples for the different *perceptual categories* that are normally used for classifying these lung images. Sometimes these *perceptual categories* can be subjective and with the help of a retrieval system such a categorization can be compared with objective texture features.

Such an application can also give way to follow–up applications in medical image retrieval. Once acceptance has been reached in one project, other projects might have an easier acceptance. There are several other domain of medical images that are well suited for CBIR, namely pathology and dermatology.

6.3 Summary

This chapter explained two possible applications for CBIRSs. A short literature review for the two fields was done and propositions are made for what needs to evolve in *Viper* to create applications for these two fields.

A user test with real trademark searches of the Institute for Intellectual Property in Bern was done and the subjective quality of research results was measured in a questionnaire. The results show that every domain has constraints that need to be regarded and that image retrieval still is far away from semantic retrieval which would ideally be needed to retrieve trademarks. The results also show that the implementation and use of content-based retrieval techniques is stable and that searches can be performed quickly, so that the normal workflow does not get interrupted.

There are many applications where image retrieval can be of great use. The medical area can become one of these areas as in medicine many images are used, and query by content may be very useful for several applications. The description of a possible application in lung image retrieval shows this usefulness.

Chapter 7

Conclusion

> Nicht alles was zählt, kann gezählt werden, und nicht alles was gezählt werden kann, zählt. (Not everything that counts can be counted, and not everything that can be counted, counts.)
>
> Albert Einstein

This thesis gives an overview of the domain of content–based visual information retrieval. The different basic problems are described and most of the standard techniques and features of retrieval systems are explained. Some example systems are described in more details. The two fields of user interaction in visual information retrieval and of evaluation of retrieval systems are then thoroughly discussed; this constitutes the main scientific achievements of the thesis.

This last chapter gives an overview of the original work presented in this thesis and mostly concerns the Chapters 4 (user interaction) and 5 (evaluation of retrieval systems). It states the scientific achievements, and then, takes an outlook or insight into the future of the *GIFT*/*Viper* system is given. A description of promising future research directions for CBIR is stated as well.

7.1 Summary

The preceding chapters give development details of our *Viper*/*GIFT* framework, including an overview of how the system evolved over times into the framework that exists now. Such a framework opens the possibility to try out many different techniques quickly and, often, little changes significantly improve the performance of the system. The decision to rely on techniques from text retrieval proved to be useful in many ways as text retrieval is a mature and well studied field of research. Many techniques from text retrieval, such as relevance feedback, have been reinvented in image retrieval. Thus, it makes sense to take a more formal look at problems and solutions encountered in the text retrieval domain. Despite a relatively simple and easy–to–calculate set of features used in *Viper*, a good retrieval performance can be achieved when using various image databases, as detailed in the manuscript.

The use of a standard query and communication protocol (*MRML*, Multimedia Retrieval Markup Language) to access a server–based system has many advantages and strongly influenced our subsequent work. It was realized early that this does not only produce a separation of user interface and actual query engine, but it also defines a standardized access to the system for other applications such as meta search engines and automatic benchmarks. It also makes possible the storage of large interaction log files from our web demonstration in a standardized format to ease their analysis.

These interaction logfiles also form the basis to learn from the user interaction with our system, not only within a query session of a user but over several temporal scales. Such learning leads to large performance improvements when performed based on the interaction that was done with the same set of images. In Chapter 4, a framework is presented to learn over several levels. Within–session learning, but also learning over all users and databases, learning over user groups or image

image databases with similar characteristics and learning on an individual user basis are proposed. Such a *hierarchy of learning* can lead to further enhancements of the retrieval quality when large logfiles of user interaction are available. The quality of the log files of course plays an important role as well. So far, our logfiles are created from a web demonstration version where the quality might not be very reliable. We still believe that when looked at from a large scale, useful information can be derived from them.

With real users working on real query tasks this quality is expected to improve. Also, the users will log into the system with their real names for profiling, because they have an interest in a good personal profile that might facilitate their future work. By contrast, in our web demonstration most users log in with the default name "anonymous".

Of course the most important part described in this thesis is the development of a benchmark or a benchmarking framework for the field of content–based visual information retrieval. Again, the use of *MRML* helps us for the project as it opens the access to the system to scripts. Based on *MRML*, a completely automatic benchmark has been developed that uses a web interface to evaluate query engines automatically at any location of the world with an internet access. This automatic evaluation includes the integration of several steps of relevance feedback, and the benchmark can also be configured in several different ways, allowing the addition of performance measures and the adaptation to new image databases and new relevance judgments. Practical experiences need to be gained now to really compare several systems in such an automatic evaluation framework.

The chapter on evaluation also describes other possible areas of image retrieval that can be evaluated on a benchmarking event, and it describes prerequisites that are needed for such a benchmark to be successful and accepted in the field. The *Benchathlon* has all chances and possibilities to become such a successful benchmark for the field of image processing but much work still needs to be done. The most important work for the moment might be the creation of an image database. Such an image database needs to be an accepted standard in the field and it needs to be available free of charge so every research group can obtain it. Essential are also the definition of query tasks for the database and ground truth data for the query tasks. Work is underway to develop annotation tools and an annotation vocabulary that can be used for ground truthing. The inclusion of real users into the process of ground truthing seems very important to model at least part of the user subjectivity.

7.2 Recall of the main achievements

This section recalls the main scientific achievements of the thesis. These contributions have already been mentioned in the introduction part and have been detailed in the different sections of this thesis. Especially the section on evaluation and user interaction bear the content of these scientific contributions.

Contributions to the *Viper* system include the following:

- Ideas about the structure of the *Viper* system and the distribution of the system were developed [179]. This includes ideas about the distribution of the computation via CORBA [179].

- A better feature weighting approach based on separately weighting the different feature groups was implemented. Such a weighting massively reduced the influence of very frequent small textures that caused problems in earlier *Viper* versions. This new weighting was first used in [260] and in all publications afterwards.

To improve on the usability of the system and especially to make it more interactive, several ways of *search pruning* were implemented to speed up the system response time:

- Part of the weighting factor were pre–calculated for a computational gain to speed up the query process.

- A parallel version of *Viper* was tested and implemented to produce methods for parallelization of the computation using PVM [181].
- Search pruning was tested in many different ways to enhance the usability and interactivity of *Viper* [181, 260]. This includes the evaluation of the information content of frequent features and how search pruning affects the retrieval performance of a CBIRS.

Several other aspects in the field of *user interaction* or more precisely relevance feedback management were analyzed within the framework of this thesis:

- An analysis of various strategies for relevance feedback was performed to compare their performances [177]. This can *e.g.* be used to automatically help novice users by adding negative feedback to a feedback query to receive better results. It can also be used to automatically generate relevance feedback for evaluating the system performance [172].
- Relevance feedback with separately weighting positive and negative documents in a query as proposed by Rocchio [218] was implemented. This helps to avoid the previous problems with the use of too much negative feedback in a query. Too much negative feedback can also eliminate good and wanted features from a query [177] by a negative weighting.
- Analysis of large logfiles of user interaction data and how we can improve the system performance by using this information was demonstrated [175].
- Importance factors were derived for features based on the interaction log files we stored, [171, 175]. This long–term learning can massively improve the performance of a system. A comparison with the very similar problem of market basket analysis (MBA) was done and fast algorithms to calculate association rules are used.

The main contributions of the thesis are certainly in the field of retrieval system *evaluation and benchmarking*:

- Several user experiments were performed to gain relevance judgments of real users to evaluate the *Viper* system, used e.g. in [185, 261, 262, 264].
- Methods to evaluate the performance of CBIRSs were reviewed and compared [178]. Proposals were made for the evaluation of CBIRSs based on evaluation in text retrieval, a much more mature field with respect to evaluation. The Text REtrieval Conference (TREC) especially inspired this research.
- Automatic scripts were created to evaluate retrieval systems based on *MRML*. This includes the creation of several steps of relevance feedback and the evaluation of the performance with relevance feedback [172].
- A web–based benchmarking framework[1] was developed to support the evaluation separate from the location of a system [173, 174]. Such a permanently available benchmark server can be a good complement to a benchmarking event, where research groups can compare systems and techniques.
- The *Benchathlon*[2] benchmarking effort for CBIR at the SPIE Photonics West conference was supported and proposals were made for a technical infrastructure.
- The creation of a freely available image database for evaluating CBIRSs[3] was supported by making available several groups of images.
- Tests with the Corel database were performed to show that evaluations using different subsets of the Corel Photo CDs can lead to very differing results [170]. This underlines the need for an image database for evaluation, including query tasks and ground truth data.

[1] `http://viper.unige.ch/evaluation/`
[2] `http://www.benchathlon.net/`
[3] `http://www.cs.washington.edu/research/imagedatabase/groundtruth/`

7.3 Future research directions

There are many possible and interesting directions for further research as the field of visual information retrieval only starts to grow up. The need for a benchmarking event and especially for image databases to perform this benchmarking becomes more and more apparent. An even stronger need might be generated through upcoming, real applications of the technology, because then the systems have to prove their performance. In return, this might bring more interest and consequently more research money to the benchmarking field.

7.3.1 *Viper / GIFT*

Within the *Viper* or *GIFT* framework there are many ideas and possibilities for improvements. One is certainly the *stability* of the system with respect to wrong or not well–formed queries. This has a close connection with separating the communication and treatment of *MRML* from the actual search engine part. This means that the communication part has to be very stable and a crash by a query accessor does not crash the entire communication. This is in close connection with the functionality of database management functions demanded in Section 7.3.5. These management function also need a clearer division of the management/communication and the query part.

With *GIFT* being a GNU project now, the number of users and possible developers is foreseen to rise which can automatically lead to a better detection and removal of errors. The fact that we already had more than 11,000 test queries on our PHP demo interface for March 2002 alone already shows this grown interest and underlines the need for a very stable core version.

7.3.2 A variety of media

From the start, the retrieval of several media was a goal of the *GIFT / Viper* project. For this reason we use URLs and other web technologies for the access to the documents. A document can easily be an HTML page containing several media, or a SMIL page containing these media even in a dynamic, synchronized form. So far only the mixture of text and visual characteristics has been integrated into *GIFT / Viper* [38].

A project is underway to integrate video data into the *GIFT* framework. *Fer de Lance*[4] is another project started for using *GIFT* on the desktop to search for varying media and the most diverse file types.

Both projects would strongly profit from a more object oriented and open structure of the way the features are stored in *Viper*. Such a framework should make it possible to extract and store features on a global document level, but also take into account smaller objects in the document separately. These smaller objects can, for example, be images in a main document being a web page or they can be object in a global image.

The features of all these sub–objects need to be stored and accessible for retrieval as well. Like this, each document can be visualized by a tree like structure and the user can chose the branches or sub branches of the tree (s)he is interested in. Certain levels of a tree might correspond to a completely different level in another tree as one tree can start with a web page as a global level whereas others start with an image or even with a single extracted object as the global level. Experiments for storage structures eventually based on inverted files need to be made for these tree structures to get fast and efficient retrieval.

7.3.3 Real applications

Our experiences in Chapter 6 have shown that it is not easy to adapt a general purpose image retrieval system to a specialized domain such as trademark retrieval because many constraints need to be taken into account. Still, these real–world applications are necessary as only these user tests can reveal to us what kind of work is needed on current systems to make them usable in a professional work environment. The few real applications studies that have been performed

[4] `http://www.fer-de-lance.org/`

and reported brought a lot of new insights and changed priorities for the system development [151]. Most current systems are, by contrast, only working with datasets from a specific domain but without taking into account the real work scenario and real relevance data for the evaluation [46, 116, 118].

Besides the use as a *trademark* retrieval systems, there are several foreseen application projects for *Viper* at the moment.

One project is in the *medical* domain where images of different lung diseases are planned to be analyzed. The textures of these images need to be analyzed in detail and then compared with a number of stored cases to assist an MD in the diagnostic process. Such a tool can also be used for teaching because the visual description groups used in the diagnostic process are not always extremely clear and are sometimes even disputed among experts.

Another project foreseen is with *nano images* in cooperation with the CERN[5] and the Danish Microelectronic Center MIC[6]. In the field of nano images, it is often needed to determine what kind of molecule is contained in a certain image. Large databases with pictures of known molecules taken from various directions exist. Effective features for similarity still need to be developed to retrieve such images.

7.3.4 Features

Real applications will automatically create a need for specialized feature sets for image retrieval. Although the feature sets used for *Viper* work reasonably well for general purpose photography databases, there is a lot of room for improvement on a feature basis in limited domains. Each special domain, such as trademark retrieval, creates an *a priory* knowledge that can be used for feature creation and will lead to better retrieval results than general purpose features.

It might also be interesting to extract a very large number of features for a certain database and then compare the performance of all these features, which can be done with the same techniques as the long–term learning in Section 4.6.1. The feature importance calculated from user interaction log files can then be used to automatically select the best feature sets for certain query tasks.

Viper, with its possible large but sparsely populated feature space is also very well suited for the use of very specific features such as detectors for water, sky, mountains or grass. Even more specific features such as a cat/dog/tiger detector or even a detector for certain persons in images are possible (such as a "George W. Bush" detector). As for any given image only a few of these possibly many detectors will give a positive result, we will still be in a very sparsely populated space and the inverted file will give us a very fast access even with a large number of detectors. Such a feature space corresponds well to the theory of perceived similarity proposed in [274]. Tversky characterized stimuli as sets of binary features instead of considering them as points in a metric space.

All these extensions of the feature space will make a new, more general numbering system for the features necessary. So far the features are simply number from $1..N$ and as types we have a choice between histogram and block features. Large feature spaces where features can be selectively used for indexing and where different databases might have different features sets will make a more elaborate *feature ID* system necessary. It will then be feasible to determine which features are available for a certain image and how they can be used to query a database with a possibly different feature set extracted.

7.3.5 Management functions and security

Other additions to the *Viper* system are certainly the integration of *database management functions* into *MRML* and *GIFT*. This means that commands like creating a new database or adding images into a database and the regeneration of the inverted file should be possible via *MRML*. This of course has many implications as this means that a *right management* system has to be created, so only privileged users can do certain operations. Such a right management needs

[5] `http://www.cern.ch/`
[6] `http://www.mic.dtu.dk/`

to include users and user groups and properties can be divided into creating databases, submitting and deleting images and executing queries.

This again has implications on the *security* side as user names and passwords need to be transmitted. They are already transmitted via *MRML* at the moment, but only in plain text. Such an unencrypted communication is thus not possible anymore when security–relevant operations can be done via *MRML*. We need to think about ways to encrypt the *MRML* data stream, so nothing can be obtained when listening to the communication. Eventually this can be done through communication via secure, encrypted sockets.

Some of the functions needed for a system that can be managed completely via the web include:

- *creation* of a new, empty image database with its properties like user/user groups with read/write access, establishment of feature sets used and possibly of other parameters;
- *deletion* of an existing image database;
- *addition* of a new image into an image database;
- *deletion* of an image from the image database;
- *retrieval* of a list with all the images in a certain image database;
- eventually the triggering for the *recreation* of the inverted file, so this does not need to be coupled with the addition and deletion of images to/from the database.

All this can make *Viper* a well–manageable system like relational database systems are at the moment, only with using images and (possibly) inverted files. A closer look needs also to be taken into the database literature to not reinvent things but build on well known principles and the experience of the database community.

This can, of course, facilitate the use of *Viper* for real applications as the ones described in Section 6. The field of CBIR is still young and more experience has to be gained with real users. Real user test such as explained in [151] bear much information that needs to be exploited to create usable applications.

7.3.6 User interaction

Many techniques to improve the retrieval results and the usability of retrieval systems with the help of user interaction have already been described in Chapter 4. This field still has a lot of potential for improvement and might be the key to narrowing the semantic gap as fully automatic methods have more or less failed as of yet.

User interaction can for example be used with interfaces that explain to the user more about the underlying feature structures so he knows why certain images have been received. This can help for the selection of relevance feedback and also for the long–term learning. Another possibility are active systems that actually propose to the user different ways to continue the query or ask concrete questions to, for example, exclude a complete part of the image space from a further query process.

Much information is for sure contained in the interaction log files from demonstration versions and we need to have a standard of these log files and the used image databases so research groups can exchange their knowledge about the interaction with users.

7.3.7 Evaluation

For the moment, the biggest challenge in image retrieval is certainly the creation of a benchmarking event where research groups can compare techniques and the performance of these techniques. This will lead to a performance gain in the field like in other fields, as promising techniques can be identified and based on the benchmark, the techniques can be optimized. Combining the best features for a certain domain with the best learning models and the best similarity functions will almost certainly improve the results.

To achieve such a benchmark, much work is necessary. First a database or even better, a set of databases from several fields need to be developed to do the benchmarking on. The databases need to be diverse and not just simple objects, so they are suited for diverse query tasks. Based on these databases, query tasks can be defined and relevance judgments have to be created. Care has to be taken to create a method for gaining relevance judgments that takes into account user subjectiveness, but at the same time does not need a complete recreation of relevance judgments when the database is enlarged.

When all this is set up, we can start to evaluate different performance measures and other factors, and we can compare their information content and the correlations among the different performance measures. This will eventually lead to a stable set of performance measures that are easy to interpret and where each of the measures contains a different information.

Most important of course is, to get acceptance and participation for such a benchmark in the field. It has to be stated that in the long run, everybody will profit from such a friendly evaluation forum.

Interpretation is the revenge of intellect upon arts.

Notation

This chapter explains the mathematical notation used throughout the thesis to keep it as homogeneous as possible. It also gives a link to where the symbol was used for the first time to get further information on it.

$\rightarrow$ Connection for an association rule (used in Equation 4.2)

$(I_a \rightarrow I_b)_+$ Positive association rule (introduced on Page 83)

$(I_a \rightarrow I_b)_-$ Negative association rule (introduced on Page 83)

a Additional factor for the calculation of feature weights (defined in Equation 4.5)

$\mathcal{A}$ Answer set of images *images returned for a query* (introduced on Page 97)

A_i The answer image number i for a certain query, ranked retrieval result (introduced on Page 97)

B Half peak radial bandwidth of Gabor filter (defined in Equation 3.2)

C, D, E used for various purposes to calculate factors

cf Collection frequency *the frequency of a feature (word) in the entire collection* (used in Equation 3.4)

cf_j Collection frequency of feature j (used in Equation 3.4)

df Document frequency *(see tf)*

ϵ Added to avoid overflows with logarithmic functions in the weighting function (used in Equation 3.8)

ε_i Single images from the evaluation set $\mathcal{E}$ (introduced on Page 97)

$\mathcal{E}$ Evaluation set *set of images chosen for an evaluation run as query images* (introduced on Page 97)

f Frequency (used in Equation 4.2)

$\mathcal{F}$ Feature space (used on Page 38)

F_j Feature *a certain feature from the feature space* (used on Page 38)

g Generality (defined in Equation 5.3)

g_{mn} Gabor filter *with scale m and orientation n* (defined in Equation 3.1)

h_{qj} Feature "importance" for a query q and a feature j (defined in Equation 4.6)

i, j, k, l, m Counters for diverse purposes

I_i A certain image from a collection (used in Equation 4.2)

$I(x, y)$ Pixel value of an image at position (x, y) (used in Equation 2.1)

$\mathcal{I}$ Image space *a set of images used as image database* (introduced on Page 97)

lf Logarithmic factor for the weighting functions (defined in Equation 3.8)

M Number of features in a query q (used on Page 47)

n Number of query images (used on Page 38)

N Number of response images retrieved (used on Page 38)

$N_{\mathcal{E}}$ Number of evaluation images *numbers of images an evaluation run is done with* (used on Page 53)

$N_{\mathcal{I}}$ Number of images in the image set (used on Page 53)

$N_{\mathcal{R}}$ Number of images in a relevance set for a query image (used on Page 53)

$N_{\mathcal{Q}}$ Number of images in the query (used in Equation 3.6)

nr Non–relevant images (defined in Table 5.1)

$nrnr$ Non–relevant not retrieved images (defined in Table 5.1)

nrr Non–relevant but retrieved images (defined in Table 5.1)

P Precision (defined in Equation 5.1)

$\overline{P}$ Averaged precision (described on Page 95)

$P(n)$ Precision after n images are retrieved (defined in Equation 5.1)

$PR-Graph$ Precision/Recall–graph (described in Section 5.2.6)

p Probability (defined in Equation 4.2)

$p(I_i \rightarrow I_j)$ Probability that from relevance of image i follows relevance of image j (defined in Equation 4.2)

q Query (used on Page 47)

$\mathcal{Q}$ Query set of images (used on Page 38)

Q_i Query image number i (used on Page 38)

Q Pseudo–Query image *calculated from all the query images in* $\mathcal{Q}$ (used on Page 48)

r Relevant images (defined in Table 5.1)

rnr Relevant but not retrieved images (defined in Table 5.1)

rr Relevant and retrieved images (defined in Table 5.1)

R Recall (defined in Equation 5.2)

$R(n)$ Recall after n images are retrieved (defined in Equation 5.2)

$Rank_1$ Rank of the first relevant image (described on Page 5.1.2)

$\mathcal{R}$ Relevance set *set of relevant images* (introduced on Page 97)

$\overline{Rank}$ *Averaged Rank* (described on Page 5.2.6)

$\widetilde{\overline{Rank}}$ Averaged rank with a penalty for non–retrieved images (defined in Equation 5.4)

$\widetilde{Rank}$ *Normalized average rank* (defined in Equation 5.8)

$\widetilde{Rank}$ *Normalized average rank with penalty for not retrieved images (MPEG–7)* (defined in Equation 5.5)

Rel Relevance *Relevance given by a user, in* Viper *in general* $\in \{-1, 0, 1\}$ (used in Equation 3.6)

$Rel(Q_i)$ Relevance of one particular image in the query set (used in Equation 3.6)

$\mathcal{S}$ Scoreboard of images that have features in common with the query

S_{qk} Score of a result image k to a query q (used in Equation 3.4)

$S(I_i)$ Score of image I_i

t Time i.e. *time it takes to execute a query* (used on Page 5.2.6)

tf Term frequency *the frequency of a feature (word) in a document* (used in Equation 3.4)

tf_{ij} Document or term frequency of feature j in image i (used in Equation 3.4)

u Number of different user judgments for the evaluation process

u_0 Center frequency of a Gabor filter (used in Equation 3.1)

wf Weighting function (used in Equation 3.9)

wf_0 Basic weighting function (defined in Equation 3.7)

x, y, z Used for various purposes

α Factor for Rocchio weighting of positive and negative components in a query (used in Equation 4.1)

β Factor for Rocchio weighting of positive and negative components in a query (used in Equation 4.1)

μ Penalty for non–retrieved images (used in Equation 5.4)

Glossary

This chapter contains a list of the most common abbreviations in the field of content–based image retrieval and all the abbreviations used in this thesis, to avoid confusion.

ACM Advanced Computing Machinery

AI Artificial Intelligence

API Application Programming Interface

ARPA Advanced Research Project Agency

ARTISAN Automatic Retrieval of Trademark Images by Shape ANalysis

CA Correspondence Analysis

CACM Communications of ACM

CBIR Content–Based Image Retrieval

CBIRS Conten–Based Image Retrieval System

CBMR Content–Based Multimedia Retrieval

CBVIR Content–Based Visual Information Retrieval

CBVR Content–Based Video Retrieval

CCD Coarseness, Contrast, Directionality

CDU Classification Décimale Universelle

CERN Centre Européan pour la Recherche Nucléaire

CGI Common Gateway Interface

CHI Computer–Human Interaction

CIE Commission Internationale de l'Eclairage, International Commission of Illumination

CLEF Cross–Language Evaluation Forum

CMY Cyan, Magenta, Yellow, color space

CNN Cable News Network

COGIR COGnitive Image Retrieval

CSS Cascading Style Sheets

CSS Curvature Scale Space

CV Computer Vision

DARPA Defense Advanced Research Project Agency

DB Database

DCMI Dublin Core Metadata Initiative

DoD Department of Defense

DOM Document Object Model

DS Description Scheme

DTD Document Type Definition

EM Expectation Maximization

EPFL Ecole Polytechnique Féderale de Lausanne

EU European Union

FFC Fuzzy Feature Contrast

FIDS Fast Image Database System

FilIB Filter Image Browsing

FRIP Finding Regions in Pictures

FSF Free Software Foundation

GIDB General–purpose Image Database

GIFT GNU Image Finding Tool

GIS Geographical Information System

GNU GNU is Not Unix

GPL General Public License

HCI Human–Computer Interaction

HMM Hidden Markov Model

HRCT High Resolution Computed Tomography

HSV Hue, Saturation, Value, color space

HTML Hyper Text Markup Language

HTTP Hyper Text Transport Protocol

IBM International Business Machines

ICDM Image Content Data Model

IDB Image Database

IDQS Image Database Query System

IGE Institut für Geistiges Eigentum

IIP Institute for Intellectual Property

IM2 Interactive Multimodal Information Management

IP Internet Protocol

IPI Institute pour la proprietée Intellectuelle

IR Information Retrieval

ISQL Image Structured Query Language

IVSR Image and Video Storage and Retrieval System

KDE K Desktop Environment

MARS Multimedia Archival and Retrieval System(s)

MAT Medial Axis Transform

MBA Market Basket Analysis

MD Medical Doctor

MDL Minimum Description Length

MDS Multi Dimensional Scaling

MFD Modified Fourier Descriptor

MIRA Multimedia Information Retrieval Applications

MIT Massachusetts Institute of Technology

ML Machine Learning

ML Maximum Likelyhood

MM Multimedia

MMDBMS Multi Media Database Management System

MP3 MPEG–1 Layer 3

MR SAR Multi Resolution Simultaneous Auto Regressive texture model

NBC National Broadcasting company

NIST National Institute of Standards and Technology

NN Nearest Neighbor

OCI Oracle Call Interface

OCR Optical Character Recognition

OOI Object Of Interest

PBIR Perception–Based Image Retrieval

PBR Pathology Bearing Region

PCA Principal Component Analysis

PDBS Pictorial Data–Base System

PDF Portable Document Format

PHMM Pseudo Hidden Markov Model

PHP PHP: Hypertext Preprocessor

POI Point of Interest

PR Pattern Recognition

PR graph Precision *vs.* Recall graph

PVM Parallel Virtual Machine

QBD Query by (subjective) Descriptions

QBE Query by Example

QBG Query by Gesture

QBIE Query by Image Example

QBIC Query by Image Content

QBPE Query by Pictorial Example

QBS Query by Sketch

QBSC Query by Scenario

QiPE Query by internal Pictorial Example

QPNE Query by Positive and Negative Examples

QPE Query by Pictorial Example

QVE Query by Visual Example

RAID Redundant Array of Independent Disks

RAM Random Access Memory

RDF Resource Description Framework

RF Relevance Feedback

RGB Red, Green Blue, color space

RIC Retrieve Image by Content

RJ Relevance Judgment

ROI Region of Interest

ROM Read Only Memory

SAM Spatial Access Method

SMIL Synchronized Multimedia Integration Language

SOM Self Organizing Map

SPEC Standard Performance Evaluation Corporation

SQL Structured Query Language

SSL Secure Socket Layer

STAR System for Trademark Archival and Retrieval

SVG Scalable Vector Graphics

SWIC Search Watermark Images by Content

TR Text Retrieval

TREC Text REtrieval Conference

TSR Télévision Suisse Romande

UDC Universal Decimal Classification

URI Uniform Resource Identifier

URL Uniform Resource Locator

VIPER Visual Information Processing for Enhanced Retrieval

VIE Virage Image Engine

VQ Vector Quantization

VVE Virage Video Engine

WIPO World Intellectual Property Organization

W3 World Wide Web

W3C World Wide Web Consortium

WWW World Wide Web

XM eXperimental Model

XML eXtensible Markup Language

XSL eXtensible Stylesheet Language

List of Figures

List of Tables

Bibliography

[1] *Proceedings of the ACM Multimedia Workshop on Multimedia Information Retrieval (ACM MIR 2001)*, Ottawa, Canada, October 2001. The Association for Computing Machinery.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, Washington DC, USA, May 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, September 12–15 1994.

[4] P. Aigrain, H. Zhang, and D. Petkovic. Content–based presentation and retrieval of visual media: A state–of–the–art review. *International Journal on Multimedia Tools and Applications*, 3:179–202, 1996.

[5] S. Aksoy and R. M. Haralick. Graph theoretic clustering for image grouping and retrieval. In CVPR'99 [103], pages 63–68.

[6] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C.-F. Shu. The Virage image search engine: An open framework for image management. In I. K. Sethi and R. C. Jain, editors, *Storage & Retrieval for Image and Video Databases IV*, volume 2670 of *IS&T/SPIE Proceedings*, pages 76–87, San Jose, CA, USA, March 1996.

[7] M. Barbieri, G. Mekenkamp, M. Ceccarelli, and Nesvadba. The color browser: A content-driven linear video browsing tool. In ICME'2001 [104], pages 808–811.

[8] M. Beigi, A. B. Benitez, and S.-F. Chang. Metaseek: A content–based meta-search engine for images. In *Symposium on Electronic Imaging: Multimedia Processing and Applications - Storage and Retrieval for Image and Video Databases VI, IST/SPIE'98, San Jose, CA*, pages 118–128, 1998.

[9] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color– and texture–based image segmentation using EM and its application to content–based image retrieval. In *Proceedings of the International Conference on Computer Vision (ICCV'98)*, pages 675–682, Bombay, India, January 1998.

[10] A. B. Benitez, M. Beigi, and S.-F. Chang. Using relevance feedback in content–based image metasearch. *IEEE Internet Computing*, July/August:59–69, 1998.

[11] 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*, 9(1):56–75, 2000.

[12] G. Beretta and R. Schettini, editors. *Internet Imaging III*, volume 4672 of *SPIE Proceedings*, San Jose, California, USA, January 21–22 2002. (SPIE Photonics West Conference).

[13] S. Beretti, A. Del Bimbo, and P. Pala. content–based retrieval of 3D cellular structures. In ICME'2001 [104], pages 1096–1099.

[14] A. P. Berman and L. G. Shapiro. Efficient content–based retrieval: Experimental results. In CBAIVL99 [28], pages 55–61.

[15] T. Berners-Lee and J. Hendler. Scientific publishing on the "semantic web". *Nature*, 410:1023–1024, 2001.

[16] I. Biedermann. Recognition–by–components: A theory of human image understanding. *Psychological Review*, 94 No 2:115–147, 1987.

[17] A. Blaser, editor. *Proceedings of the Conference on Database Techniques for Pictorial Applications*, volume 81. Lecture Notes in Computer Science, 1979.

[18] M. Bober. MPEG–7 visual shape descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):716–719, 2001.

[19] P. Borlund and P. Ingwersen. The development of a method for the evaluation of interactive information retrieval systems. *Journal of Documentation*, 53:225–250, 1997.

[20] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM (1)*, pages 126–134, 1999.

[21] C. Brodley, A. Kak, C. Shyu, J. Dy, L. Broderick, and A. M. Aisen. Content–based retrieval from medical image databases: A synergy of human interaction, machine learning and computer vision. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 760–767, 1999.

[22] R. Brunelli and O. Mich. Histogram analysis for image retrieval. *Pattern Recognition*, 34:1625–1637, 2001.

[23] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[24] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region–based image querying. In CVPR'97 [102], pages 42–51.

[25] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Color- and texture–based segmentation using EM and its application to image querying and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002 (to appear).

[26] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region–based image indexing and retrieval. In Huijsmans and Smeulders [99], pages 509–516.

[27] V. Castelli and L. D. Bergmann. *Image Databases – Search and Retrieval of Digital Imagery.* Wiley Interscience, 2002.

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

[29] K. Chakrabarti, K. Porkaew, and S. Mehrotra. Efficient query refinement in multimedia databases. In *Proceedings of the 16th International Conference on Data Engineering (ICDE2000)*, San Diego, CA, USA, March 1–3 2000. IEEE Computer Society.

[30] D. Y.-M. Chan and I. King. Genetic algorithm for weights assignment in dissimilarity function for trademark retrieval. In Huijsmans and Smeulders [99], pages 557–565.

[31] C. Chang and S. Chatterjee. Ranging through Gabor logons, a consistent, hierarchical approach. *IEEE Transactions on Neural Networks*, 4(5):827–843, September 1993.

[32] E. Chang, K.-T. Cheng, W.-C. Lai, C.-T. Wu, C. Chang, and Y.-L. Wu. PBIR: Perception–based image retrieval – a system that learns subjective image query concepts. In *Proceedings of the 9th ACM International Conference on Multimedia (ACM MM 2001)*, pages 611–614, Ottawa, Canada, October 2001. The Association for Computing Machinery.

[33] E. Y. Chang and K.-T. Cheng. Supporting subjective image queries without seeding requirements – proposing test queries for Benchathlon. In Beretta and Schettini [12], pages 225–232. (SPIE Photonics West Conference).

[34] M.-T. Chang and S.-Y. Chen. Deformed trademark retrieval based on 2D pseudo-hidden markov model. *Pattern Recognition*, 34:953–967, 2001.

[35] N.-S. Chang and K.-S. Fu. Query–by–pictorial–example. *IEEE Transactions on Software Engineering*, SE 6 No 6:519–524, 1980.

[36] S.-K. Chang and T. Kunii. Pictorial data–base applications. *IEEE Computer*, 14(11):13–21, 1981.

[37] Y.-C. Chang, L. Bergmann, J. R. Smith, and C.-S. Li. Query taxonomy of multimedia databases. In Panchanathan et al. [201]. (SPIE Symposium on Voice, Video and Data Communications).

[38] J. Chiupricha. Integration of textual and visual features for content–based multimedia retrieval. Master's thesis, Computer Science and Software Engineering (CSSE), Monash University, Melbourne, Australia, 2001.

[39] G. Ciocca and R. Schettini. Content–based similarity retrieval of trademarks using relevance feedback. *Pattern Recognition*, 34:1639–1655, 2001.

[40] C. W. Cleverdon. Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. Technical report, Aslib Cranfield Research Project, Cranfield, USA, September 1962.

[41] C. W. Cleverdon, L. Mills, and M. Keen. Factors determining the performance of indexing systems. Technical report, ASLIB Cranfield Research Project, Cranfield, 1966.

[42] D. Comaniciu, P. Meer, D. Foran, and A. Medl. Bimodal system for interactive indexing and retrieval of pathology images. In *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision (WACV'98)*, pages 76–81, Princeton, NJ, USA, October 1998.

[43] D. Comaniciu, P. Meer, K. Xu, and D. Tyler. Retrieval performance improvement through low rank corrections. In CBAIVL99 [28], pages 50–54.

[44] A. S. Constantinidis, M. C. Fairhurst, and A. F. R. Rahman. A new multi–expert decision combination algorithm and its application to the detection of circumscribed masses in digital mammograms. *Pattern Recognition*, 34:1527–1537, 2001.

[45] J. M. Corridoni, A. Del Bimbo, and E. Vicario. Image retrieval by color semantics with incomplete knowledge. *Journal of the American Society for Information Science*, 49(3):267–282, 1998.

[46] G. Cortelazzo, G. A. Mian, G. Vezzi, and P. Zamperoni. Trademark shapes description by string–matching techniques. *Pattern Recognition*, 27 No 8:1005–1018, 1994.

[47] I. J. Cox, J. Ghosn, M. L. Miller, T. V. Papathomas, and P. N. Yianilos. Hidden annotation in content based image retrieval. In *IEEE Workshop on Content–based Access of Image and Video Libraries (CBAIVL'97)*, pages 76–81, June 1997.

[48] I. J. Cox, M. L. Miller, T. P. Minka, and P. N. Yianilos. An optimized interaction strategy for bayesian relevance feedback. In *Proceedings of the 1998 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 553–558, Santa Barbara, California, USA, June 1998.

[49] 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.

[50] W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, August 1998. ACM Press, New York.

[51] *Proceedings of the 1996 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, San Francisco, California, June 1996.

[52] J. Dahmen, M. O. Güld, H. Ney, and K. Beulen. A mixture density based approach to object recognition for image retrieval. In *Recherche d'Informations Assistée par Ordinateur (RIAO'2000) Computer–Assisted Information Retrieval*, volume 1, pages 1632–1647, Paris, France, April12–14 2000.

[53] J. G. Daugman. An information theoretic view of analog representation in striate cortex. *Computational Neuroscience*, 2:9–18, 1990.

[54] J. 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.

[55] N. Day and J. M. Martìnez. Introduction to MPEG–7 (V 3.0), Sydney, Australia. Doc. ISO/IEC JTC1/SC29/WG11 N4325, International Organisation for Standardization, July 2001.

[56] A. P. de Vries, M. G. L. M. van Doorn, H. M. Blanken, and P. M. G. Apers. The mirror MMDBMS architecture. In *Proceedings of 25th International Conference on Very Large Databases (VLDB'99)*, pages 758–761, Edinburgh, Scotland, September 1999.

[57] A. del Bimbo. *Visual Information Retrieval.* Academic Press, 1999.

[58] A. Dimai. *Scene Configuration Based Preattentive Similarity Assessments for Content Based Image Retrieval.* PhD thesis, Communications Technology Laboratory, Swiss Federal Institute of Technology, ETH, CH–8092 Zürich, Switzerland, February 1998.

[59] A. Dimai. Assessment of effectiveness of content–based image retrieval systems. In Huijsmans and Smeulders [99], pages 525–532.

[60] D. Doermann, E. Rivlin, and I. Weiss. Applying algebraic and differential invariants for logo recognition. *Machine Vision and Applications*, 9:73–86, 1996.

[61] D. Dori. Cognitive image retrieval. In Sanfeliu et al. [230], pages 42–45.

[62] L. Duan, W. Gao, and J. Ma. A rich get richer strategy for content–based image retrieval. In Laurini [135], pages 290–299.

[63] M. Dunlop. Reflections on MIRA: Interactive evaluation in information retrieval. *Journal of the American Society for Information Science*, 51 (14):1269–1275, 2000.

[64] J. G. Dy, C. E. Brodley, A. Kak, C.-R. Shyu, and L. S. Broderick. The customized–queries approach to CBIR using using EM. In CVPR'99 [103], pages 400–406.

[65] J. P. Eakins, J. M. Boardman, and K. Shields. Retrieval of trademark images by shape feature – the artisan project. Technical report, BL research and innovation report 26, Newcastle upon Tyne, England, 1996.

[66] J. P. Eakins and M. E. Graham. content–based image retrieval. Technical Report JTAP–039, JISC Technology Application Program, Newcastle upon Tyne, 2000.

[67] J. P. Eakins, B. J. M., and M. E. Graham. Similarity retrieval of trademark images. *IEEE Multimedia Magazine*, April:53–63, 1998.

[68] M. J. Egenhofer. Spatial–query–by–sketch. In *Proceedings of the IEEE Symposium on Visual Languages (VL 1996)*, pages 6—67, September 1996.

[69] P. G. B. Enser. Pictorial information retrieval. *Journal of Documentation*, 51(2):126–170, 1995.

[70] S. Flank. Multimedia IR in context. In ACMMIR2001 [1], page 55.

[71] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[72] M. Frydrych, J. Parkkinen, and A. Visa, editors. *The 10th Scandinavian Conference on Image Analysis (SCIA'97)*, Lappeenranta, Finland, June 1997. Pattern Recognition Society of Finland.

[73] U. Gargi and R. Kasturi. Image database querying using a multi–scale localized color representation. In CBAIVL99 [28], pages 28–32.

[74] J. Gauvain, L. Lamel, G. Adda, and D. Matrouf. Developments in continuous speech dictation using the 1995 ARPA NAB news task. In *Proceedings of ICASSP 1996*, pages 73–76, Atlanta, GA, 1996.

[75] Z. Geradts, H. Hardy, A. Poortmann, and J. Bijhold. Evaluation of contents based image retrieval methods for a database of logos on drug tablets. In E. M. Carapezza, editor, *Technologies for Law Enforcement*, volume 4232 of *SPIE Proceedings*, Boston, Massachusetts, USA, November 5–8 2000.

[76] J.-M. Geusebroek, R. van den Boogaard, A. W. M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001.

[77] T. Gevers. PicToSeek: A content–based image search system for the World Wide Web. In VISUAL97 [285].

[78] T. Gevers and A. W. M. Smeulders. $\sum$nigma: An image retrieval system. In *Proceedings of the 11th International Conference on Pattern Recognition (ICPR'92)*, pages 697–700, The Hague, Netherlands, 1992. IEEE.

[79] T. Gevers and A. W. M. Smeulders. A comparative study of several color models for color image invariant retrieval. In Smeulders and Jain [246], pages 17–26.

[80] T. Gevers and A. W. M. Smeulders. Evaluating color and shape invariant image indexing of consumer photography. In C. Leung, editor, *First International Conference On Visual Information Systems (VISUAL'96)*, number 1306 in Lecture Notes in Computer Science, Melbourne, Australia, February 1996. Springer–Verlag.

[81] J. J. Gibson. Ecological optics. *Vision Research*, 1:253–262, 1961.

[82] K.-S. Goh, E. Chang, and K.-T. Cheng. Support vector machine pairwise classifiers with error reduction for image classification. In ACMMIR2001 [1], pages 32–37.

[83] R. C. Gonzales and P. Wintz. *Digital Image Processing.* Addison-Wesley, Reading Massachussetts, 1987.

[84] M. E. Graham and J. P. Eakins. Artisan: A prototype retrieval system for trademark images. *Vine*, 107:73–80, 1998.

[85] W. R. Greiff. A theory of term weighting based on exploratory data analysis. In Croft et al. [50], pages 11–19.

[86] O.-J. Grüsser and U. Grüsser-Cornehls. Gesichtssinn und Okulomotorik. In *Physiologie des Menschen* [239], pages 278–315.

[87] V. N. Gudivada and V. V. Raghavan. Content–based image retrieval systems. *IEEE Computer*, 18:18–22, 1995.

[88] V. N. Gudivada and V. V. Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems*, 13 (2):115–144, 1995.

[89] N. J. Gunther and G. Beretta. A benchmark for image retrieval using distributed systems over the internet: BIRDS–I. Technical report, HP Labs, Palo Alto, Technical Report HPL–2000–162, San Jose, 2001.

[90] A. Hampapur, A. Gupta, B. Horowitz, C.-F. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain. Virage video engine. In I. K. Sethi and R. C. Jain, editors, *Storage and Retrieval for Image and Video Databases V*, volume 3022 of *SPIE Proceedings*, pages 352–360, February 1997.

[91] D. Harman. Overview of the first Text REtrieval Conference (TREC–1). In *Proceedings of the first Text REtrieval Conference (TREC–1)*, pages 1–20, Washington DC, USA, 1992.

[92] A. G. Hauptmann, M. J. Witbrock, and M. G. Christel. Artificial intelligence techniques in the interface to a digital video library. In *Proceedings of the Conference on Human Factors in Computing System (CHI 1997)*, pages 213–239, Atlanta, GA, USA, March 1997.

[93] Q. He. An evaluation on MARS – an image indexing and retrieval system. Technical report, Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA, 1997.

[94] M. Hearst, F. Gey, and R. Tong, editors. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, USA, August 1999. ACM Press, New York.

[95] D. O. Hebb. A neuropsychologigal theory. In *Psychology: A study of Science*, pages 622–643. McGraw–Hill, 1959.

[96] M. K. Hu. Visual pattern recognition by moment invariants. In *IEEE Transactions on Information Theory*, volume IT–8, pages 179–187, February 1962.

[97] B. Huet and E. R. Hancock. Inexact graph retrieval. In CBAIVL99 [28], pages 40–44.

[98] D. P. Huijsmans and N. Sebe. Extended performance graphs for cluster retrieval. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pages 26–31, Kauai, Hawaii, USA, December 9–14 2001. IEEE Computer Society.

[99] D. P. Huijsmans and A. W. M. Smeulders, editors. *Third International Conference On Visual Information Systems (VISUAL'99)*, number 1614 in Lecture Notes in Computer Science, Amsterdam, The Netherlands, June 2–4 1999. Springer–Verlag.

[100] W.-S. Hwang, J. J. Weng, M. Fang, and J. Qian. A fast image retrieval algorithm with automatically extracted discriminant features. In CBAIVL99 [28], pages 8–12.

[101] *Proceedings of the 14th International Conference on Pattern Recognition (ICPR'98)*, Brisbane, Australia, August 1998. IEEE.

[102] IEEE Computer Society. *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, San Juan, Puerto Rico, June 1997.

[103] IEEE Computer Society. *Proceedings of the 1999 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, Fort Collins, Colorado, USA, June 23–25 1999.

[104] IEEE Computer Society. *Proceedings of the second International Conference on Multimedia and Exposition (ICME'2001)*, Tokyo, Japan, August 2001. IEEE Computer Society.

[105] Q. Iqbal and J. K. Aggarwal. Applying perceptual grouping to content–based image retrieval: Building images. In CVPR'99 [103], pages 42–48.

[106] A. Jaimes, A. B. Benitez, S.-F. Chang, and C. Jörgensen. Experiments for multiple level classification of visual descriptors. Contribution ISO/IEC JTC1/SC29/WG11 MPEG99/M5593, International Organisation for Standardization, December 1999.

[107] A. Jaimes, C. Jörgensen, A. B. Benitez, and S.-F. Chang. A conceptual framework and empirical research for classifying visual descriptors. *Journal of the American Society for Information Science*, 52 No 11:938–947, 2001.

[108] A. Jain and G. Healey. A multiscale representation including opponent color features for texture recognition. *IEEE Transactions on Image Processing*, 7(1):124–128, January 1998.

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

[110] A. K. Jain and A. Vailaya. Shape–based retrieval: A case study with trademark images. *Pattern Recognition*, 31 No 9:1369–1390, 1998.

[111] M. Y. Jaisimha. Wavelet feature for similarity based retrieval of logo images. In L. M. Vincent and J. Hull, editors, *Document Recognition III*, volume 2660 of *SPIE Proceedings*, January 1996.

[112] F. Jing, B. Zhang, F. Lin, W.-Y. Ma, and H.-J. Zhang. A novel region–based image retrieval method using relevance feedback. In ACMMIR2001 [1], pages 28–31.

[113] C. Jörgensen. Classifying images: Criteria for grouping as revealed in a sorting task. In *Proceedings of the 6th ASIS SIG/CR Classification research Workshop*, pages 65–78, Chicago, Il, USA, October 1995.

[114] C. Jörgensen. Retrieving the unretrievable in electronic imaging systems: emotions, themes and stories. In B. Rogowitz and T. N. Pappas, editors, *Human Vision and Electronic Imaging IV*, volume 3644 of *SPIE Proceedings*, San Jose, California, USA, January 23–29 1999. (SPIE Photonics West Conference).

[115] C. Jörgensen and P. Jörgensen. Testing a vocabulary for image indexing and ground truthing. In Beretta and Schettini [12], pages 207–215. (SPIE Photonics West Conference).

[116] T. Kato. Database architecture for content–based image retrieval. In A. A. Jamberdino and W. Niblack, editors, *Image Storage and Retrieval Systems*, volume 1662 of *SPIE Proceedings*, pages 112–123, San Jose, California, February 1992.

[117] P. M. Kelly, M. Cannon, and D. R. Hush. Query by image example: the CANDID approach. In W. Niblack and R. C. Jain, editors, *Storage and Retrieval for Image and Video Databases III*, volume 2420 of *SPIE Proceedings*, pages 238–248, March 1995.

[118] Y.-S. Kim and W.-Y. Kim. Content–based trademark retrieval system using visually salient features. In CVPR'97 [102].

[119] B. Ko, J. Peng, and H. Byun. A new content–based image retrieval system using hand gesture and relevance feedback. In ICME'2001 [104], pages 501–504.

[120] J. J. Koenderink and A. J. van Doorn. Receptive field families. *Biological Cybernetics*, 63:291–297, 1990.

[121] K. Koffka. *Principles of Gestalt Psychology.* Lund Humphries, London, 1935.

[122] W. Köhler. An old pseudoproblem (Ein altes Scheinproblem). *Die Naturwissenschaften*, 17:395–401, 1929.

[123] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, pages 199–204, Vienna, Austria, February 1999. IOS Press.

[124] P. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast and effective retrieval of medical tumor shapes. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):889–904, 1998.

[125] M. Koskela, J. Laaksonen, S. Laakso, and E. Oja. Evaluating the performance of content–based image retrieval systems. In Laurini [135], pages 430–441.

[126] C.-C. J. Kuo, S.-F. Chang, and S. Panchanathan, editors. *Multimedia Storage and Archiving Systems III (VV02)*, volume 3527 of *SPIE Proceedings*, Boston, Massachusetts, USA, November 1998. (SPIE Symposium on Voice, Video and Data Communications).

[127] T. Kurita and T. Kato. Learning of personal visual impression for image database systems. In *Proceedings of the 4th Internation Conference on Document Analysis and Recognition (ICDAR 1993)*, pages 547–552, Tsukuba Science City, Japan, October 1993. IEEE.

[128] T. Kuyel and J. Ghosh. A fast space localized computation of the outputs of a Gabor filter bank. In *Proceedings of the IASTED Conference on Signal and Image Processing (SIP'95)*, pages 511–514, Las Vegas, USA, November 1995.

[129] P. W. H. Kwan, K. Kameyama, and K. Toraichi. Trademark retrieval by relaxation matching on fluency function approximated image contours. In *Proceedings of the 2001 Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM 2001)*, pages 255–258, Victoria, B.C., Canada, August 28–28 2001.

[130] T.-S. Lai and J. Tait. CHROMA: A content–based image retrieval system. In Hearst et al. [94], page 324.

[131] R. Lakmann and L. Priese. A reduced covariance color texture model for micro–textures. In Frydrych et al. [72], pages 947–953.

[132] P. M. Lam, J. K. Wu, and B. M. Mehtre. STAR – a system for trademark archival and retrieval. In *Proceedings of the Second Asian Conference on Computer Vision (ACCV 1995)*, pages 214–217, Singapore, December 5–8 1995.

[133] T. Lamb and J. Bourriau. *Colour: Art&Science*. The Darwin College Lectures, Cambridge University Press, Cambridge, 1995.

[134] M.-C. Larabi, N. Richard, and C. Fernandez-Maloigne. Using combination of color, texture and shape features for image retrieval in melanomas databases. In Beretta and Schettini [12], pages 147–156. (SPIE Photonics West Conference).

[135] R. Laurini, editor. *Fourth International Conference On Visual Information Systems (VISUAL'2000)*, number 1929 in Lecture Notes in Computer Science, Lyon, France, November 2000. Springer–Verlag.

[136] C. S. Lee, W.-Y. Ma, and H. Zhang. Information embedding based on user's relevance feedback in image retrieval. In Panchanathan et al. [201], pages 294–304. (SPIE Symposium on Voice, Video and Data Communications).

[137] C. Leung and H. Ip. Benchmarking for content–based visual information search. In Laurini [135], pages 442–456.

[138] M. Leyton. Principles of information structure common to six levels of the human cognitive system. *Information Sciences*, 38:1–120, 1986.

[139] B. Li, E. Chang, and C.-S. Li. Learning image query concepts via intelligent sampling. In ICME'2001 [104], pages 1168–1171.

[140] M. Li, Z. Chen, L. Wenyin, and H.-J. Zhang. A statistical correlation model for image retrieval. In ACMMIR2001 [1], pages 42–45.

[141] C.-T. Liu, P.-L. Tai, A. Y.-J. Chen, C.-H. Peng, T. Lee, and J.-S. Wang. A content–based CT lung retrieval system for assisting differential diagnosis images collection. In ICME'2001 [104], pages 241–244.

[142] B. Logan and A. Salomon. A music similarity function based on signal analysis. In ICME'2001 [104], pages 952–955.

[143] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31 No 8:983–1001, 1998.

[144] D. G. Lowe. Three–dimensional object recognition from single two–dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

[145] C.-S. Lu and P.-C. Chung. Wold features for unsupervised texture segmentation. In ICPR'98 [101], pages 1689–1693.

[146] C. Luoni. Development of an interface to a database storing the features of a multimedia retrieval system. License thesis (BSc),Computer Vision and Multimedia Laboratory at the University of Geneva, 2000.

[147] J. Lyons. Colour in language. In *Colour: Art&Science* [133], pages 194–224.

[148] W. Ma and B. Manjunath. Texture features and learning similarity. In CVPR'96 [51], pages 425–430.

[149] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715, 2001.

[150] S. Marchand-Maillet. Content-based video retrieval: An overview. Technical Report 00.06, CUI - University of Geneva, Geneva, Switzerland, 2000.

[151] M. Markkula and E. Sormunen. Searching for photos – journalists' practices in pictorial IR. In J. P. Eakins, D. J. Harper, and J. Jose, editors, *The Challenge of Image Retrieval, A Workshop and Symposium on Image Retrieval*, Electronic Workshops in Computing, Newcastle upon Tyne, 5–6 February 1998. The British Computer Society.

[152] M. Markkula and E. Sormunen. End–user searching challenges indexing practices in the digital photo archive. *Information Retrieval*, 1(4), 2000.

[153] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings Royal Society of London*, 207:187–217, 1980.

[154] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV2001)*, Vancouver, Canada, July 2001.

[155] A. Martinez. Face image retrieval using HMMs. In CBAIVL99 [28], pages 35–39.

[156] J. Mauro. Oracle intermedia: Managing multimedia content. An Oracle White Paper, 2001.

[157] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of the 2nd ACM International Conference on Digital Libraries (ACMDL96)*, pages 11–18, Bethesda, MD, USA, 1996. Association for Computing Machinery.

[158] M. T. Meharga. *An integrated Approch for Querying General–Purpose Image Databases.* Ph.D. Dissertation, École Polytechnique Fédérale de Lausanne, Switzerland, October 2001.

[159] R. Milanese and M. Cherbuliez. A rotation, translation and scale–invariant approach to content–based image retrieval. *Journal of Visual Communication and Image Representation*, 10:186–196, 1999.

[160] R. Milanese, M. Cherbuliez, and T. Pun. Invariant content–based image retrieval using the Fourier–Mellin transform. In S. Singh, editor, *International Conference on Advances in Pattern Recognition (ICAPR'98)*, pages 73–82. Springer–Verlag, 1999. (Plymouth, UK, 23–25 November 1998).

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

[162] T. P. Minka and R. W. Picard. Interactive learning using a "society of models". In CVPR'96 [51], pages 447–452.

[163] S. Mizzaro. Relevance: The whole (hi)story. *Journal of the American Society for Information Science*, 48 No 9:810–832, 1997.

[164] B. Moghaddam, Q. Tian, and T. S. Huang. Spatial visualization for content–based image retrieval. In ICME'2001 [104], pages 229–232.

[165] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In Smeulders and Jain [246], pages 35–42.

[166] MPEG Requirements Group. MPEG–7: Context and objectives (version 10 Atlantic City). Doc. ISO/IEC JTC1/SC29/WG11, International Organisation for Standardization, October 1998.

[167] S. Mukherjea, K. Hirata, and Y. Hara. AMORE: A world wide web image retrieval engine. In *Proceedings of the Conference on Human Factors in Computing System (CHI 1999)*, pages 17–18, Pittsburgh, PA, USA, May 1999.

[168] H. Müller. Suchen ohne Worte – wie inhaltsbasierte Suche funktioniert. *ct Magazin für Computertechnik*, 15:162–172, 2001.

[169] H. Müller. Jäger des verlorenen Fotos – Das GNU Image Finding Tool in der Praxis. *ct Magazin für Computertechnik*, 6:252–257, 2002.

[170] H. Müller, S. Marchand-Maillet, and T. Pun. The truth about corel – evaluation in image retrieval. In *Proceedings of the International Conference on the Challenge of Image and Video Retrieval (CIVR 2002)*, July 2002 (to appear).

[171] H. Müller, W. Müller, S. Marchand-Maillet, D. M. Squire, and T. Pun. Long term learning in content–based image retrieval. Technical Report 00.04, Computer Vision Group, Computing Centre, University of Geneva, rue Général Dufour, 24, CH–1211 Genève, Switzerland, February 2000.

[172] H. Müller, W. Müller, S. Marchand-Maillet, D. M. Squire, and T. Pun. Automated benchmarking in content–based image retrieval. In ICME'2001 [104], pages 321–324.

[173] H. Müller, W. Müller, S. Marchand-Maillet, D. M. Squire, and T. Pun. A web–based evaluation system for content–based image retrieval. In ACMMIR2001 [1], pages 50–54.

[174] H. Müller, W. Müller, S. Marchand-Maillet, D. M. Squire, and T. Pun. A framework for benchmarking in visual information retrieval. *International Journal on Multimedia Tools and Applications*, 2002 (to appear). (Special Issue on Multimedia Information Retrieval).

[175] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun. Learning feature weights from user behavior in content–based image retrieval. In S. Simoff and O. Zaiane, editors, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Workshop on Multimedia Data Mining MDM/KDD2000)*, pages 67–72, Boston, MA, USA, August20-23 2000.

[176] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun. Lernen von Merkmalsgewichtungen beim inhaltsbasierten Suchen in grossen Bilddatenbanken (Content–Based Image Retrieval). In *Informatiktage 2000*, pages 304–307, Bad Schussenried, Germany, 27–28 October 2000.

[177] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun. Strategies for positive and negative relevance feedback in image retrieval. In Sanfeliu et al. [230], pages 1043–1046.

[178] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, and T. Pun. Performance evaluation in content–based image retrieval: Overview and proposals. *Pattern Recognition Letters*, 22(5):593–601, April 2001.

[179] H. Müller, W. Müller, D. M. Squire, Z. Pečenović, S. Marchand-Maillet, and T. Pun. An open framework for distributed multimedia retrieval. In *Recherche d'Informations Assistée par Ordinateur (RIAO'2000) Computer–Assisted Information Retrieval*, volume 1, pages 701–712., Paris, France, April12–14 2000.

[180] H. Müller, D. M. Squire, W. Müller, and T. Pun. Content–Based Image Retrieval, bildinhaltsbasiertes Suchen in großen Bilddatenbanken. In *Informatiktage 1999*, pages 107–109, Bad Schussenried, Germany, 12–13 November 1999.

[181] 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. [201], pages 461–472. (SPIE Symposium on Voice, Video and Data Communications).

[182] H. Müller, D. M. Squire, and T. Pun. Learning from user behavior in image retrieval: Application of the market basket analysis. *International Journal of Computer Vision*, 2002 (submitted). (Special Issue on Content–Based Image Retrieval).

[183] S. Müller and G. Rigoll. Improved stochastic modeling of shapes for content–based image retrieval. In CBAIVL99 [28], pages 23–27.

[184] W. Müller. *Design and implementation of a flexible Content–Based Image Retrieval Framework - The GNU Image Finding Tool.* PhD thesis, Computer Vision and Multimedia Laboratory, University of Geneva, Geneva, Switzerland, September 2001.

[185] W. Müller, S. Marchand-Maillet, H. Müller, and T. Pun. Towards a fair benchmark for image browsers. In *SPIE Photonics East, Voice, Video, and Data Communications*, pages 262–271, Boston, MA, USA, November 5–8 2000.

[186] W. Müller, S. Marchand-Maillet, H. Müller, D. M. Squire, and T. Pun. Evaluating image browsers using structured annotation. *Journal of the American Society for Information Science and Technology (JASIST)*, 52(11):961–968, 2001.

[187] W. Müller, H. Müller, S. Marchand-Maillet, T. Pun, D. M. Squire, Z. Pečenović, C. Giess, and A. P. de Vries. MRML: A communication protocol for content–based image retrieval. In *International Conference on Visual Information Systems (Visual 2000)*, pages 300–311, Lyon, France, November 2–4 2000.

[188] W. Müller, Z. Pečenović, A. P. de Vries, D. M. Squire, H. Müller, and T. Pun. MRML: Towards an extensible standard for multimedia querying and benchmarking – Draft proposal. Technical Report 99.04, Computer Vision Group, Computing Centre, University of Geneva, rue Général Dufour, 24, CH–1211 Genève, Switzerland, October 1999.

[189] W. Müller, Z. Pečenović, H. Müller, S. Marchand-Maillet, T. Pun, D. M. Squire, A. P. D. Vries, and C. Giess. MRML: An extensible communication protocol for interoperability and benchmarking of multimedia information retrieval systems. In *SPIE Photonics East - Voice, Video, and Data Communications*, pages 961–968, Boston, MA, USA, November 5–8 2000.

[190] 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. [201], pages 328–337. (SPIE Symposium on Voice, Video and Data Communications).

[191] M. Nakazato and T. S. Huang. 3D MARS: Immersive virtual reality for content–based image retrieval. In ICME'2001 [104], pages 45–48.

[192] A. D. Narasimhalu, M. S. Kankanhalli, and J. Wu. Benchmarking multimedia databases. *Multimedia Tools and Applications*, 4:333–356, 1997.

[193] W. Niblack, R. Barber, W. Equitz, M. D. Flickner, E. H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. QBIC project: querying images by content, using color, texture, and shape. In W. Niblack, editor, *Storage and Retrieval for Image and Video Databases*, volume 1908 of *SPIE Proceedings*, pages 173–187, April 1993.

[194] J. Nielsen. *Usability Engineering.* Academic Press, Boston, MA, 1993.

[195] M. R. Ogiela and R. Tadeusiewicz. Sematic-oriented syntactic algorithms for content recognition and understanding of images in medical databases. In ICME'2001 [104], pages 621–624.

[196] M. Ortega, K. P. Chakrabarti, and S. Mehrotra. Cross media validation in a multimedia retrieval system. In *Proceedings of the Workshop on Digital Library Metrics at the ACM International Conference on Digital Libraries (ACMDLM'1998)*, Pittsburgh, PA, June 1998. Association for Computing Machinery, ACMPress.

[197] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang. Supporting similarity queries in MARS. In *Proceedings of The Fifth ACM International Multimedia Conference (ACM Multimedia 97)*, pages 403–413, Seattle, WA, USA, November 9–13 1997.

[198] M. Ortega, Y. Rui, K. Chakrabarti, K. Porkaew, S. Mehrotra, and T. S. Huang. Supporting ranked boolean similarity queries in MARS. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):905–925, December 1998.

[199] M. Ortega-Binderberger, S. Mehrotra, K. Chakrabarti, and K. Porkaew. WebMARS: A multimedia search engine. In G. Beretta and J. J. McCann, editors, *Internet Imaging*, volume 3963 of *SPIE Proceedings*, pages 216–224, San Jose, California, USA, January 23–28 2000. (SPIE Photonics West Conference).

[200] B. Ozer, W. Wolf, and A. N. Akansu. A graph based object description for information retrieval in digital image and video libraries. In CBAIVL99 [28], pages 79–83.

[201] 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).

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

[203] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Tools for content–based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, June 1996.

[204] Z. Pečenović. Image retrieval using latent semantic indexing. Final year graduate thesis, AudioVisual Communications Lab, Ecole Polytechnique Fédérale de Lausanne, Switzerland, June 1997.

[205] Z. Pečenović, M. Do, S. Ayer, and M. Vetterli. New methods for image retrieval. In *Proceedings of the International Congress on Imaging Science*, volume 2, pages 242–246, University of Antwerp, Belgium, September 1998.

[206] T. Pfund and S. Marchand-Maillet. Dynamic multimedia annotation tool. In Beretta and Schettini [12], pages 216–224. (SPIE Photonics West Conference).

[207] J. H. Piater and R. A. Grupen. Toward learning visual discriminant strategies. In CVPR'99 [103], pages 410–415.

[208] J. H. Piater and R. A. Grupen. Feature learning for recognition with bayesian networks. In Sanfeliu et al. [230], pages 17–20.

[209] J. Preece. *Human–Computer Interaction*. Addison-Wesley, Harlow, England, 1994.

[210] T. Pun and D. M. Squire. Statistical structuring of pictorial databases for content–based image retrieval systems. *Pattern Recognition Letters*, 17:1299–1310, 1996.

[211] F. Qian, M. Li, W.-Y. Ma, F. Ling, and B. Zhang. Alternating features spaces in relevance feedback. In ACMMIR2001 [1], pages 14–17.

[212] E. M. Rasmussen. Indexing images. *Annual Review of Information Science and Technology*, 32:169–196, 1997.

[213] A. L. Ratan, O. Maron, W. E. L. Grimson, and T. Lozano-Perez. A framework for learning query concepts in image classification. In CVPR'99 [103], pages 423–429.

[214] S. Ravela, R. Manmatha, and E. M. Riseman. Image retrieval using scale space matching. In B. Buxton and R. Cippola, editors, *4th European Conference on Computer Vision (ECCV1996)*, number 1064 in Lecture Notes in Computer Science, pages 273–282, Cambridge, UK, April 15–18 1996. Springer–Verlag.

[215] V. Rehrmann and L. Priese. Fast and robust segmentation of natural color scenes. In *Proceedings of the 3rd Asian Conference on Computer Vision (ACCV'98)*, pages 598–606, Hong Kong, January 1998.

[216] B. Ribeiro-Net, E. S. Moura, M. S. Neubert, and N. Ziviani. Efficient distributed algorithms to build inverted files. In Hearst et al. [94], pages 105–112.

[217] A. R. Robertson. Historical development of CIE recommended color difference equations. *COLOR research and applications*, 15(3):167–170, 1990.

[218] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System, Experiments in Automatic Document Processing* [226], pages 313–323.

[219] P. Roth, D. Richoz, L. Petrucci, and T. Pun. An audio–haptic tool for non–visual image representation. In M. Deriche, N. Shaik-Husin, and I. Kamisian, editors, *The 6th International Symposium on Signal Processing and its applications (ISSPA 2001)*, August 2001.

[220] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Past, present and future. In M. Liao, editor, *Proceedings of the International Symposium on Multimedia Information Processing*, Taipei, Taiwan, December 1997.

[221] Y. Rui, T. S. Huang, and S. Mehrotra. Relevance feedback techniques in interactive content–based image retrieval. In I. K. Sethi and R. C. Jain, editors, *Storage and Retrieval for Image and Video Databases VI*, volume 3312 of *SPIE Proceedings*, pages 25–36, December 1997.

[222] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for 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).

[223] M. E. Ruiz and P. Srinivasan. Cross–language information retrieval: An analysis of errors. In C. M. Preston, editor, *Proceedings of the 61st annual meeting of the American Society for Information Science (ASIS 1998)*, pages 153–165, Pittsburgh, PA, USA, October 1998.

[224] P. S. Salembier and B. S. Manjunath. Audiovisual content description and retrieval: Tools and MPEG–7 standardization techniques. In *IEEE International Conference on Image Processing (ICIP 2000)*, Vancouver, BC, Canada, December 2000.

[225] G. Salton. Evaluation parameters. In *The SMART Retrieval System, Experiments in Automatic Document Processing* [226], pages 55–112.

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

[227] G. Salton. The state of retrieval system evaluation. Technical Report 91–1206, Department of Computer Science, Cornell University, Ithaca, New York 14853–7501, May 1991.

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

[229] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–287, 1990.

[230] A. Sanfeliu, J. J. Villanueva, M. Vanrell, R. Alcézar, J.-O. Eklundh, and Y. Aloimonos, editors. *Proceedings of the 15th International Conference on Pattern Recognition (ICPR 2000)*, Barcelona, Spain, September 2000. IEEE.

[231] S. Santini and R. Jain. Gabor space and the development of preattentive similarity. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96)*, pages 40–44, Vienna, Austria, August 1996. IEEE.

[232] S. Santini and R. Jain. Similarity queries in image databases. In CVPR'96 [51], pages 646–651.

[233] S. Santini and R. Jain. Beyond query by example. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pages 345–350, Los Angeles, CA, USA, December 1998.

[234] S. Santini and R. Jain. Direct manipulation of image databases. Technical report, Department of Computer Science, University of California San Diego, San Diego California, November 1998.

[235] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[236] T. Saracevis. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, November/December:321–343, 1975.

[237] L. Schamber, M. B. Eisenberg, and M. S. Nilan. A re–examination of relevance: Toward a dynamic, situational definition. *Information Processing and Management*, 26 No 6:755–775, 1990.

[238] C. Schmid. A structured probabilistic model for recognition. In CVPR'99 [103], pages 485–490.

[239] R. F. Schmidt and G. Thews. *Physiologie des Menschen.* Springer–Verlag, Heidelberg, 26 edition, 1995.

[240] K. A. Schroeder. Layered indexing of images. *The Indexer*, 21, No 1:11–14, 1998.

[241] S. Sclaroff, M. La Cascia, and S. Sethi. Unifying textual and visual cues for content–based image retrieval on the world wide web. *Computer Vision and Image Understanding*, 75 Nos 1/2:86–98, 1999.

[242] S. Sclaroff, L. Taycher, and M. 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.

[243] S. Shatford Layne. Some issues in the indexing of images. *Journal of the American Society for Information Science*, 45(8):583–588, 1994.

[244] C.-R. Shyu, C. E. Brodley, A. C. Kak, A. Kosaka, A. M. Aisen, and L. S. Broderick. ASSERT: A physician in the loop content–based retrieval system for HRCT image databases. *Computer Vision and Image Understanding*, 75 Nos 1/2:111–132, 1999.

[245] C.-R. Shyu, A. Kak, C. Brodley, and L. S. Broderick. Testing for human perceptual categories in a physician–in–the–loop CBIR system for medical imagery. In CBAIVL99 [28], pages 102–108.

[246] A. W. M. Smeulders and R. Jain, editors. *Image Databases and Multi–Media Search*, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, August 1996. Intelligent Sensory Information Systems, Faculty of Mathematics, Computer Science, Physics and Astronomy, Amsterdam University Press.

[247] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content–based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 No 12:1349–1380, 2000.

[248] J. R. Smith. *Integrated Spacial and Feature Image Systems: Retrieval, Compression and Analysis.* PhD thesis, Graduate School of Arts and Sciences, Columbia University, 2960 Broadway, New York, NY, USA, 1997.

[249] J. R. Smith. Image retrieval evaluation. In *IEEE Workshop on Content–based Access of Image and Video Libraries (CBAIVL'98)*, pages 112–113, Santa Barbara, CA, USA, June 21 1998.

[250] J. R. Smith and S.-F. Chang. Tools and techniques for color image retrieval. In I. K. Sethi and R. 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.

[251] 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.

[252] J. R. Smith and S.-F. Chang. Querying by color regions using the *VisualSEEk* content–based visual query system. In M. T. Maybury, editor, *Proceedings of the IJCAI Workshop on Intelligent Multimedia Information Retrieval*, 1997.

[253] W. W. S. So, C. H. C. Leung, and Z. J. Zheng. Search space reduction techniques for image databases. In *Proceedings of the First International Workshop on Image Databases and Multimedia Search (IDB-MMS'1996)*, pages 179–186, Amsterdam, The Netherlands, August 1996.

[254] A. Soffer and H. Samet. Using negative shape features for logo similarity matching. In ICPR'98 [101], pages 571–573.

[255] K. Sparck Jones. *Information Retrieval Experiment.* Butterworths, London, 1981.

[256] K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an ideal information retrieval test collection. British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

[257] K. Sparck Jones and C. J. Van Rijsbergen. Progress in documentation. *Journal of Documentation*, 32:59–75, 1976.

[258] D. M. Squire. General methods for invariant pattern recognition. In *Model–based Neural Networks for Invariant Pattern Recognition*, chapter 2, pages 8–33. School of Computing, Curtin University of Technology, Perth, Western Australia, October 1996.

[259] D. M. Squire. Generalization performance of factor analysis techniques used for image database organization. Technical Report 98.01, Computer Vision Group, Computing Centre, University of Geneva, rue Général Dufour, 24, CH–1211 Genève, Switzerland, January 1998.

[260] 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 [28], pages 45–49.

[261] D. M. Squire, H. Müller, W. Müller, S. Marchand-Maillet, and T. Pun. *Design & Management of Multimedia Information Systems: Opportunities & Challenges*, chapter 7, pages 125–151. Idea Group Publishing, London, 2001.

[262] D. M. Squire, W. Müller, and H. Müller. Relevance feedback and term weighting schemes for content–based image retrieval. In Huijsmans and Smeulders [99], pages 549–556.

[263] D. M. Squire, W. Müller, H. Müller, and T. Pun. content–based query of image databases: inspirations from text retrieval. *Pattern Recognition Letters (Selected Papers from The 11th Scandinavian Conference on Image Analysis SCIA '99)*, 21(13-14):1193–1198, 2000. B.K. Ersboll, P. Johansen, Eds.

[264] 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.

[265] D. M. Squire and T. Pun. A comparison of human and machine assessments of image similarity for the organization of image databases. In Frydrych et al. [72], pages 51–58.

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

[267] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. Wavelets for computer graphics: A primer. *IEEE Computer Graphics and Applications*, pages 76–84, May 1995.

[268] P. Suda, C. Bridoux, B. Kämmerer, and G. Maderlechner. Logo and word matching using a general approach to signal registration. In *Proceedings of the 4th Internation Conference on Document Analysis and Recognition (ICDAR 1997)*, Ulm, Germany, August 1997. IEEE.

[269] A. G. Sutcliffe, M. Ennis, and J. Hu. Evaluating the effectiveness of visual user interfaces for information retrieval. *The International Journal of Human-Computer Studies*, 53:741–763, 2000.

[270] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[271] H. D. Tagare, C. Jaffe, and J. Duncan. Medical image databases: A content–based retrieval approach. *Journal of the American Medical Informatics Association*, 4(3):184–198, 1997.

[272] H. Tamura and N. Yokoya. Image database systems: A survey. *Pattern Recognition*, 17 No 1:29–43, 1983.

[273] L. Taycher, M. La Cascia, and S. Sclaroff. Image digestion and relevance feedback in the ImageRover WWW search engine. In VISUAL97 [285], pages 85–94.

[274] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.

[275] N. Vasconcelos and A. Lippman. Probabilistic retrieval: New insights and experimental results. In CBAIVL99 [28], pages 62–66.

[276] N. Vasconcelos and A. Lippman. Feature representations for image retrieval: Beyond the the color histogram. In *Proceedings of the first International Conference on Multimedia and Exposition (ICME'2000)*, pages 899–901, New York, NY, USA, August 2000. IEEE.

[277] N. Vasconcelos and A. Lippman. Learning over multiple temporal scales in image databases. In D. Vernon, editor, *6th European Conference on Computer Vision (ECCV2000)*, number 1842 in Lecture Notes in Computer Science, pages 33–47, Dublin, Ireland, June 26–30 2000. Springer–Verlag.

[278] N. Vasconcelos and A. Lippman. A probabilistic architecture for content–based image retrieval. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pages 216–221, Hilton Head Island, South Carolina, USA, June 13–15 2000. IEEE Computer Society.

[279] N. Vasconcelos and A. Lippmann. A unifying view of image similarity. In Sanfeliu et al. [230], pages 1–4.

[280] A. Vellaikal and C.-C. J. Kuo. Hierarchical clustering techniques for image database organization and summarization. In Kuo et al. [126], pages 68–79. (SPIE Symposium on Voice, Video and Data Communications).

[281] J. Vendrig, M. Worring, and A. W. M. Smeulders. Filter image browsing: Exploiting interaction in image retrieval. In Huijsmans and Smeulders [99], pages 147–154.

[282] C. C. Venters and M. Cooper. content–based image retrieval. Technical Report JTAP–054, JISC Technology Application Program, 2000.

[283] M. Vetterli and J. Kovačević. *Wavelets and sub–band coding*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[284] M. Vissac, J.-L. Dugelay, and K. Rose. Indexation d'images par fractale/viterbi. In *5èmes Journées d'Études et d'Échanges COmpression et REprésentation des Signaux Audiovisuels (CORESA 1999)*, Sophia Antipolis, France, June 14–15 1999.

[285] *Second International Conference On Visual Information Systems (VISUAL'97)*, San Diego, CA, USA, December 1997.

[286] E. M. Voorhees. Variations in relevance judgements and the measurement of retrieval effectiveness. *Information Processing and Management*, 36:697–716, 2000.

[287] E. M. Voorhees and D. Harmann. Overview of the seventh Text REtrieval Conference (TREC–7). In *The Seventh Text Retrieval Conference*, pages 1–23, Gaithersburg, MD, USA, November 1998.

[288] E. M. Voorhees and D. Harmann. Overview of the ninth Text REtrieval Conference (TREC–9). In *The Ninth Text Retrieval Conference*, pages 1–13, Gaithersburg, MD, USA, November 2000.

[289] G. Wald. The receptors of human color vision. *Science*, 145:1007–1016, 1964.

[290] X. Wan, Z. Yang, and C.-C. J. Kuo. Efficient interactive image retrieval with multiple seed images. In Kuo et al. [126], pages 13–24. (SPIE Symposium on Voice, Video and Data Communications).

[291] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics–sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 No 9:1–17, 2001.

[292] Y.-H. Wang and T.-L. Chien. A new spatial relation model for image indexing and similarity retrieval. In ICME'2001 [104], pages 1164–1167.

[293] T. P. Weldon and W. E. Higgins. Integrated approach to texture segmentation using multiple Gabor filters. In P. Delogne, editor, *IEEE International Conference on Image Processing (ICIP'96)*, pages 955–958, Lausanne, Switzerland, September 1996.

[294] T. Westerveld. Image retrieval: Content versus context. In *Recherche d'Informations Assistée par Ordinateur (RIAO'2000) Computer–Assisted Information Retrieval*, volume 1, pages 276–284, Paris, France, April12–14 2000.

[295] J. S. Weszka, C. R. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man and Cybernetics*, 6(4):269–285, 1976.

[296] D. A. White and R. 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.

[297] L. T. Wilkins. *Social deviance.* Tavistock, London, 1965.

[298] A. Winter and C. Nastar. Differential feature distribution maps for image segmentation and region queries in image databases. In CBAIVL99 [28], pages 9–17.

[299] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and indexing documents and images.* Van Nostrand Reinhold, 115 Fifth Avenue, New York, NY 10003, USA, 1994.

[300] S. K. M. Wong, Y. Y. Yao, G. Salton, and C. Buckley. Formalization and evaluation of linear relevance feedback. Technical Report 89–992, Department of Computer Science, Cornell University, Ithaca, New York 14853–7501, April 1989.

[301] M. E. Wood, N. W. Campbell, and B. T. Thomas. Iterative refinement by relevance feedback in content–based digital image retrieval. In *Proceedings of The Fifth ACM International Multimedia Conference (ACM Multimedia 98)*, pages 13–20, Bristol, UK, September 1998.

[302] M. Worring, A. W. M. Smeulders, and S. Santini. Interaction in content–based image retrieval: An evaluation of the state of the art. In Laurini [135], pages 26–36.

[303] J. K. Wu, B. M. Mehtre, Y. J. Gao, P. M. Lam, and A. D. Narasimhalu. STAR – a multimedia database system for trademark recognition. In W. Litwin and T. Risch, editors, *Proceedings of the First International Conference on Applications of Databases (ADB 1994)*, number 819 in Lecture Notes in Computer Science, pages 109–122, Vadstena, Sweden, June 1994. Springer–Verlag.

[304] 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):89–105, 1998.

[305] P. Wu, B. S. Manjunath, S. D. Newsam, and H. D. Shin. A novel texture descriptor for image retrieval and browsing. In CBAIVL99 [28], pages 3–7.

[306] K. Yanai. Image collector: An image gathering system from the world-wide web employing keyword–based search engines. In ICME'2001 [104], pages 704–707.

[307] H.-C. Yang. Shape–based image retrieval by spatial topology distances. In ACMMIR2001 [1], pages 38–41.

[308] J. Ze Wang, G. Wiederhold, O. Firschein, and S. Xin Wei. Wavelet–based image indexing techniques with partial sketch retrieval capability. In *Proceedings of the Fourth Forum on Research and Technology Advances in Digital Libraries*, pages 13–24, Washington D.C., May 1997.

[309] L. Zhu, C. Tang, A. Rao, and A. Zhang. Using thesaurus to model keyblock–based image retrieval. In ICME'2001 [104], pages 237–240.

[310] G. K. Zipf. *La Psychobiologie du langage.* Centre d'étude et de promotion de la lecture, 114, Champs-Élysées, 75008 Paris, 1974.

[311] A. Zisserman, D. A. Forsyth, J. L. Mundy, C. Rothwell, J. Liu, and N. Pillow. 3D object recognition using invariance. *Artificial Intelligence*, 78(1–2):239–288, 1995.

[312] M. M. Zloof. Query by example. In *Proceedings of the National Computer Conference 1975*, pages 431–438, Anaheim, CA, USA, March 1975.

[313] J. Zobel. How reliable are the results of large–scale information retrieval experiments? In Croft et al. [50], pages 307–314.

Index