Diplomarbeit im Fach Informatik
Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. H. Ney

Features for Image Retrieval

vorgelegt von:
Thomas Deselaers
Matrikelnummer 218894

Gutachter:
Prof. Dr.-Ing. H. Ney
Prof. Dr. T. Seidl

Betreuer:
Dipl.-Inform. D. Keysers

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, im Dezember 2003

Thomas Deselaers

Acknowledgements

The present work originates from my work as student researcher at the Chair of Computer Science VI of the RWTH Aachen University of Technology, where I have been a member of the image recognition group since July 2001. I would like to thank Prof. Dr.-Ing. Hermann Ney for the interesting possibilities at this department and the chance to attend the conference "25th Pattern Recognition Symposium" in Magdeburg.

Furthermore, I would like to thank Daniel Keysers for the helpful suggestions, discussions and assistances this work received. His ideas and suggestions were a great help.

I would like to thank Prof. Dr. Thomas Seidl as well, who kindly accepted to attend the work.

Also, I would like to thank the other student members of the image recognition group at the Chair of Computer Science VI – Christian Gollan, Tobias Kölsch, Philippe Dreuw, and Ilja Bezrukov – for many discussions, helpful tips and comments.

Also I would like to thank Diego Biurrun, Stefan Jacobs, and Arne Mauser for proof reading the manuscript.

And of course special thanks go to my parents for all the possibilities they gave me and are still giving to me and to my girl friend Daniela for supporting me while doing this work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The growing amount of digital images caused by the more and more ubiquitous presence of digital cameras and, as a result, the many images on the world wide web confronts the users with new problems. Images are a fundamental part of our daily communication. The German saying “Ein Bild sagt mehr als tausend Worte” (literally: “A picture says more than a thousand words.”) reflects this. The huge amount of pictures digitally available is not manageable by humans any more [Chen & Wang 02]. A person searching for a picture in a database of 100 images will probably find what she searches for quite fast by just viewing the images or small versions of the images (thumbnails). If a thousand, ten thousand, or even more images are involved, the task becomes boring and interminable [Markkula & Sormunen 98]. Computers might be able to help here in the same way as they already do for searching text documents. A well-established example for text retrieval is the Internet search engine Google. Entering some keywords often helps finding related documents from the vast amount of documents available on the Internet. Google also offers a possibility to search for images, but the way the search is performed does not always lead to satisfactory results.

One way to search in image databases is to create a textual description of all the images in the database and use the methods from text-based information retrieval to search based on the textual descriptions. Unfortunately, this method is not feasible. On the one hand annotating images has to be done manually and is a very time-consuming task and on the other hand images may have contents that words cannot convey. Due to the rich content of images and the subjectivity of human perception no textual description can be absolutely complete and 100% correct [Siggelkow 02]. For example it is difficult to cover the complete spirit of Da Vinci’s Mona Lisa in words or the mood of a sunset at the seaside. Similarity of images depends on the user and the context of the query. In a general image database a radiograph might only be labeled as “radiograph” whereas this obviously is not sufficient within a medical database consisting of different types of radiographs only.

Apparently, other methods to search and index images are needed. A broad variety of applications requires searching for images: in medical applications many images are produced and a physician might search for similar images to learn about treatments of former patients and their outcomes [Petrakis & Faloutsos$^+$ 02]. A journalist might look for an image to illustrate an article [Markkula & Sormunen 98]. All these examples confront us with the same problem: The search is not an exact search like in traditional databases (e.g. the physician asks the database for the treatment of a patient with a certain name) but instead, the search is imprecise. That is, we are looking for similar entries from the database. Similar (this imprecise criterion) has to be precisely defined to enable an automatic search process. In fact it is possible to reason precisely about imprecise data [Subrahmanian & Tripathi 98].

Some of the areas mentioned allow for or even require the inclusion of special domain knowledge to obtain suitable results. Other tasks are too general to find special domain knowledge [Smeulders & Worring$^+$ 00]. To retrieve general image data from an unrestricted image database no special knowledge can be applied, but to retrieve similar faces it is possible and recommendable to include domain-knowledge.

In this work a focus is set on features for image retrieval. That is, we investigate the development of representations that allow searching for images similar to a given image. These features also open up new perspectives in other fields. The features investigated here are used for the tasks of classification and for clustering images into visually similar groups. Many features that were used in image retrieval before are exemplified and compared.

The main contributions of this work are quantifieable examinations of a wide variety of different features and different distance measures and a method to compare different content-based image retrieval systems. The method is an extension of the method proposed in [Müller & Müller$^+$ 01]. We implemented a system using the features and distance measures examined for classification and a system to cluster images according to their visual appearance. To achieve these goals, a flexible and easily extensible content-based image retrieval system was developed as well as a program to cluster images.

The remainder of this document is organized as follows: Chapter 2 presents the tasks investigated and introduces the underlying principles. Chapter 3 gives a survey of image retrieval techniques and systems available to give an idea of what has already been achieved in this field and to present the methods used. In Chapter 4 we introduce different features tested in the context of this work. The features are the first step towards a definition of similarity. We present the preprocessing that is applied to the images to make it possible to compare them. Chapter 5 introduces a variety of distance measures to compare features. Together, the two Chapters define different ways to measure similarity. In Chapter 6 we present the applications built on top of the features and the distance measures. Chapter 7 presents the databases used to test the applications. These databases are mainly used for image retrieval, but also some new and interesting results for classification and clustering have been achieved. Chapter 8 presents an approach to evaluate the performance of image retrieval systems and clustering algorithms. Results for the three applications are given. For the image retrieval and clustering task the results are based on the proposed measures and for the classification task error rates are given. Finally, we conclude this work in Chapter 9 and propose further research.

Chapter 2

Basic principles

This chapter introduces the fundamentals for the tasks investigated in this work, gives a broad overview on the methods used, and presents some notations used in the remainder of this document.

2.1 Basic Principles for Image Retrieval

Image retrieval is the task of searching for images from an image database. The query to the database can be of various types as depicted in Figure 2.1.

Query-by-text: The user gives a textual description of the image he is looking for.

Query-by-sketch: The user provides a sketch of the image she is looking for.

Query-by-example: The user gives an example image similar to the one he is looking for.

In this work we focus on the query-by-example approach. Formally, query-by-example can be defined as follows:

Let B be the image database with $B := \{X_n \mid n = 1 \dots N\}$ where X_n is an image represented by a set of features: $X_n := \{X_{nm} \mid m = 1 \dots M\}$. Because a query Q is also an image, we have $Q := \{Q_m \mid m = 1 \dots M\}$. To query the database, a dissimilarity measure $D(Q, X_n)$ is calculated for each n as

$$D(Q, X_n) := \sum_{m=1}^{M} w_m \cdot d_m(Q_m, X_{nm}) \text{ for } n = 1 \dots N.$$

From these $D(Q, X_n)$, scores $S(Q, X_n) = e^{-D(Q,X_n)}$ are calculated and the image X_n with $n = \operatorname{argmax}_{n'}\{S(Q, X_{n'})\}$ is returned. Here, d_m is a distance function or dissimilarity measure and $w_m \in \mathbb{R}$ is a weight. For each d_m, $\sum_{n=1}^{N} d_m(Q_m, X_{nm}) = 1$ holds.

By adjusting the weights w_m it is possible to emphasize properties of different features. For example to search for an image of a sunset the colors in the image might be very important, but to search for images of zebras texture is a very important feature.

In image retrieval an interaction with the user is given in most cases and this interaction allows the user to control the search and maybe help to come to good results faster. One widely used approach in information retrieval is relevance feedback. That is, after an initial search the user is presented with the set of results. Naturally, some of the results match

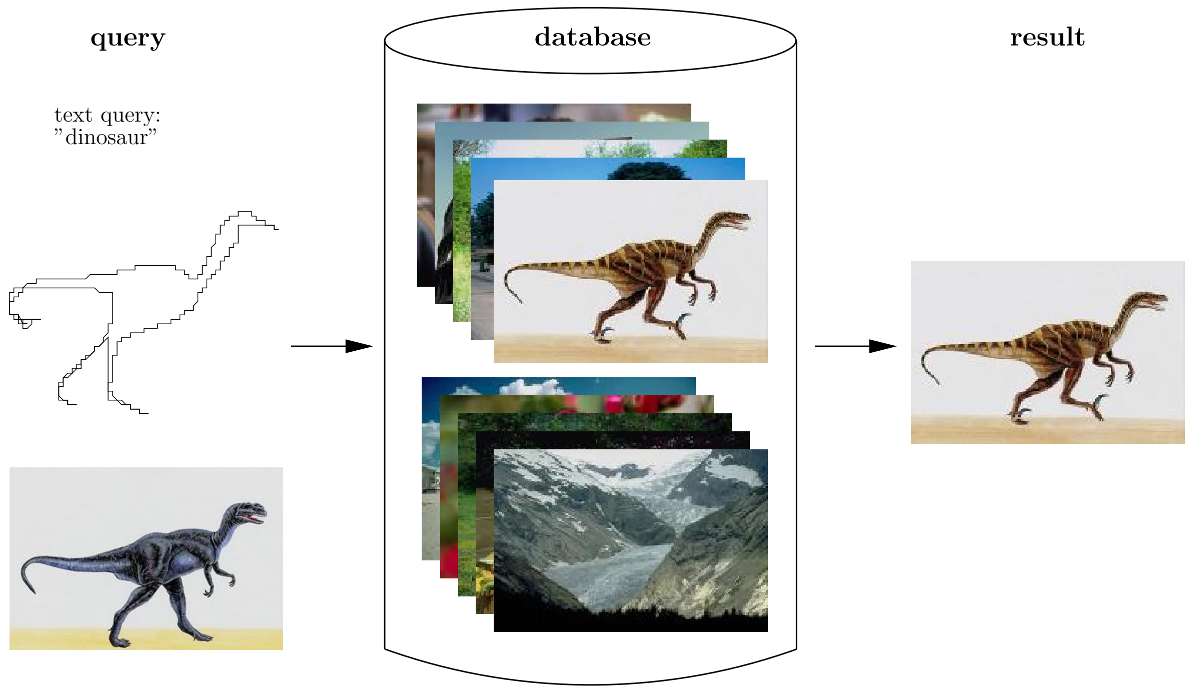


Figure 2.1: Content-based image retrieval.

the query and some do not. Now the user marks good results as relevant and bad results as irrelevant.

It has been shown that relevance feedback is extremely useful in information retrieval tasks and some good results have been achieved in image retrieval as well [Müller & Müller+]. We have to distinguish between two different types of relevance feedback: just positive relevance feedback or positive and negative relevance feedback. Negative relevance feedback can improve the results strongly, but may lead to bad results if too much negative feedback is given.

Formally, feedback can be seen as follows: Let $Q^+ = \{Q_1^+, \dots, Q_{N^+}^+\}$ be the set of positive query images and $Q^- = \{Q_1^-, \dots, Q_{N^-}^-\}$ be the set of negative query images. Then, for each of the positive and for each of the negative query images we calculate the scores for each of the database images and calculate

$$S(Q^+, Q^-, X_n) = \sum_{n'=1}^{N^+} S(Q_{n'}^+, X_n) + \sum_{n'=1}^{N^-} \big(1 - S(Q_{n'}^-, X_n)\big)$$

for each of the database images. Finally, the database images with highest score are returned.

When using relevance feedback it is important not to use too many negative examples since this might cause problems [Müller & Müller+]: Selecting too many negative examples can inhibit important features from positive examples to be taken into account.

2.2 Basic Principles for Image Clustering

Another task in the context of searching pictures arises from the following situation: Many image databases index the images by text which can be found in the context of the image or has been created manually to annotate the image. In this case image retrieval is reduced to text retrieval. Probably the best established example for this method is the Google image search. This image search engine crawls the web, saves thumbnails of the images found, and indexes them using the text from their context, e.g. captions or surrounding texts. Searching for an image by text confronts the user with a new problem: many words are ambiguous and searching for them results in different types of images. Searching with Google image search for the word "cookie" results in at least three different types of images: images of edible cookies, screenshots of programs dealing with cookies in the context of the Internet, and images not concerned with cookies at all. Even when searching for words with less ambiguity nearly always two groups of images are returned: one meeting the requirements and an unsuitable one. To solve this problem we propose to take the results obtained by the textual retrieval method and regroup them using methods from computer vision to present a more conveniently ordered set of results to the user. That is, the images returned are reordered to form groups of visually similar images. An application reordering its input images in such a manner is described in Chapter 6.2.

The idea of grouping data into clusters is not a new one, but has been a research topic for a long time in different contexts. A detailed overview can be found in [Jain & Dubes 88].

To describe this situation more precisely, we give the following formal definition: Let A be a set of observations to be clustered: $A = \{X_n \mid n = 1 \dots N\}$. This data is to be partitioned into a set of clusters $U = \{u_c \mid c = 1 \dots C\}$ such that $u_c = \{X_{n'} \mid n' = 1 \dots N'_c\}$ with $u_c \cap u_{c'} = \emptyset$ for all $c \neq c'$ and $\bigcup_{c=1}^{C} u_c = A$. This partitioning should have the property that only similar observations are in each cluster u_c. Here, similar means that a given distance measure $d(X_n, X'_n)$ is small for observations X_n and X'_n from the same cluster but large for observations from different clusters.

To meet these requirements, many algorithms have been proposed. In this work two different algorithms are used. The k-means clustering is probably the simplest cluster algorithm based on the squared error criterion. The LBG-Clustering-Algorithm can be seen as an extension of the k-means algorithm.

2.2.1 k-Means Clustering

The k-means algorithm is a very popular method to partition data. Its objective is to find C mean vectors $\mu_1, \dots, \mu_C$, one for each cluster for a given C. The basic idea of this iterative clustering algorithm is to start with an initial randomly chosen partition and assign patterns to the clusters such that the squared error

$$e^2 = \sum_{c=1}^{C} \sum_{n=1}^{N'_c} d(X_n^{(c)}, \mu_c)^2$$

is reduced, where $X_n^{(c)}$ is the n-th pattern belonging to the c-th cluster, μ_c is the mean vector for the c-th cluster, and d is a distance function comparing images.The k-means clustering algorithm works as follows:

1. Select an initial partition with C clusters.

2. Generate a new partitioning by assigning each pattern X_n to its closest cluster center μ_c. That is, the cluster center μ_c with $c = \operatorname{argmin}_{c'=1}^{C} d(X_n, \mu'_c)$ where d is a dissimilarity measure.

3. Compute new cluster centers $\mu_1, \dots, \mu_C$ as the centroids of the clusters.

4. Repeat step 2 and 3 until a stopping criterion is reached.

In step 1 the initial partition can be formed by first specifying a set of C seed points. This initial partition could be the first C points $X_1, \dots, X_C \in A$ or uniformly distributed points in the pattern space. In step 2 the distances between each of the observations and each of the cluster centers is calculated and then for each point the nearest cluster center is chosen. In step 3 the center μ_c for each cluster is reestimated by determining the mean of the cluster members. The second and the third step are repeated until a convergence criterion is met.

The k-means algorithm is easily implemented and has a time complexity of $\mathcal{O}(nC)$ for each iteration. A major drawback with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function.

2.2.2 LBG-Clustering

The LBG-clustering algorithm has been proposed by [Linde & Buzo$^+$ 80] and can be seen as an extension of the fuzzy k-means algorithm. Initially, the data is described by one Gaussian. Then this density is iteratively split until a criterion is reached. This criterion might be e.g. "the variance is below a certain threshold" or "there are less than a certain number of observations in a cluster". After each split, the densities are reestimated for several iterations similar to the k-means algorithm. Instead of the fuzzy memberships we use maximum approximation for the cluster memberships. This clustering algorithm works as follows:

1. Estimate the initial Gaussian density.

2. Split each density into two parts if it fulfills some split criterion.

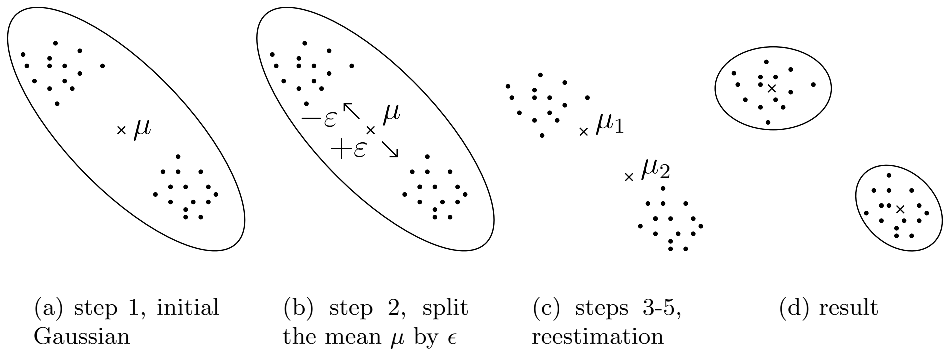
3. Generate a new partitioning by assigning each pattern X_n to its closest cluster center μ_c. That is, the cluster center μ_c with $c = \operatorname{argmin}_{c'} d(X_n, \mu'_c)$ where d is a dissimilarity measure.

4. Compute new cluster centers $\mu_1, \dots, \mu_C$ as the centroids of the clusters.

5. Repeat step 3 and 4 until an optimum value for the error criterion is reached.

6. Repeat from step 2 until some criterion or the maximum number of splits is reached.

In step 1 the mean μ and the variance σ^2 of the data is estimated (cp. Figure 2.2(a)). In step 2 each density meeting a criterion is split into two densities (cp. Figure 2.2 (b)). Several split criterions are possible: split all clusters, split only the cluster with the highest number of members, or split only the cluster with the highest variance. In practice splitting all clusters has proven to work best for most of the cases. Splitting can be done in various manners: Disturbing the mean by a constant value $\mu \mapsto \{\mu' = \mu - \epsilon, \mu'' = \mu + \epsilon\}$, disturbing the mean by multiples of the mean itself $\mu \mapsto \{\mu' = \mu - \epsilon\mu, \mu'' = \mu + \epsilon\mu\}$, disturbing the mean by the variance $\mu \mapsto \left\{\mu' = \mu - \epsilon\sigma^2, \mu'' = \mu + \epsilon\sigma^2\right\}$. These methods have shown not to work for histograms since histograms are normalized and these methods break the normalization. Thus, for histograms we propose to split the means by addition and subtraction: $\mu \mapsto \left\{\mu' = \mu \binom{+}{-} \epsilon\mu, \mu'' = \mu \binom{-}{+} \epsilon\mu\right\}$. The sign of the disturbance factor is changed after half of the entries of μ. That is, μ has D entries, and μ_d is the d-th entry: $\mu'_d = \mu_d + c\mu$ for



$\times\,\mu$
$-\varepsilon\nwarrow\ \times\ \mu$
$+\varepsilon\ \searrow$
$\times\,\mu_1$
$\times\,\mu_2$
(a) step 1, initial Gaussian
(b) step 2, split the mean μ by ϵ
(c) steps 3-5, reestimation
(d) result

Figure 2.2: Linde Buzo Gray clustering.

$d = 1, \ldots \lfloor \frac{D}{2} \rfloor$ and $\mu'_d = \mu_d - c\mu$ for $\lfloor \frac{D}{2} \rfloor + 1, \ldots, D$. The variance σ is just copied in the split $\sigma \mapsto \{\sigma', \sigma''\}$. In step 3 the distances between each of the observations and each of the cluster centers is calculated and then for each point the nearest cluster center is chosen (cp. Figure 2.2 (c)). For the calculation of the distances the variances are taken into account. After this step clusters with less members than a certain threshold are removed. In step 4 the center μ_κ for each cluster is reestimated by determination of the mean of the cluster members (cp. Figure 2.2 (d)). Steps 3 and 4 are repeated until a convergence criterion is met. These two steps are exactly the same as in the k-means algorithm. When the criterion is met it is decided whether the clusters are split again. If not, the algorithm terminates.

A main benefit here is that the number of clusters is found automatically and is not required to be given by the user as in the k-means algorithm. Experience shows that this algorithm often yields better partitions than the simple k-means algorithm.

2.3 Basic Principles for Classification

The task of image classification is to assign class labels to images. Classification is needed for optical character recognition and for face recognition for example. Usually a large amount of training data is available and the classification process is based on this training data. A classification system assigns a class label $k \in \{1, \ldots, K\}$ to an observation. To classify an image X a decision rule r is applied:

$$r : \mathbb{R}^D \mapsto \{1, \ldots, K\}, \quad r(X) = k$$

For this decision rule, the image X is represented by features. One possibility is to use the pixel values of the image directly as features. Other possibilities are explained in Chapter 4.

One possibility to define the decision rule r is to determine the class for every possible observation and store it, thus converting the classification problem into a search problem. This approach is usually infeasible since the amount of possible observations is simply too large. Consider 16×16 images where each pixel can have a gray value from the range 0 to 255. Then, $256^{16\times 16}$ different observations are possible.

A more feasible possibility is to use a set of labeled training observations $\{X_1, \ldots, X_N\}$ with labels $\{k_1, \ldots, k_N\}$ to define the decision rule by a discriminant function $g(X, k)$:

$$r(X) = \operatorname*{argmax}_{k'} \{g(X, k')\}.$$

In this work the classification is always performed using the nearest neighbor (NN) rule. That is, an image is classified to be from the same class as the closest image from the training corpus according to a given distance measure. To realize this classification rule the discriminant function g_{NN} is chosen as:

$$g_{\text{NN}}(X, k) = \begin{cases} 1, & \text{if } k = \operatorname*{argmin}_{k'} \min_{n=1,\ldots,N_{k'}} \{D(X, X_{k'n})\} \\ 0, & \text{else} \end{cases}$$

For many applications it is recommendable to use other discriminant functions as several works show [Schölkopf 97, Keysers & Och$^+$ 02, Dahmen & Hektor$^+$ 00, Ney 99] but in this work we only consider the nearest neighbor rule as there is a very close connection between classification using this rule and image retrieval as explained in Chapter 8.1.

2.4 Notation

This section is a reference for the symbols used in the context of this work.

Table 2.1: Symbols used in this work.

Symbol	Description
B	database of N images B=$\{X_n \mid n = 1 \ldots N\}$
X	image represented by M features $X = \{X_m \mid m = 1 \ldots M\}$
	$N_0 \times N_1$ image with gray value $X(n_0, n_1)$ at position $X(n_0, n_1)$
Q	query image represented by M features $Q = \{Q_m \mid m = 1 \ldots M\}$
$d_{\_}(\cdot, \cdot)$	dissimilarity function comparing two features
A	set of N images to be clustered $A = \{X_n \mid n = 1 \ldots N\}$
U	partitioning of A into C clusters $U = \{u_c \mid c = 1 \ldots C\}$
u_c	cluster with N'_c members $u_c = \{X_n^{(c)} \mid n = 1 \ldots N'_c\}$
ε	disturbance factor for density splits
K	number of classes
k_n	class of image X_n
r	decision rule
g	discriminant function
H	histogram with M bins
$\mathcal{S}$	feature space of the histogram, partitioned into M regions $\mathcal{S}^m$
$\mathcal{S}^m$	region belonging to the m-th bin
H_m	empirical probability for any point falling into $\mathcal{S}^m$
$\mathcal{V}^m$	value belonging to the m-th bin, e.g. center of $\mathcal{S}^m$
q_m	membership function of the m-th bin
$F(X)$	feature calculated from the image X invariant with respect to transformation g from a group of transformations G
$f(X) : \mathbb{X} \mapsto \mathbb{R}$	function mapping images to single values
$\mathcal{X}(u_0, u_1)$	2-D Fourier transform of the image $X(n_0, n_1)$

Chapter 3

State of the Art in Content-Based Image Retrieval

In content-based image retrieval many disciplines are combined. On the one hand there are image recognition and pattern analysis which try to describe the images in a way that makes it possible to distinguish between similar and dissimilar images. On the other hand there is the database part which tries to store the images and features to allow for efficient access to the data. This work is only concerned with the first part. The database and indexing part, which is a research area of its own, is not considered here. The remainder of this chapter presents an overview on the systems and methods available for content-based image retrieval to give an idea of what has already been achieved in this field. Finally, we give references to classification systems and works dealing with the clustering of visually similar images. Since only some of the works presented here give quantitative, comparable results, results are not presented here. Instead, results from these works are given in Chapter 8 in comparison to the results obtained in the context of this work.

3.1 Related Work in Content-Based Image Retrieval

[Smeulders & Worring^{+} 00, Rui & Huang^{+} 99, Fend & Siu^{+} 03] give overviews on the technical achievements in the field of content-based image retrieval. They review the processing applied to images for retrieval and discuss features extracted from images for searching. They also review different distance and similarity measures for different types of features. Finally, they give summaries about possible system architectures and the database techniques used. Also these surveys list image retrieval systems available and introduce the methods used.

One of the first content based image retrieval systems available was the QBIC system (Query By Image Content) from IBM [Faloutsos & Barber^{+} 94]. The QBIC system uses three types of features: color histograms to describe the color distribution, a moment based shape feature to describe shapes, and a texture descriptor based on contrast, coarseness, and directionality to account for textures. This system also uses database technology to handle the high dimensionality of the data. The system is available online[1].

A very popular content-based image retrieval system is the BlobWorld system, which has been developed at the University of California, Berkeley. This system is presented in

[1]http://wwwqbic.almaden.ibm.com

[Carson & Thomas+ 99, Carson & Belongie+ 02]. The system uses image features which are extracted using segmentation of images. This segmentation is done using an EM-style algorithm clustering the image pixels using color, texture, and position information. To query the database the user selects a region from an image and the system returns images containing similar regions. BlobWorld is available online[2].

A system which has strongly influenced this work is the SIMBA(Search IMages By Appearance) system [Siggelkow 02, Siggelkow & Schael+ 01, Siggelkow & Burkhardt 97]. This system uses features invariant against rotation and translation (cp. Chapter 4.3) accounting mainly for color and texture. By a weighted combination the user can adapt the similarity measures according to his needs. SIMBA is available online[3].

[Squire & Müller+ 99] present an approach for content-based image retrieval which is oriented towards the methods used in textual information retrieval. They propose a very high dimensional space (dimensionality: 80 000) of binary features and an inverted file to allow for efficient access. The features they use are a color histogram in HSV space and a set of Gabor coefficients. A weighting of different features is done depending on the number of occurrences in images. This approach is realized in the image retrieval system VIPER (Visual Information Processing for Enhanced Retrieval)[4] and it is freely available[5] under GNU Public License (GPL) as GIFT (GNU Image Finding Tool). Since it is freely available this system is also used in other institutions and is currently extended to a medical image retrieval system.

The CIRES system [Iqbal & Aggarwal 02] uses a color histogram with 15 bins as color features, Gabor features as texture representation. Additionally image structures like line crossings and line junctions are extracted from the images. These structures enhances the retrieval performance because structures of this type usually are found in man-made objects. They also show that this approach leads to a performance increase for images containing man-made objects. CIRES is available online[6].

[Wang & Li+ 01] propose another approach to increase image retrieval performance. The authors claim that preclassification improves retrieval results. They propose to preclassify images into semantic categories like graph/photograph, textured/non-textured which are relatively simple to classify. After this classification they return images belonging to the same semantic categories only. Apart from this, region-based features similar to the approach in BlobWorld are used, but the region descriptions of the images are matched automatically. This system is available online[7].

Efforts in the area of image retrieval are also made for medical applications. The IRMA (Image Retrieval in Medical Applications) project [Lehmann & Güld+ 03] is a cooperation of the Department of Diagnostic Radiology, the Department of Medical Informatics, and the Lehrstuhl für Informatik VI of the Aachen University of Technology (RWTH Aachen). Aim of the project is the development and implementation of high-level methods for content based image retrieval with prototypical application to medico-diagnostic tasks on a radiologic image archive[8].

Other image retrieval systems are available, but it is beyond the scope of this work to

[2]http://elib.cs.berkeley.edu/photos/blobworld/
[3]http://simba.informatik.uni-freiburg.de
[4]http://viper.unige.ch
[5]http://www.gnu.org/software/gift/gift.html
[6]http://amazon.ece.utexas.edu/ qasim/research.htm
[7]http://wang.ist.psu.edu/
[8]http://www.irma-project.org

present them all. They mainly differ in the features and the search structures they use. In Chapter 4 we present a selection of features and many of them have been used for image retrieval before.

A common problem in image retrieval is the performance evaluation. It is difficult to compare the available systems, because no common performance measure for image retrieval is established and even constructing a performance measure is difficult since the success or failure of an image query strongly depends on the requirements of the user. In Chapter 8 we focus on this problem and introduce and extend an approach proposed by [Müller & Müller$^+$ 01]. In this context a serious problem is the nonexistence of a standard database for comparison of different image retrieval systems. This problem is not directly addressed here but results are presented for a wide variety of different databases.

3.2 Related Work in Clustering of Images

In [Iyengar & Lippman 98] the authors propose to use clustering techniques to allow for efficient access to large image databases. More efficient access is important, since due to the size of large image databases, querying becomes expensive even if the images are represented in a compact manner. With clustering, the task of retrieval is decomposed into a two stage process. In the first step an appropriate cluster is selected and in the second step the best matches from this cluster are returned. They compare a clustering technique which uses relative entropy to techniques using the Euclidean norm.

[Käster & Wendt$^+$ 03] propose to use image clustering techniques to allow for faster searching in image databases. They compare different clustering techniques to find out which suits the task of clustering images best.

In [Saux & Boujemaa 02a, Saux & Boujemaa 02b] the authors propose to use image clustering to give a good overview of an image database to help a user find a sought image faster. To cluster this images, they estimate the distribution of image categories and search the best representative for each cluster. They represent images by a high-dimensional feature vector and propose a new clustering algorithm which they compare to other clustering techniques.

[Berkhin 02, Jain & Murty$^+$ 99, Jain & Dubes 88] give general information about clustering of data and the evaluation of results. In [Linde & Buzo$^+$ 80] a new clustering algorithm based on the EM algorithm is proposed and a method to avoid the problem of finding an initial partition by iterative splitting of an initial Gaussian describing all data points is introduced.

[Starik & Selding$^+$ 03] describe a method to cluster images into meaningful classes using a segmentation technique to compare the images. The segmentation is done using centroid models common to all images in the set and for clustering the information bottleneck algorithm is used to cluster the images based on the result of the segmentation.

[Barnard & Forsyth 01, Barnard & Duygulu$^+$ 01] present a method to combine the advantages of clustering images based on image features with the advantages of clustering images based on textual description. Using this combination improves the results strongly as was empirically tested in this works.

In [Deselaers & Keysers$^+$ 03a] preliminary results of this work are presented. A combination of methods from computer vision and data mining is proposed to improve the user-friendliness of text-based image search engines.

3.3 Related Work in Object-Recognition and Classification of Images

In this section some approaches to object recognition in images are presented. At the moment it is not yet possible to retrieve images from a database with arbitrary scenes based on the objects contained in the images, but this is an objective for future research [Smeulders & Worring$^+$ 00]. Here an overview of approaches for object recognition with a focus on object recognition in complex scenes like they usually can be found in general images is given.

A broad overview about the methods used in pattern analysis and classification is given in [Duda & Hart$^+$ 01]. Since object recognition is a special case of pattern recognition this book gives a good introduction into this topic.

In [Fergus & Perona$^+$ 03] the authors present a method to learn and recognize object class models from unlabeled and unsegmented scenes. Objects are modeled as flexible constellations of parts. For all aspects of the objects a probabilistic representation is used.

[Deselaers & Keysers$^+$ 03b] propose to use local representations to recognize multiple objects in one scene. Local representations, as proposed, convey some properties that are highly important for the task of multi object recognition: They are inherently invariant with respect to translations and they can cope well with partial occlusions. [Kölsch 03] describes and investigates variations to the methods of local representations which are of interest for these tasks.

In [Keysers & Motter$^+$ 03] the authors propose a holistic statistical model for automatic object training and recognition in complex scenes. That is, no local decisions about object boundaries, segmentation, or object transformations are taken. Instead, all pixels in the given scene are explained using an appearance based approach. Details about the statistical model as applied to medical images can be found in [Keysers & Dahmen$^+$ 03].

In [Frey & Jojic 03] a statistical model for automatic training of invariant object models is introduced. The authors propose the use of transformed mixtures of Gaussians to learn representatives of transformed objects from unsegmented data.

[Keysers 00] describes approaches to invariant object recognition. Translations considered are mainly affine transformations. [Gollan 03] considers a broader class of, especially non-linear, transformations for object recognition. Both approaches obtain state of the art results in optical character recognition and radiograph classification.

In [Reinhold & Paulus$^+$ 01] the authors present an appearance-based approach for the localization and classification of 2-D objects situated in heterogeneous background. Local features are derived from wavelet multiresolution analysis by statistical density functions. In addition to the object model, the background is modelled by a uniform distribution. That is, one density function is given to describe all possible backgrounds. It is experimentally shown that this model is well suited for the addressed recognition task.

[Obdrzalek & Matas 03] describe an approach to invariant object recognition using local representations normalized with respect to affine transformations. The local representations are taken from regions where certain shapes are detected and the subimages are normalized with respect to certain transformations. Discrete cosine transform is applied to the local representations to reduce the memory usage.

[Duygulu & Barnard$^+$ 02] use the same features as the image retrieval system BlobWorld and proposes to use methods from machine translation for object recognition. For the translation, the image features are considered as source language and the image description as target language.

[Barnard & Forsyth 01, Barnard & Duygulu$^+$ 01, Barnard & Duygulu$^+$ 02] present a statistical method to learn the relationship between image features and textual annotations. The annotations are used for minimally supervised object training from an annotated image database and full automatic object recognition.

Chapter 4

Features for Content-Based Image Retrieval

The basic idea of content-based image retrieval is not to rely on textual descriptions of image content. Instead, a set of features is used that allows the user to find images that are visually similar to a presented query image. Here "similar" may mean different things. A radiologist, for example, may have different criteria of similarity than a journalist. Obviously, these different needs cannot be fulfilled by exactly the same method. To allow for these different demands different descriptions of the images are needed. Different features may account for different properties of images. Some works divide the used features into different groups, e.g. color features, texture features, and shape features. Most of the features are indeed members of two or more of these groups. Therefore, we do not distinguish between these groups of features a priori but instead try to show which features have similar properties using an empirical correlation analysis.

In the remainder of this chapter a variety of features describing different properties of images is described which allow the user to search for images taking into account these different properties is given.

4.1 Image Features

The most direct approach to query for images is to compare the images directly. That is, the pixel values of the image itself or the pixel values of a scaled version are compared directly to the corresponding values of other images. For many applications this approach is not feasible as it is not clear which pixels from the one image correspond to which pixels in the other image. In optical character recognition this method is suitable when the symbols are already segmented since a letter to be recognized will probably be similar to another observation of the same letter when the letters are of equal size and contained at the same position in the image. Research about this and improved methods for finding possible pixel alignments are presented in [Gollan 03].

In addition to taking the pixel values themselves several extensions are possible. Filters and transformations can be applied to the image to give a more compact representation or to account for certain properties of the image, e.g. Sobel filters are applied to emphasize edges and discrete cosine transformation or PCA transformation are applied to give a more compact representation.

4.2 Color Histograms

A histogram is a way to approximate the distribution of a random variable. It is also a simple approach to give a description of an estimated density.

The feature space $\mathcal{S}$ is partitioned into M regions $\mathcal{S}^m, m = 1, \ldots, M$. Usually these regions form a regularly spaced grid, i.e. the regions $\mathcal{S}^m$ are hypercubes of the same size, but this is not a requirement. Formally:

$$\mathcal{S}^m \subset \mathcal{S} \quad \text{with} \quad \bigcup_{m=1}^{M} \mathcal{S}^m = \mathcal{S} \tag{4.1}$$

$$\text{and} \quad \mathcal{S}^m \cap \mathcal{S}^{m'} = \emptyset \quad \forall m \neq m' \tag{4.2}$$

The empirical probability for data points falling into one of these regions is determined by counting. Let K^m be the number of data points in region $\mathcal{S}^m$ from a total of N points, then the empirical probability H_m for any data point x to be from this region is given by $H_m = P\left(x \in \mathcal{S}^m\right) = \frac{K^m}{N}$.

To create a color histogram the color space has to be divided into regions. For example, the widely used 24 bit RGB color space contains 2^{24} regions. A histogram containing as many histogram bins would be too large to be dealt with efficiently. To reduce the amount of memory needed the feature space is quantized. Here, it is required to find a good trade-off between loss of precision and memory requirement. For gray images, the situation is somewhat better because gray images usually contain 256 different gray levels only. 256 bins are still a manageable amount of data. After partitioning the feature space, for each region the number of pixels from this region is counted to calculate the empirical probabilities.

A problem with histograms is the discontinuity [Siggelkow 02]. That is, slightly changing the image might change the bin assignments and thus the resulting histogram completely. To overcome this problem fuzzy histograms can be used. The goal of fuzzy histograms is to remove the discontinuous bin assignment of the traditional histogram. The membership function $q_m(x)$ for bin m of a discontinuous histogram is defined as

$$q_m(x) = \int_{-\Delta}^{+\Delta} g(x - \mathcal{V}^m + z)dz$$

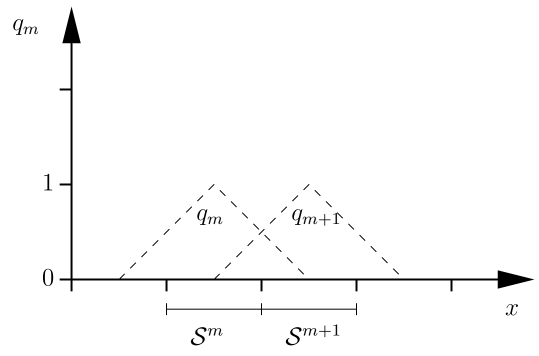
where each bin starts at position $\mathcal{V}^m - \Delta$ and ends at position $\mathcal{V}^m + \Delta$ with $g(x) = \delta(x)$ where $\delta(x)$ is the Kronecker Delta.

That is, only one of the M membership functions is not zero for each point x. The assignment is discontinuous at the boundaries of the membership functions and due to this discontinuouity very small variations can cause jumps in the assignments.

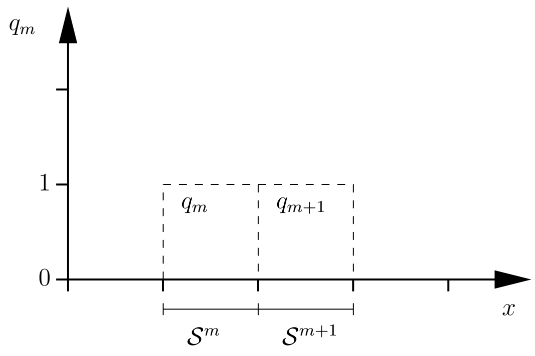
[Siggelkow 02] proposes a modified histogram with a continuous bin assignment function. In Figure 4.1(a) a continuous bin-assignment function with

$$g(x) = \begin{cases} \frac{1}{\Delta^2}(x + \Delta) & \text{for } x < 0 \\ -\frac{1}{\Delta^2}(x - \Delta) & \text{for } x >= 0 \end{cases}$$

is shown. In Figure 4.1(b) the standard discontinuous bin assignment function is shown Nearly the same effect can be achieved by creating the normal discontinuous histogram first and then smoothing it, for example by convolution with a Gauss filter.



(a) example membership function for fuzzy histogram



(b) membership function for normal histogram

Figure 4.1: Examples of bin-assignment functions.

4.3 Invariant Features

An invariant feature is a feature calculated from an image that is invariant with respect to certain transformations, i.e. it does not change when these transformations are applied to the image. The transformations considered here are mainly translation, rotation, and scaling. In the remainder of this section we consider gray images only, but the methods are easily extensible to color images.

Formally, an image X is an $N_0 \times N_1$ matrix of values $X(n_0, n_1)$, that can e.g. be interpreted as gray values. Considering affine transformations that transform a coordinate system into another coordinate system such that the point (n_0, n_1) in the transformed coordinate system is (n_0', n_1') in the original coordinate system as

$$\begin{pmatrix} n_0' \\ n_1' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} n_0 \\ n_1 \end{pmatrix} + \begin{pmatrix} t_0 \\ t_1 \end{pmatrix},$$

an image X can be transformed into another image X' with $X'(n_0, n_1) = X(n_0', n_1')$. Let g be the transformation with $X' = gX$ then we are looking for a feature $F(X)$ invariant with respect to the transformation g, that is, $F(X) = F(gX)$.

The set of images which can be obtained from one image by any valid transformation g is called an equivalence class and all images from one equivalence class should yield the same invariant feature $F(X)$. Details about further requirements for invariant features can be found in [Siggelkow 02].

To construct invariant features, three approaches are available:

Normalization. The goal of image normalization is to find a distinctive representation for any equivalence class of images. That is, each image X from an equivalence class is mapped to the same representation. For example, normalization with respect to translation on gray value images can be achieved easily by calculating the center of gravity of the gray values of the image and translating this center of gravity to the image center.

Differential approach. Invariants may be obtained by solving partial differential equations. Various types of differential invariants have been used for shape-descriptors e.g. curvature and torsion [Brill & Barrett$^+$ 92]. A small set of differential invariants may contain all the essential information about the curve. This approach is theoretically interesting, but segmentation is needed since the contours of the segments are processed. Segmentation is usually not possible in common image retrieval as complete image understanding is needed for a perfect segmentation. Also, in practice the differential equations involved become complex and hard to solve, though using a priori knowledge can help to reduce the number of necessary partial derivatives [Squire & Caelli 00].

Integral approach. The construction of invariant features is possible by integration over the transformation group [Schulz-Mirbach 95]. That is, all possible transformed images are considered and an integral over all these images is carried out. This approach is described in more detail in the following sections.

4.4 Invariant Features by Integration

To create features invariant to certain transformations an integral over all transformed images is calculated [Siggelkow 02]. That is, all possible transformations of the image are considered, to each of these images a certain function is applied, and the integral over these functions is calculated. This integral is invariant with respect to the transformations considered.

Let X be a gray value image, $X\left(n_0, n_1\right)$ the gray value at position (n_0, n_1) and let $g \in G$ be a transformation from the group of transformations G. Then gX is the image X transformed by g such that $X(n_0', n_1') = gX(n_0, n_1)$. Let f be a function from an image to a real number: $f : \mathbb{X} \mapsto \mathbb{R}$, then $F(X)$ with

$$F\left(X\right) = \int\limits_{g \in G} f\left(gX\right) dg$$

is invariant with respect to any transformation from G.

For example to create a translation invariant feature for an image X of size $N_0 \times N_1$ the following can be calculated:

$$F\left(X\right) = \frac{1}{N_0 N_1} \int\limits_{t_0=1}^{N_0} \int\limits_{t_1=1}^{N_1} f\left(g_{n_0,n_1} X\right) dt_1 dt_0$$

This method is easily extensible to affine transformations. An affine transformation transforms the coordinate system of a point (n_0, n_1) by

$$\begin{pmatrix} n_0' \\ n_1' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} n_0 \\ n_1 \end{pmatrix} + \begin{pmatrix} t_0 \\ t_1 \end{pmatrix}$$

In the following consider the group of Euclidean motion is considered, that is the group of rotations and translations $G_{r,t}$ and the group of rotation, scaling, and translation $G_{r,s,t}$.

A transformation rotating the image by φ and translating it by (t_0, t_1) can be expressed as affine transformation as follows. The point (n_0, n_1) is transformed to the point (n_0', n_1') with

$$\begin{pmatrix} n_0' \\ n_1' \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} n_0 \\ n_1 \end{pmatrix} + \begin{pmatrix} t_0 \\ t_1 \end{pmatrix}.$$

To obtain a feature invariant against these translations integration over t_0, t_1 and φ is necessary. In practice, the parameters are discretized

$$F\left(X\right) = \frac{1}{N_0 N_1 R} \sum_{t_0=1}^{N_0} \sum_{t_1=1}^{N_1} \sum_{r=1}^{R} f\left(g_{t_0, t_1, \frac{2\pi r}{R}} X\right)$$

and a function f is selected. Choosing e.g. $f\left(X\right) = X\left(1, 0\right) \cdot X\left(0, 2\right)$ leads to the following expression which is evaluated:

$$\begin{aligned} F\left(X\right) = \frac{1}{N_0 N_1 R} \sum_{t_0=1}^{N_0} \sum_{t_1=1}^{N_1} \sum_{r=1}^{R} & X\left(\cos\frac{2\pi r}{R} + t_0, \sin\frac{2\pi r}{R} + t_1\right) \\ & \cdot X\left(-2\sin\frac{2\pi r}{R} + t_0, 2\cos\frac{2\pi r}{R} + t_1\right) \end{aligned}$$

The same applies for extension to scaling. Here, the transformation can be expressed as the coordinate transformation given by

$$\begin{pmatrix} n_0' \\ n_1' \end{pmatrix} = s \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} n_0 \\ n_1 \end{pmatrix} + \begin{pmatrix} t_0 \\ t_1 \end{pmatrix}$$

where s is the scaling factor. Choosing f as above $f\left(X\right) = X\left(1, 0\right) \cdot X\left(0, 2\right)$ and selecting a set $S = \{s_1, \ldots, s_n\}$ of scale factors this leads to

$$\begin{aligned} F\left(X\right) = \frac{1}{N_0 N_1 R |S|} \sum_{t_0=1}^{N_0} \sum_{t_1=1}^{N_1} \sum_{r=1}^{R} \sum_{s \in S} & X\left(s\cos\frac{2\pi r}{R} + t_0, s\sin\frac{2\pi r}{R} + t_1\right) \\ & \cdot X\left(-2s\sin\frac{2\pi r}{R} + t_0, 2s\cos\frac{2\pi r}{R} + t_1\right) \end{aligned}$$

For the calculation of perfectly scale invariant features it is necessary to consider all possible scalings of an image. That is, the image has to be scaled from the size where the function's support is larger than the image (here: image size 2×2 pixels) up to the size where the function $f(X)$ is completely contained in one pixel (here: pixel size 2×2 pixels). This requirement leads to a large amount of scalings to be considered.

In many applications it is desired to consider features not completely invariant to rotation but only invariant to rotation to a certain degree. In optical character recognition rotation invariance is a problem since using completely rotation invariant features makes e.g. “6” and “9” indistinguishable which is not wanted. Using the approach presented here, it is not possible to create features invariant to a certain degree of rotation because considering some rotations only does not lead to the same features for rotated images. Nevertheless we tried to obtain partially rotation invariant features by integrating over sectors of a circle instead of integrating over complete circles and present the results of these experiments in Chapter 8.

For the calculation of these features it is not necessary to really transform the image using all the transformations from the transformation groups, but it is possible to calculate the features more directly. This fact can be seen from the equations above and the calculation strategy as depicted in Figure 4.2. Instead of transforming the image using all transformations, the inverse transformations are applied to the function f and then this function is applied to

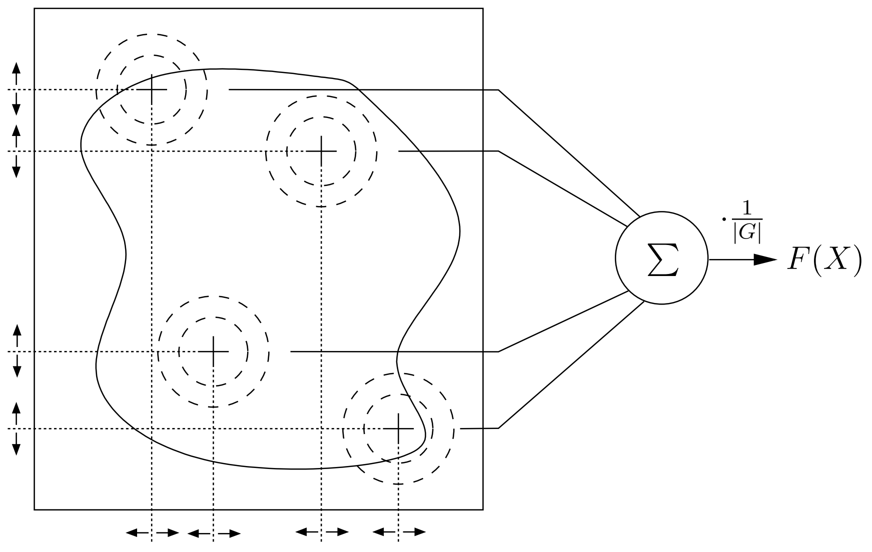


Figure 4.2: Calculation strategy for invariant integration.

the untransformed image. For the case of translation and rotation the sum over the rotation angle r is calculated for each point of the image and then these sums are summed up (cp. Figure 4.2).

To speed up this process further, it is possible to approximate the exact invariant feature using Monte Carlo integration. This method allows to get an invariant feature with a given precision at a very high probability. Instead of integrating over all positions of the image the integration is carried out over a fixed number of positions and thus it is possible to create an algorithm which can extract an invariant feature in a fixed time. Details about this approximation method can be found in [Siggelkow 02, Siggelkow & Schael 99].

Now we come to the issue of choosing an appropriate $f(X)$. $f(X)$ is a function $f : \mathbb{X} \mapsto \mathbb{R}$. This function can be of various types. Monomial and relational functions are considered. The first kind of functions to be considered are monomials:

$$f(X) = \prod_{j=1}^{J} X(n_0^{(j)}, n_1^{(j)})$$

Using this type of functions it is possible to construct complete feature sets for finite transformation groups [Siggelkow 02].

As a first example consider the trivial monomial $f(X) = X(0,0)$. Using this function to obtain the invariant feature $F(X)$ yields simply the mean gray value. For other monomials we obtain average results of products calculated on certain constellations of pixels under all possible rotations and translations. For monomials with small support the result is very similar to the mean gray value. Using monomials of different support size allows us to construct features capturing information of different resolutions from an image. In practice monomials of the type

$$f(X) = \sqrt[J]{\prod_{j=1}^{J} X(n_0^{(j)}, n_1^{(j)})}$$

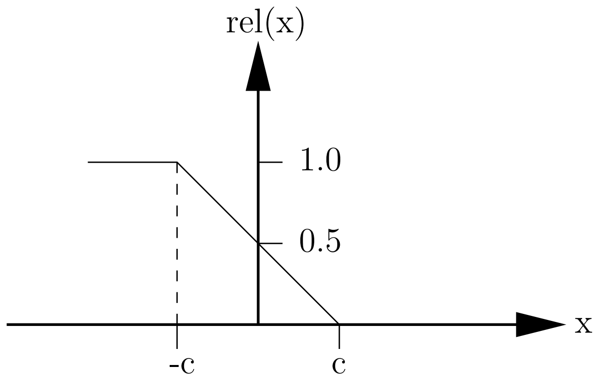


Figure 4.3: The rel-function.

are used to not distort the range of the results and to keep the result within the same range as the original image values. Since the monomial functions are very sensitive to illumination changes another type of functions for the invariant features was proposed: Relational functions [Schael 01] have been used for texture representation in the same framework of integrational invariants. Features obtained using relational functions are robust to illumination changes in contrast to features obtained using monomial kernels. The according function $f(X)$ is defined as

$$f(X) = \mathrm{rel}\left(X\left(n_0^{(1)}, n_1^{(1)}\right) - X\left(n_0^{(2)}, n_1^{(2)}\right)\right)$$

with

$$\mathrm{rel}(x) = \begin{cases} 1 & \text{if } x \leq -c \\ \frac{1}{2c}(c - x) & \text{if } -c < x \leq c \\ 0 & \text{if } c < x \end{cases} \qquad \text{(cp. Figure 4.3)}$$

The rel-operator maps the relations between the gray values of the pixel $X\left(n_0^{(1)}, n_1^{(1)}\right)$ and the pixel $X\left(n_0^{(2)}, n_1^{(2)}\right)$. In practice, we do not add up the resulting values for the different rotations but create a 3-bin "fuzzy" histogram of these values. This histogram counts the amount of pixels that are brighter, equal, and darker than the first pixel. The resulting invariant feature is not completely invariant against changes of brightness but is still robust to monotonic gray value transformations.

The approach of integration over the transformation group thus yields features invariant against a given group of transformations and is easily extensible to other transformations. A problem is that the invariant feature only consists of one value per gray value image respectively three for an RGB image, and this is not rich enough to discriminate between different images. To solve this problem, two approaches are available.

4.4.1 Invariant Feature Histograms

The first approach to solve the problem of insufficient data in the invariant feature is based on the idea of replacing one or more of the integrals by histogramization. In fact, any commutative operator preserves the invariance property and the sum and histogramization are only practical examples. Thus it is possible to predetermine the dimensionality of the information

Table 4.1: Monomials used for invariant feature vectors.

$X(0,0)$	$\sqrt{X(0,0)X(0,1)}$	$\sqrt[3]{X(0,0)X(0,1)X(0,1)}$
	$\sqrt{X(0,0)X(0,2)}$	$\sqrt[3]{X(0,0)X(0,1)X(0,2)}$
	$\sqrt{X(0,0)X(0,4)}$	$\sqrt[3]{X(0,0)X(0,1)X(0,4)}$
	$\vdots$	$\vdots$
	$\sqrt{X(0,0)X(0,32)}$	$\sqrt[3]{X(0,0)X(0,1)X(0,32)}$
		$\sqrt[3]{X(0,0)X(0,2)X(0,1)}$
		$\vdots$
		$\sqrt[3]{X(0,0)X(0,2)X(0,32)}$
		$\vdots$
		$\sqrt[3]{X(0,0)X(0,32)X(0,32)}$

gathered and the computational complexity remains unchanged. The histogram H_F is given by

$$H_F\left(X\right) = \operatorname*{hist}_{t_0=1,t_1=1}^{N_0,N_1} \frac{1}{R} \sum_{r=1}^{R} f\left(g_{t_0,t_1,\frac{2\pi r}{R}} X\right)$$

Note that using the function $f\left(X\right) = X\left(0,0\right)$ results in a simple color histogram as described in Section 4.2.

In practice, for RGB images we create 3-dimensional histograms with six, seven, or eight splits in each dimension. This results in histograms of 216, 343, or 512 bins respectively. For gray images histograms with 64, 128 or 256 bins are created.

4.4.2 Invariant Feature Vectors

Another approach to solve the problem of insufficient data in the invariant feature is to calculate various different invariant features using different functions f. This approach results in a vector $V = (v_1 \dots v_N)$ of invariant features with $v_n = F_n\left(X\right)$ and the f_n are functions from a set of functions $\{f \mid f : \mathbb{X} \mapsto \mathbb{R}\}$.

The calculation of invariant feature vectors is computationally expensive. The complexity is increased by factor N in comparison to the calculation of an invariant feature or invariant feature histogram.

In practice we use vectors with 44 different monomial functions. The functions used are listed in Table 4.1. Thus 44 dimensional vectors for gray images and 132 dimensional vectors for RGB images are obtained.

4.4.3 Invariant Fourier Mellin Features

It is well known that the amplitude spectrum of the Fourier transformation is invariant against translation. Using this knowledge and log-polar coordinates it is possible to create a feature invariant with respect to rotation, scaling, and translation [Reddy & Chatterji 96, Dahmen & Hektor$^+$ 00], of which a short overview is given here.

The discrete Fourier transformation $\mathcal{X}\left(u_0, u_1\right)$ of a 2D discrete image $X\left(n_0, n_1\right) \in R^{N \times N}$

is defined as

$$\mathcal{X}(u_0, u_1) = \frac{1}{N^2} \sum_{n_0=1}^{N} \sum_{n_1=1}^{N} X(n_0, n_1) e^{-\frac{2\pi i(u_0 n_0 + u_1 n_1)}{N}}$$

with $i = \sqrt{-1}$ and $u_0, u_1 = 0, 1, \ldots, N-1$.

Let $f(n_0, n_1)$ be a function with the Fourier transform $\mathcal{F}(u_0, u_1)$, then:

- $f'(n_0, n_1) = f\left(n_0 - n_0^{(0)}, n_1 - n_1^{(0)}\right) \Rightarrow \mathcal{F}'(u_0, u_1) = \mathcal{F}(u_0, u_1) \cdot e^{-2\pi i\left(n_0^{(0)} u_0 + n_1^{(0)} u_1\right)}$
- $f'(n_0, n_1) = f(\beta n_0, \beta n_1) \Rightarrow \mathcal{F}'(u_0, u_1) = \frac{1}{\beta^2}\mathcal{F}\left(\frac{u_0}{\beta}, \frac{u_1}{\beta}\right)$
- $f'(n_0, n_1) = f(n_0 \cos\alpha + n_1 \sin\alpha, -n_0 \sin\alpha + n_1 \cos\alpha)$
  $\Rightarrow \mathcal{F}'(u_0, u_1) = \mathcal{F}(u_0 \cos\alpha + u_1 \sin\alpha, -u_0 \sin\alpha + u_1 \cos\alpha)$.

Using these 3 properties it is possible to derive the following characteristics of the amplitude spectrum $A(u_0, u_1)$: It is invariant with respect to translation, inverse-variant with respect to scaling and variant with respect to rotation. Thus, features based on the amplitude spectrum of an image are translation invariant. By transforming the rectangular coordinates (u_0, u_1) of $A(u_0, u_1)$ to polar coordinates (r, θ) and using a logarithmic scale for the radial axis, image scales and rotations become shifts in the log-polar representation $\hat{A}$ of A. Thus the amplitude spectrum of $\hat{A}$ is invariant with respect to rotation, scaling and translation. From the amplitude spectrum of $\hat{A}$ we extract 15×25 features considering the symmetry which is in fact a low-pass filtering. That is, we take the values around the direct current such that no symmetrical values are taken. Thus, a 375-dimensional feature vector is taken. For the coordinate transformation from rectangular to log-polar coordinates a B-spline interpolation is used.

The whole process is shown in Figure 4.4. Two images are Fourier transformed, then converted to log-polar coordinates, and then again Fourier transformed. It can be seen that the resulting features are identical as the differnce is 0 in all positions.

4.5 Gabor Features

Gabor filters are a well known technique for texture analysis and have been used in several works [Park & Jin+ 02, Zhang & Wong+ 00, Iqbal & Aggarwal 99] before. In this work we use the approach presented in [Palm & Keysers+ 00, Keysers 99] where the HSV color space (hue, saturation, value) is used. Hue and saturation are represented as one complex value in the images. Here a short overview of the proposed method is given.

For texture analysis, frequency elements, as obtained by Fourier transformations, are an important part of an image but the simple Fourier transform discards all spatial information. To retain spatial information, the windowed Fourier transform (WFT) is used and results in a spatial/frequency representation of the image. For this purpose, the image $X(n_0, n_1)$ is multiplied by the window function $w(n_0, n_1)$ and then the Fourier transform is applied:

$$\begin{aligned}
\mathcal{X}\left(n_0^{(0)}, n_1^{(0)}, u_0, u_1\right) &= \int_{-\infty}^{\infty} X(n_0, n_1)\, w\left(n_0 - n_0^{(0)}, n_1 - n_1^{(0)}\right) e^{-2\pi i(u_0 n_0 + u_1 n_1)} dn_0 dn_1 \\
&= e^{-2\pi i\left(u_0 n_0^{(0)} + u_1 n_1^{(0)}\right)} \left[X\left(n_0^{(0)}, n_1^{(0)}\right) * \left(w\left(n_0^{(0)}, n_1^{(0)}\right) e^{2\pi i\left(u_0 n_0^{(0)} + u_1 n_1^{(0)}\right)}\right)\right] \\
&= e^{-2\pi i\left(u_0 n_0^{(0)} + u_1 n_1^{(0)}\right)} \left[X * m_{u_0, u_1}\right](n_0^{(0)}, n_1^{(0)}).
\end{aligned}$$

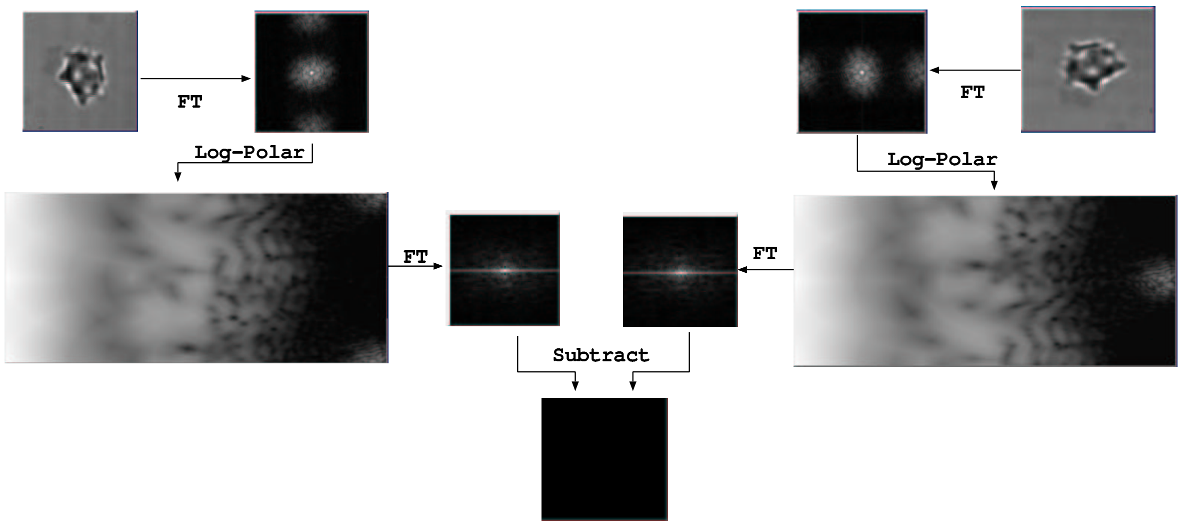


Figure 4.4: Creation of RST-invariant features: A rotation example. Note that the image rotation becomes a vertical shift in the log-polar plane [Dahmen & Hektor$^+$ 00].

Here u_0 and u_1 are the horizontal and vertical frequencies respectively and $(n_0^{(0)}, n_1^{(0)})$ is the position in the image where the frequencies are determined. The WFT is a convolution of the image with the filter m_{u_0,u_1}. For texture analysis, spatial and frequency locations are desired but with respect to the uncertainty principle of the Fourier transform a good trade off between these two goals has to be found. The Gabor transformation uses a Gaussian function as the optimally concentrated function in the spatial as well as in the frequency domain. Here, non-isotropic Gaussians of the form

$$\begin{aligned} m_{f,\varphi}(n_0,n_1) &= \frac{1}{2\pi\sigma^2\lambda} e^{-\frac{1}{2\sigma^2}\left(\frac{n_0'^2}{\lambda^2}+n_1'^2\right)} e^{2\pi i f n_0'} \\ M_{f,\varphi}(u_0,u_1) &= e^{-2\pi^2\sigma^2\left[(u_0'-f)^2\lambda^2+u_1'^2\right]} \end{aligned}$$

with the center frequency $f = \sqrt{u_0^2+u_1^2}$ and the rotated coordinates $(n_0', n_1') = (n_0 + \cos\varphi + n_1 \sin\varphi, -n_0\sin\varphi + n_1\cos\varphi)$ are used. $\frac{1}{\lambda}$ is the aspect ratio.

Due to the convolution theorem the filter interpretation of the Gabor transform allows the efficient computation of the Gabor coefficients $\mathcal{G}_{f,\varphi}(n_0,n_1)$ by multiplication of the Fourier transformed image $\mathcal{X}(u_0,u_1)$ with the Fourier transform of the Gabor filter $M_{f,\varphi}(u_0,u_1)$ and application of the inverse Fourier transform:

$$\mathcal{G}_{f,\varphi}(n_0,n_1) = FFT^{-1}\left\{\mathcal{X}(u_0,u_1)\cdot M_{f,\varphi}(u_0,u_1)\right\}$$

Since the correlation of information between the spectral bands is not integrated in the RGB color space it is proposed to use a special interpretation of the HSV color space.

A complex representation of the H and S color channel is used with $b(n_0,n_1) = S(n_0,n_1)\cdot e^{iH(n_0,n_1)}$ and the V layer is used as gray value image. These two image planes are Gabor transformed using a set of 25 Gabor filters accounting for different scales and directions. That is, in total 50 Gabor transformations are done, 25 for the V color channel and 25 for the b

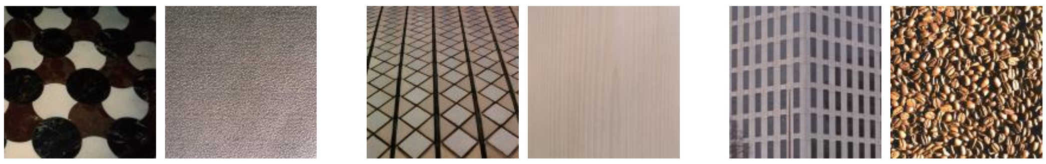


Figure 4.5: Example images for texture properties: a) high coarseness b) low coarseness c) high contrast d) low contrast e) directed f) not directed. (Images from [Graczyk 95])

color channel as a combination of the H and the S color channel. It is advantageous to use the convolution theorem as given above, which allows to calculate the coefficients efficiently.

After this process 50 Gabor coefficients for each pixel have been extracted which is a huge amount of data. To make this data manageable the 50 dimensional Gabor feature vector is extracted for a limited set of pixels only. Those pixels with high local variance are selected since we consider these pixels as important for the content of the image. This selection still results in a large set of 50 dimensional vectors. Since it is not possible to create a histogram of 50 dimensional data due to the high dimensionality, first a possibility to represent the 50 dimensional Gabor feature vectors more compactly is created.

To obtain this representation, first the set of 50 dimensional Gabor vectors is extracted from all images from the database, then a partitioning of this set of Gabor feature vectors is created with a fixed number of clusters using one of the clustering algorithms introduced in Chapter 2.2. Finally, for each image a histogram from the cluster memberships of its Gabor feature vectors is created. The clustering algorithms allow us to obtain a different number of clusters and thus differently sized histograms.

Other approaches to use Gabor features for image retrieval work very similar to the approaches for local features. Details about this are presented in Section 4.8.

4.6 Tamura Features

In [Tamura & Mori$^+$ 78] the authors propose six texture features corresponding to human visual perception: coarseness, contrast, directionality, line-likeness, regularity, and roughness. They make experiments to test the significance of the features. They found the first three features to be very important. That is, these correlate strongly with the human perception. Examples to illustrate the meaning of these features are given in Figure 4.5. These three features, coarseness, contrast, and directionality, are defined as follows:

Coarseness The coarseness gives information about the size of the texture elements. The higher the coarseness value is, the rougher is the texture. If there are two different textures, one macro texture of high coarseness and one micro texture of low coarseness, the macro texture is considered. The essence of calculating the coarseness value is to use operators of various sizes. A large operator is chosen when a coarse texture is present even if there is a micro-texture and a small operator is chosen when micro texture is present only. The coarseness measure is calculated as follows:

1. For every point (n_0, n_1) calculate the average over neighborhoods. The size of the neighborhoods are powers of two, e.g.: 1×1, 2×2, 4×4, …, 32×32:

$$A_k(n_0, n_1) = \frac{1}{2^{2k}} \sum_{i=1}^{2^{2k}} \sum_{j=1}^{2^{2k}} X\left(n_0 - 2^{k-1} + i, n_1 - 2^{k-1} + j\right)$$

2. For every point (n_0, n_1) calculate differences between the not overlapping neighborhoods on opposite sides of the point in horizontal and vertical direction:

$$E_k^h(n_0, n_1) = \left| A_k\left(n_0 + 2^{k-1}, n_1\right) - A_k\left(n_0 - 2^{k-1}, n_1\right) \right|$$

and

$$E_k^v(n_0, n_1) = \left| A_k\left(n_0, n_1 + 2^{k-1}\right) - A_k\left(n_0, n_1 - 2^{k-1}\right) \right|$$

3. At each point (n_0, n_1) select the size leading to the highest difference value:

$$S(n_0, n_1) = \operatorname*{argmax}_{k=1\ldots5} \max_{d=h,v} E_k^d(n_0, n_1)$$

4. Finally take the average over 2^S as a coarseness measure for the image:

$$F_{crs} = \frac{1}{N_0 N_1} \sum_{n_0=1}^{N_0} \sum_{n_1=1}^{N_1} 2^{S(n_0, n_1)}$$

Contrast In the narrow sense, contrast stands for picture quality. More detailed, contrast can be considered to be influenced by the following four factors:

- dynamic range of gray-levels
- polarization of the distribution of black and white on the gray-level histogram
- sharpness of edges
- period of repeating patterns.

The contrast of an image is calculated by

$$F_{con} = \frac{\sigma}{{\alpha_4}^z} \quad \text{with} \quad \alpha_4 = \frac{\mu_4}{\sigma^4}$$

where $\mu_4 = \frac{1}{N_0 N_1} \sum_{n_0=1}^{N_0} \sum_{n_1=1}^{N_1} \left(X(n_0, n_1) - \mu\right)^4$ is the fourth moment about the mean μ, σ^2 is the variance of the gray values of the image, and z has experimentally been determined to be $\frac{1}{4}$.

Directionality Not the orientation itself but presence of orientation in the texture is relevant here. That is, two textures differing only in the orientation are considered to have the same directionality.

To calculate the directionality the horizontal and vertical derivatives Δ_H and Δ_V are calculated by convolution of the image $X(n_0, n_1)$ with the following 3×3 operators respectively

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

and then for every position (n_0, n_1)

$$\theta = \frac{\pi}{2} + \tan^{-1} \frac{\Delta_V(n_0, n_1)}{\Delta_H(n_0, n_1)}$$

is calculated. These values are then histogramized in a 16 bin histogram H_D and the directionality can be calculated as the sum of second moments around each peak from valley to valley.

To be able to use these features for image retrieval, they are slightly changed. For each of these three features a per-pixel value is desired. To achieve per-pixel values, only the steps 1 to 3 are done in the calculation of the coarseness, resulting in a coarseness measure per pixel. The contrast is calculated in 13×13 neighborhoods for each pixel and the directionality is calculated pixelwise, too: Instead of the derivative filters, a Sobel filter is used and θ is calculated for each pixel denoting the direction of the area around this pixel. Now, three values are available for each pixel. One denoting the coarsenss, one denoting the contrast, and one denoting the directionality for the neighborhood of the pixels. These values are used in two different ways. First, we consider these values to be an RGB image and save it as such and second, a 3 dimensional histogram is created from these values.

A second reason for changing this method is that [Tamura & Mori$^+$ 78] do not make completely clear how to calculate the global directionality measure. The QBIC system [Faloutsos & Barber$^+$ 94] uses these features and the authors write that they also altered the processing steps slightly to obtain histograms describing the texture of the image.

4.7 Global Texture Descriptor

[Terhorst 03] describes a texture descriptor to characterize complete images. The descriptor consists of five parts, where each part models different properties of the texture of the image. The parts are

Fractal dimension measures the roughness or the crinkliness of a surface. In this work it is calculated using the reticular cell counting method[Rao 90, Haberäcker 95].

Coarseness characterizes the grain size of an image. Here it is calculated depending on the variance of the image.

Entropy is a measure of unorderedness or information content in an image. Entropy is a well-known measure from information theory.

Spatial gray-level difference statistics (SGLD) This statistics describes the brightness relationship of pixels within neighborhoods. It is also known as co-occurrence matrix analysis [Zucker & Terzopoulos 80, Haralick & Shanmugam$^+$ 73].

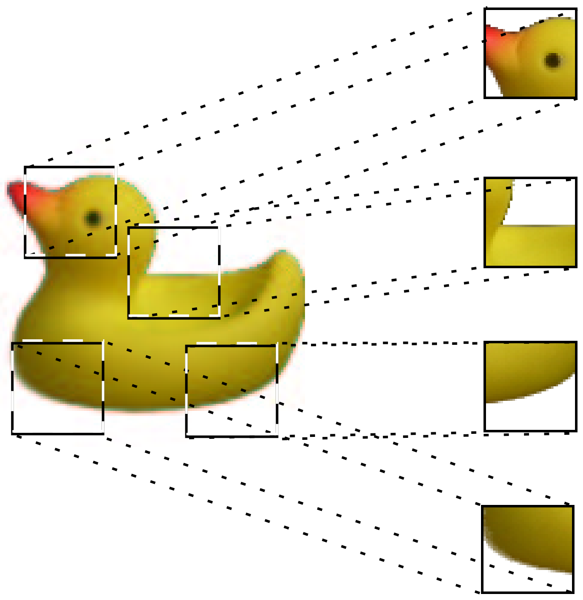


Figure 4.6: Extraction of local features.

Circular Moran autocorrelation function measures the roughness of the textures. For the calculation a set of autocorrelation functions is used. [Gu & Duncan+ 89].

In this work we obtain a 43 dimensional vector consisting of one value for the fractal dimension, one value for the coarseness, one value for the entropy and 32 values for the SGLD statistics, and 8 values for the circular Moran autocorrelation function. Details about the computation and the background can be found in [Terhorst 03]

4.8 Local Features

Local features are small square images taken from the original images. It is known, that local features can yield good results in classification tasks. Details about this can be found in [Paredes & Perez-Cortes+ 01, Kölsch 03]. Local features have some interesting properties for image recognition, e.g. they are inherently robust against translation. In the remainder of this section we briefly summarize how local features are used for classification and how this method is adopted for image retrieval.

Local features are small square images taken from the original image. Local representations might be of size $15 \times 15, 17 \times 17$ or larger. Many local features are extracted from one image. The number of local features extracted per image varies, but usually is between 100 and 1000. The positions from which the local features are extracted are usually determined by local variance of the gray values. That is, we assume that positions with high local variance are of some importance for the content of this image and thus extract a local feature from these positions. The feature extraction is depicted in figure 4.6.

The classification process with local features is a two step process: The training phase and the testing phase. In the training phase, local features are extracted from all of the training images, resulting in a huge amount of local features. To reduce the amount of data a PCA

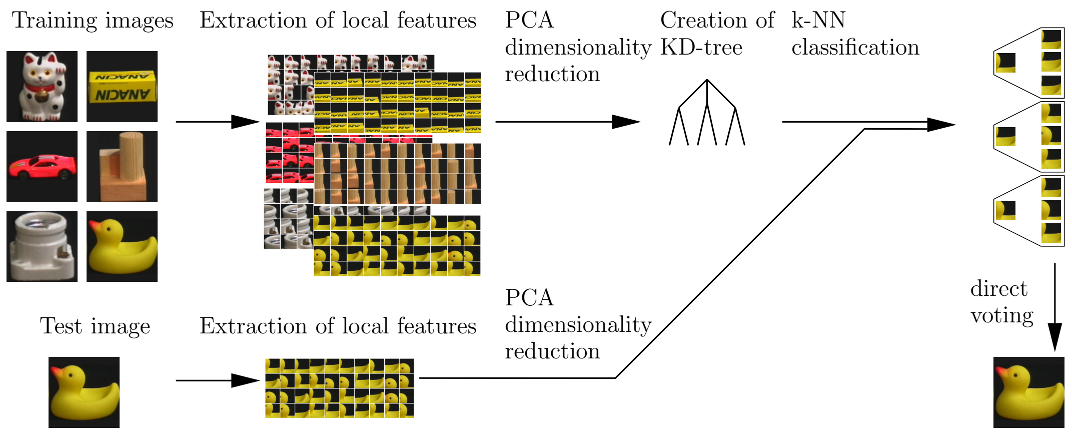

Training images
Extraction of local features
PCA dimensionality reduction
Creation of KD-tree
k-NN classification
direct voting
Test image
Extraction of local features
PCA dimensionality reduction

Figure 4.7: Classification using local features.

dimensionality reduction is applied where each local feature is reduced to be 40 dimensional. Then a KD tree is created from these local features to allow faster searching in the testing phase.

To classify a test image, first the local features are extracted in the same way as in the training phase, then the same PCA dimensionality reduction is applied. This step results in a set of local features. Each of these local features is then used to query the KD tree from the training phase for its nearest neighbors. That is, each of the local features is classified. These classification results are then used to obtain the global classification result by direct voting.

The whole classification process is depicted in Figure 4.7. Detailed research on variations of this method is reported in [Kölsch 03].

4.9 Histograms of Local Features

Using histograms of local features is motivated by the fact that for image retrieval good response times are required and this is hard to achieve using the huge amount of data incorporated in approaches using local feature queries. Here the amount of data is reduced by estimating the distribution of local features for each of the images.

To give an estimation of the distribution we create histograms of the local features. This is not possible in a straightforward manner, because even after the PCA dimensionality reduction the data still is 40 dimensional and even if each dimension is split into two bins only, a 2^{40} bin histogram would emerge which is not feasible. To circumvent this problem we apply a two step processing:

First a clustering algorithm is applied to a reasonably large set of local features. The obtained partitioning allows us to represent all local features by a cluster number, thus discretizing the local features by using a code book. To create a local feature histogram for an image, the local features are extracted and for each of the local features the cluster representing it best is determined. Then a histogram of these cluster memberships is created.

This process allows adjusting the amount of data easily. By creating many cluster centers a large histogram is yielded and by creating only a few cluster centers smaller histograms are yielded. The histogram has the same number of bins as the partitioning has clusters. This process is depicted in Figure 4.8.

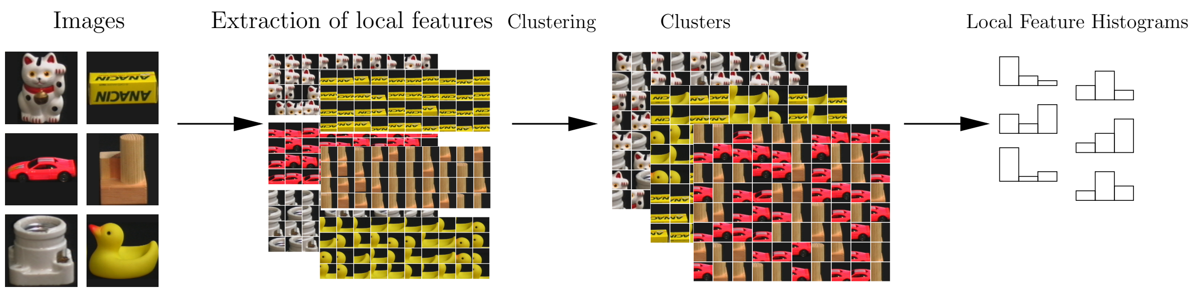


Figure 4.8: Creation of local feature histograms.

4.10 Region-Based Features

Another approach to represent images is based on the idea to find image regions which roughly correspond to objects in the images. To achieve this objective, the image is segmented into regions. The task of segmentation has been thoroughly studied [Pal & Pal 93] but most of the algorithms are limited to special tasks because image segmentation is closely connected to image understanding. Nevertheless, image segmentation is used in many applications and some image retrieval systems use image segmentation techniques [Wang & Li$^+$ 01, Carson & Thomas$^+$ 99].

Here we propose an image segmentation algorithm closely related to the methods from the BlobWorld system [Carson & Thomas$^+$ 99].

To segment an image we represent it as a set of pixels where each pixel is represented as a feature vector of 8 dimensions. The 8 dimensions are the R, G, and B value of the pixel, the coarseness, the contrast, and the directionality calculated for the pixel (cp. Section 4.6) and the x and y position of the pixel. Given this set of pixel vectors LBG-clustering is applied (cp. Section 2.2.2) and a partitioning of the pixels is obtained. From this partitioning a new image is created, where each pixel value is the number of the cluster to which it has been assigned. To this image a post processing algorithm is applied to delete noise. That is, a maximum vote filter over $k \times k$ windows with $k = 5, 10$, or 15 is applied to the image.

Given this partitioning, the average color, the average coarseness, the average contrast and the average directionality for each of the regions is calculated and saved together with the size of the regions. In Figure 4.9 some example images with different segmentations are given. The regions are depicted in their average color.

To compare images based on region descriptions, special comparison methods have to be used because images might contain different numbers of regions. Some approaches to comparing region based image descriptors are described in Chapter 5.4. A similar approach to segmentation has been proposed in [Carson & Thomas$^+$ 99] and the feature extraction code is freely available[1].

4.11 PCA Transformed Features

Principal component analysis (PCA) (also known as Karhunen-Loève Transformation) is a method for dimensionality reduction. Assuming data is high-dimensional it aims at reducing

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

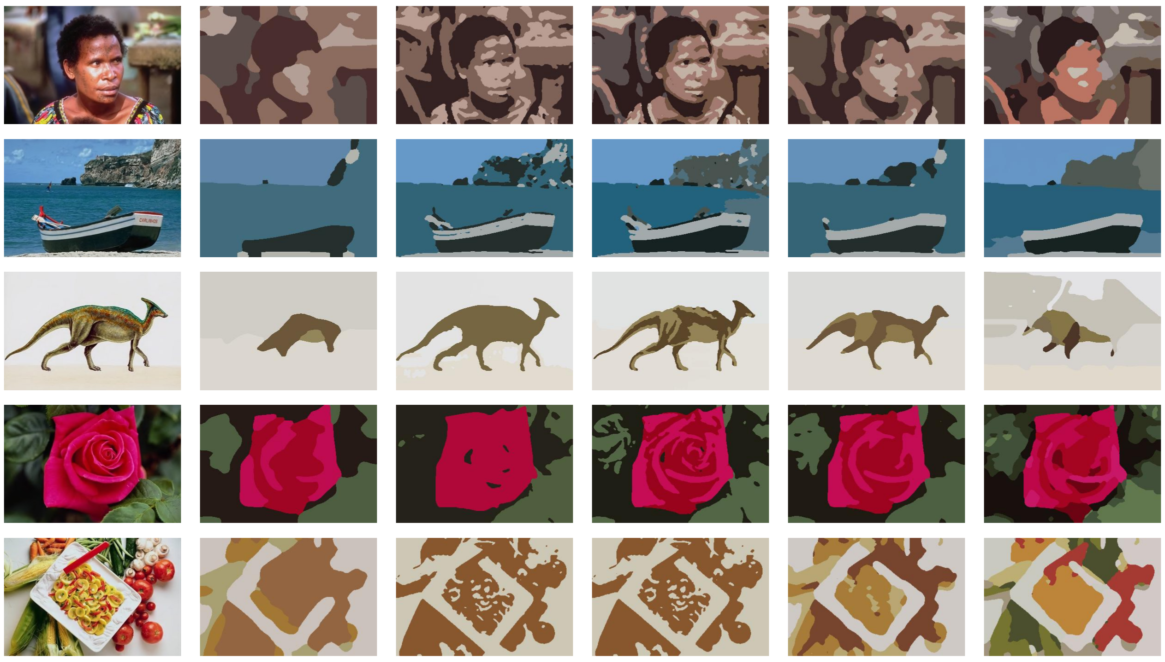


Figure 4.9: Example segmentations of images with varying number of regions and varying k for smoothing

the number of features. Unnecessary features, for example, might be strongly correlated with other features or constant. Feature reduction is a common problem in many branches of science like psychology and mathematics.

Principle component analysis is an unsupervised approach to extract the appropriate features from the data. To achieve this, first the D-dimensional mean vector μ and the $D \times D$ covariance matrix Σ are computed from the full data set. Then the eigenvectors and the eigenvalues are computed. In this work, the calculation of eigenvectors and eigenvalues is done using Singular Value Decomposition (SVD). The eigenvalues and eigenvectors are sorted according to decreasing absolute eigenvalue. Let v_1 be the eigenvector with eigenvalue λ_1, v_2 with eigenvalue λ_2, ..., v_D with eigenvalue λ_D. Then the k eigenvectors with largest eigenvalues are chosen to form a $k \times D$ matrix A. Using this matrix, the data vectors are projected to a k-dimensional subspace

$$x' = A(x - \mu).$$

This method can be applied to any type of data and we use it to reduce the dimensionality of the local features, the raw pixel data, as well as the different types of histograms.

4.12 Correlation of Different Features

Since we described several different types of features, we are faced with the problem of selecting an appropriate set of features to use for a concrete image retrieval task. Obviously, there are some similarities between different features. To detect these, we propose to create a distance matrix of a database using all the features available, that is, for each image all distances to all other images are calculated. For a database of N entries with M features each, an $N^2 \times M$ matrix D is created. From this matrix the covariance Σ is calculated as

$$\Sigma_{i,j} = E\left(D_i D_j - E(D_i D_j)\right)$$

where D_i and D_j denotes the distances between the i-th and j-th features respectively. From this $M \times M$ covariance matrix the correlation matrix R is calculated as

$$R_{i,j} = \frac{\Sigma_{i,j}}{\sqrt{\Sigma_{i,i}\Sigma_{j,j}}}.$$

The entries from this correlation matrix can be interpreted as similarities of different features. A high value $R_{i,j}$ means a high similarity in the distances calculated by features i and j, respectively. This similarity matrix R can be converted easily to a dissimilarity matrix $\mathcal{D}$ by

$$\mathcal{D}_{i,j} = 1 - |R_{i,j}|$$

and then visualized using multi-dimensional scaling.

Multidimensional scaling seeks a representation of data points in a lower dimensional space while preserving the distances between data points as good as possible. In this case the data is presented in a 2 dimensional space though it is easily possible to extend this methods to higher dimensions.

Many different possibilities to obtain such representations exist. These methods differ in the way they define which distances are being preserved. A basic version of multidimensional scaling is contained in PCA dimensionality reduction. There, the best possible projection into a subspace is searched. In this work a MatLab library which is freely available[2] is used for multi-dimensional scaling.

[2]http://www.biol.ttu.edu/Strauss/Matlab/matlab.htm

Chapter 5

Comparing Features

In Chapter 4 we introduced several different features to represent images in order to be able to decide whether two images are similar or not. In this chapter we introduce different dissimilarity measures to compare features. That is, a measure of dissimilarity between two features and thus between the underlying images is calculated.

5.1 Histogram Comparison Measures

Many of the features presented are in fact histograms (color histograms, invariant feature histograms, texture histograms, local feature histograms, and Gabor feature histograms). As comparison of distributions is a well known problem, a lot of comparison measures have been proposed and compared before [Puzicha & Rubner$^+$ 99].

5.1.1 Bin-by-Bin Comparison Measures

Bin-by-bin comparison measures for histograms can usually be computed very fast but due to the fact that they only compare bin-wise they cannot account for similarities between the underlying values of the bins. For example in Figure 5.1 the histograms a) and b) have the same distance as the histograms a) and c), but obviously a) and b) should be more similar if they were gray value histograms.

In the following, dissimilarity measures to compare two Histograms H and H' are proposed. Each of these histograms has M bins and H_m is the value of the m-th bin of histogram H.

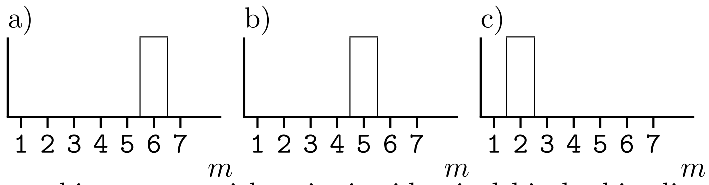


Figure 5.1: Three histograms with pairwise identical bin-by-bin distances although a) and b) should be more similar than e.g. a) and c).

Minkowski Distances

Minkowski Distances are a group of distance functions defined by

$$d_p\left(H,H'\right) = \left(\sum_{m=1}^{M}\left(H_m - H'_m\right)^p\right)^{\frac{1}{p}}.$$

The well known and widely used Euclidean distance is d_2 and has been proposed for computing dissimilarity scores between images. These distances are easily computable in $\mathcal{O}(M)$. This group of distances is not restricted to histograms. Other frequently used distances from the group of Minkowski distances are d_1, which is known as Manhattan distance, and d_∞, known as maximum distance.

Histogram intersection

Histogram intersection [Swain & Ballard 91] is a distance measure specially developed to compare histograms. It is intuitively motivated by calculating the common part of two histograms. It explicitly neglects features occurring in one histogram only. It is given by

$$d_\cap\left(H,H'\right) = \sum_{m=1}^{M}\min\left(H_m, H'_m\right)$$

and can be seen as a generalization of d_1 since when comparing histograms with the same bins the following holds:

$$d_\cap\left(H,H'\right) = 1 - \frac{d_1\left(H,H'\right)}{2}$$

Relative Deviation

Relative Deviation gives the deviation between two histograms.

$$d_{rd}\left(H,H'\right) = \frac{\sqrt{\sum_{m=1}^{M}\left(H_m - H'_m\right)^2}}{\frac{1}{2}\left(\sqrt{\sum_{m=1}^{M}{H_m}^2} + \sqrt{\sum_{m=1}^{M}{H'_m}^2}\right)}$$

Relative Bin Deviation

The Relative Bin Deviation is the bin-wise deviation between two histograms.

$$d_{rbd}\left(H,H'\right) = \sum_{m=1}^{M}\frac{\sqrt{\left(H_m - H'_m\right)^2}}{\frac{1}{2}\left(\sqrt{{H_m}^2} + \sqrt{{H'_m}^2}\right)}$$

χ^2-Distance

The χ^2-distance is a formal method to determine whether two distributions differ. To compare two histograms, we calculate:

$$d_{\chi^2}\left(H,H'\right) = \sum_{m=1}^{M}\frac{H_m - H'_m}{H_m + H_m}.x$$

Kullback-Leibler Divergence

The Kullback-Leibler divergence, which has its roots in information theory, is defined by

$$d_{KL}\left(H, H'\right) = \sum_{m=1}^{M} H_m \log \frac{H_m}{H'_m}.$$

It measures how inefficient it would be to code one histogram using the other. A problem with the Kullback-Leibler divergence is that it is neither symmetric nor numerically stable.

Jensen Shannon Divergence

The Jensen Shannon divergence, also referred to as Jeffrey divergence, is an empirical extension of the Kullback-Leibler divergence. It is symmetric and numerically more stable. It is given by

$$d_{JSD}\left(H, H'\right) = \sum_{m=1}^{M} H_m \log \frac{2H_m}{H_m + H'_m} + H'_m \log \frac{2H'_m}{H'_m + H_m}.$$

Fidelity Based Distance Measures

Another measure of distance between probability distributions is the so called fidelity. The fidelity is also known as Bhattacharyya distance in image processing:

$$d_F\left(H, H'\right) = \sum_{m=1}^{M} \sqrt{H_m}\sqrt{H'_m}$$

In [Nölle 03] the author proposes to use the following distance measures and points out that several of them are metrics:

$$\begin{aligned}
d_{\overline{F}}\left(H, H'\right) &= 1 - d_F\left(H, H'\right) \\
d_{\sqrt{1-F}}\left(H, H'\right) &= \sqrt{1 - d_F\left(H, H'\right)} \\
d_{\log(2-F)}\left(H, H'\right) &= \log\left(2 - d_F\left(H, H'\right)\right) \\
d_{\arccos F}\left(H, H'\right) &= \frac{2}{\pi} \arccos d_F\left(H, H'\right) \\
d_{\sin F}\left(H, H'\right) &= \sqrt{1 - d_F^2\left(H, H'\right)}
\end{aligned}$$

5.1.2 Cross-Bin Comparison Measures

Since the distance measures described so far neglect similarities between different bins of the histograms, even small changes in color or lighting conditions may lead to major changes in histogram distances. The measures described in the remainder of this chapter are developed to overcome this problem. Similarities between the underlying values represented by different bins are taken into consideration.

Quadratic Forms

Quadratic Forms are capable of considering the similarities between different bins by incorporating a matrix $A = (A_{m,n})$ with $A_{m,n}$ denoting the dissimilarity between the bins m and n.

Let H and H' be the histograms represented as vectors, then the Quadratic form can be calculated as

$$d_{\square}(H, H') = \sqrt{(H - H')^T \cdot A \cdot (H - H')}.$$

A high dissimilarity between the underlying values of different bins H_m and H'_n is denoted by a high value $A_{m,n}$, thus differences between these bins are taken into account stronger than differences between bins H'_m and $H'_{n'}$ where $A_{m',n'}$ is a low value. A common setting for the $A_{m,n}$ is

$$A_{m,n} = 1 - \frac{d_2(\mathcal{V}^m, \mathcal{V}^n)}{d_{\max}}$$

where $d_2(\mathcal{V}^m, \mathcal{V}^n)$ is the Euclidean distance between the values represented by bins m and n respectively and $d_{\max} = \max_{m,n} d_2(\mathcal{V}^m, \mathcal{V}^n)$ [Faloutsos & Barber$^+$ 94].

Earth Movers Distance

The Earth Movers Distance (EMD) [Rubner & Tomasi$^+$ 98] reflects the minimal amount of work that has to be performed to transform one distribution into the other by shifting portions of the distribution between bins. This is a special case of the transportation problem.

That is, computing the EMD requires a transportation problem to be solved. The EMD $d_{EMD}(H, H')$ between the histograms H and H' is calculated as

$$d_{\text{EMD}}(H, H') = \frac{\sum_{i,j} d_{i,j} g_{i,j}}{\sum_{i,j} g_{i,j}}.$$

Here $d_{i,j}$ denotes the dissimilarity between bin i and bin j and $g_{i,j} \geq 0$ is the optimal flow between the two distributions such that the total cost $\sum_{i,j} d_{i,j} g_{i,j}$ is minimized. The following constraints have to be taken into account for all i, j:

$$\begin{aligned} \sum_i g_{i,j} &\leq H'_j \\ \sum_j g_{i,j} &\leq H_i \\ \sum_{i,j} g_{i,j} &= \min(H_i, H'_j) \end{aligned}$$

A major advantage of the EMD is that each image may be represented by a histogram with individual binning.

Time Warp Distance

The time warp distance is deduced from the non-linear time alignment in speech recognition. In speech recognition time alignment is necessary to account for different speaking rates, to be able to determine the word and phoneme borders and to account for breaks in spoken

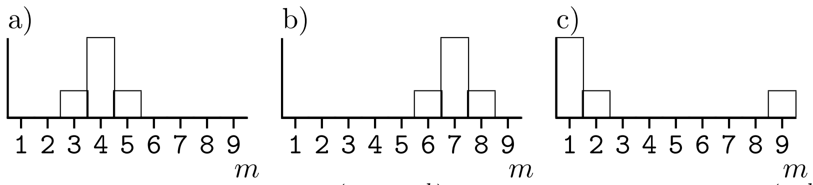
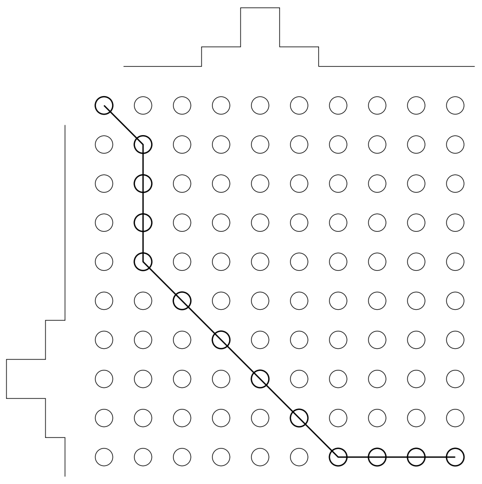
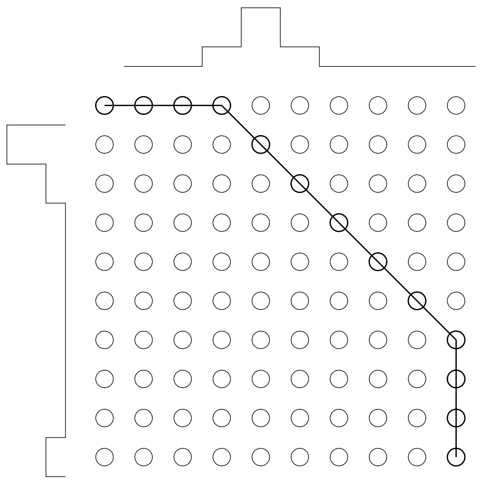


Figure 5.2: Time warp distances: $T\left(H^a, H^b\right) = 6$, $T\left(H^a, H^c\right) = 16$, $T\left(H^b, H^c\right) = 22$ in contrast to Euclidean distance, which is 46.9 for all three combinations.



(a) alignment graph between histogram a and b from Figure 5.2.



(b) alignment graph between histogram a and c from Figure 5.2.

Figure 5.3: Examples of alignment graphs.

language. Time alignment can be done linearly, that is, if the signals are of different lengths, the shorter one is stretched and then the alignment is done element wise. This method does not work well, to cope with this problem, a non-linear time alignment function has to be introduced. Non linear time alignment can be seen as an optimization problem: The optimal path, minimizing the differences between the two sequences has to be found. The search of the optimal path is done with dynamic programming. Details about this method can be found in [Ney 99].

We propose to use this method of aligning two different sequences to align histograms. Brightening or darkening an image shifts its histogram and changing the contrast stretches or compresses the histogram. Using the optimization method from non-linear time alignment it is possible to account for these operations. That is, a shifted histogram should not have a high distance to the original histogram. An example of different histograms and the distances between them is given in Figure 5.2. Three histograms are compared. Two of them are very similar and the third is completely different. The calculated bin-by-bin distance measures are pairwise identical. The calculated time warp distances reflect the similarities. Two example alignment graphs are given in Figure 5.3.

The time warp distance is the minimal distance between two histograms of all possible

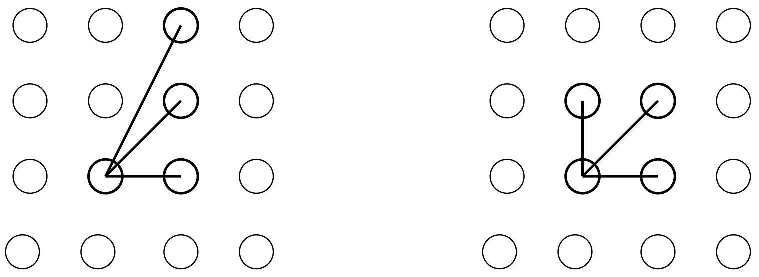

(a) $(0, 1, 2)$ model (b) $(-1, 0, 1)$ model

Figure 5.4: Deformation constraints for time warp distance.

alignments given some deformation constraints. For the time warp distance it is possible to select certain parameters. A penalty for a distortion can be selected and different distortion restrictions can be used. In this work we consider two different deformation constraints known as $(0, 1, 2)$-standard model and $(-1, 0, 1)$-model and each deformation is given the same penalty. The allowed distortions for these two models are depicted in Figure 5.4. For the time warp distance it is required that neighboring bins are similar and that the bins are ordered. That is, the similarity of a bin to other bins lowers with the distance in the histogram between the bins.

5.2 Comparing Images

Comparing images directly, that is comparing the values of the pixels of the image directly is quite often used in object recognition. Different methods have been proposed to do this and a selection of these methods is presented here and can be used in the image retrieval system which evolved from this work.

5.2.1 Euclidean Distance

Probably the most common approach to compare images directly is the Euclidean distance or other distances from the group of Minkowski distances. To be able to compare images using a Minkowski distance, the images have to be of the same size which can be achieved easily with scaling algorithms. The Euclidean distance has been used successfully e.g. in optical character recognition and has been extended by different methods.

5.2.2 Tangent Distance

Image objects are usually subject to affine transformations, such as scaling, translation, and rotation. The Euclidean distance is not able to account for such transformations if they are not part of the training corpus. Tangent distance [Keysers & Macherey^{+} 01] is an approach to incorporate invariance with respect to certain transformations into a classification system. Here, invariant means that image transformations that do not change the class of the image should not have a large impact on the distance between two images.

Let $X \in \mathbb{R}^D$ be a pattern and $t(X, \alpha)$ denote a transformation of X that depends on a parameter L-tuple $\alpha \in \mathbb{R}^L$. We assume that t does not change class membership (for small α). The manifold of all transformed patterns $\mathcal{M}_X = \left\{t(X,\alpha) : \alpha \in \mathbb{R}^L\right\} \subset \mathbb{R}^D$ now offers new possibilities for distance calculations. The distance between two patterns X and Q can be defined as the minimum distance between the two manifold $\mathcal{M}_X$ and $\mathcal{M}_Q$, which is truly invariant with respect to the regarded transformations:

$$d_t(X, Q) = \min_{\alpha,\beta\in\mathbb{R}^L} \left\{||t(X, \alpha) - t(Q, \beta)||^2\right\}$$

Since the distance calculation between these manifolds is a hard non-linear optimization problem it is necessary to find optimization techniques. In this case optimization is done using a tangent subspace approximation $\widetilde{\mathcal{M}}$. This subspace is spanned by a set of tangent vectors X_l that are the partial derivatives of the transformation t with respect to the parameters α_l. Thus, the transformation $t(X, \alpha)$ can be approximated using a Taylor expansion around $\alpha = 0$. The set of points consisting of all linear combinations of the tangent vectors X_l in the point X forms the tangent subspace $\widetilde{\mathcal{M}_X}$. This is a first-order approximation of $\mathcal{M}_X$.

The use of linear approximation allows to calculate the distances as a solution of a least square problem or projections into subspaces. Both are computationally inexpensive operations.

In optical character recognition the use of six affine derivatives and one derivative accounting for line thickness yields very good results. In other domains (e.g. radiograph recognition) line thickness is replaced by brightness.

5.2.3 Image Distortion Model

The image distortion model has been investigated earlier at the Lehrstuhl für Informatik IV of the RWTH Aachen [Keysers & Dahmen$^+$ 03] and further research is presented in [Gollan 03]. The image distortion model is an easily implemented method allowing for small local deformations of an image. Each pixel is aligned to the pixel with the smallest squared distance from its neighborhood. These squared distances are summed up for the complete image to get the global distance. To compare a query image Q with a database image X, $d(Q, X)$ is calculated as

$$d_{idm}(Q, X) = \sum_{n_0=1}^{N_0} \sum_{n_1=1}^{N_1} \min_{n'_0=n_0-w}^{n_0+w} \min_{n'_1=n_1-w}^{n_1+w} \left\{d'\left(Q(n_0, n_1), X(n'_0, n'_1)\right)\right\}$$

Here w is the warp range, that is the radius of the neighborhood in which a pixel may be chosen for alignment and d' is a pixel distance comparing the image pixels $Q(n_0, n_1)$ and $X(n'_0, n'_1)$. This method can be improved strongly by enhancing the pixel distance d' to compare sub images instead of single pixels only:

$$d'\left(Q(n_0, n_1), X(n'_0, n'_1)\right) = \sum_{x=-\omega}^{\omega} \sum_{y=-\omega}^{\omega} \left(Q(n_0 + x, n_1 + y) - X(n'_0 + x, n'_1 + y)\right)^2$$

Further improvement is achieved by using derivatives instead of the images directly. Intuitively, the use of derivatives makes the image distortion model align edges to edges and homogeneous areas to homogeneous areas.

In [Gollan 03] further methods for aligning images are proposed, but these are not considered due to the high computational complexity.

5.3 Comparing Images Based on Local Features

The method how local features are used for classification (cp. Section 4.8) is not directly transferable to image retrieval. In classification, for each class there is one large set of local features, but in image retrieval there is usually no class information. The aim is to find the most similar images.

To use local features for image retrieval at least three different methods are applicable: 1. direct transfer, 2. local features on a per image basis, 3. histograms of local features.

5.3.1 Direct Transfer

The method used in classification with local features is transferable to image retrieval. For this, local features are extracted from each of the database images, the PCA dimensionality reduction is applied and the KD tree is created. To query the database for an image, we extract the local features from the query image and apply the same PCA dimensionality reduction. Then the KD tree is queried for each of these query local features and for each of the database images we count how many of its local features have been found as a nearest neighbor to one of the query local feature. Finally the images from the database with the highest number of votes are returned. This method is equivalent to the method used in classification if each image is considered to constitute its own class.

5.3.2 Local Feature Image Distortion Model

Another possibility to use local features for image retrieval is based on the idea to take into account the distances between the local features from the images. That is, we calculate a distance $d\left(Q,X_n\right)$ between the image X_n and the query image Q represented by their sets of local features $\{x_{n1}\dots x_{nM}\}$ and $\{q_1\dots q_K\}$ respectively.

To calculate the distance $d_{lf}\left(Q,X_n\right)$ comparing the images X_n and Q we try to explain each of the local features from the query image Q using the local features from image X_n. That is, for each of the local features q_k from the query image, the nearest neighbor $\hat{x}_{nm}$ is searched from the set of local features $\{x_{n1}\dots x_{nM}\}$, the distances $d(q_k,\hat{x}_{nm})$ are calculated, and summed up to get the distance between the original images X and Q:

$$d_{lf}\left(Q,X_n\right)=\sum_{k=1}^{K}d(q_k,\hat{x}_{nm})$$

This method is closely related to the image distortion model (cp. Section 5.2.3) without any deformation constraints. The IDM considers all possible subimages and here only a subset is considered.

5.4 Comparing Region-Based Descriptions of Images

To compare images based on the regions occuring in the images the distance measures described up to here are not applicable since different images might contain a different number of regions and even if the images do have the same number of regions it is not obvious how to compare them: which region in the one image corresponds to which region in the other image is not clear, an appropriate alignment has to be found.

Different approaches to comparing regions based descriptions of images have been proposed. In the BlobWorld system [Carson & Belongie$^+$ 02] the user has to select a region and then images containing similar regions are searched. Here, the user gives additional information to the system. Since we want a full-automatic system, this approach is not applicable.

5.4.1 Integrated Region Matching

An approach called integrated region matching (IRM) [Wang & Li$^+$ 01] leads to good results. The IRM finds a matching from one set of regions to the other and is then able to compute a dissimilarity score. To find this matching, the IRM allows a region in one image to be matched with several regions in another image. Assume that image 1 and image 2 are represented by region sets $R_1 = \{r_1, \ldots, r_m\}$ and $R_2 = \{r'_1, \ldots, r'_n\}$. First an $m \times n$ distance matrix between these two region sets is computed. To compute the distance between the region sets R_1 and R_2, first a matching of all regions from image 1 to the regions of image 2 is created. To create this matching, significances s_i and s'_i respectively for each of the regions are calculated as fraction of the size of the region to the complete image. These significances denote the importance of the region in the image and a significance is assigned to each region alignment to denote the importance of this alignment with respect to the complete matching. Finally, the distance $d(R_1, R_2)$ is calculated as

$$d_{irm}(R_1, R_2) = \sum_{i,j} s_{i,j} d_{i,j}$$

where $d_{i,j}$ is the distance between the regions r_i and r'_j and $s_{i,j}$ is the according significance.

The matching is created in a greedy manner. That is, the regions that fit well are matched first: Assume $d_{i,j}$ is the smallest distance between any two regions, then the region pair r_i and r'_j is matched next. Then the significances of both regions are decreased by the minimum of these two significances:

$$s_i \leftarrow s_i - \min\{s_i, s_j\} \quad s'_j \leftarrow s'_j - \min\{s_i, s'_j\}.$$

The process ends when there is no $s_i, s'_j \neq 0$ left, that is, the regions of image 1 are completely matched to the regions of image 2, where each region might be partially matched to different regions of the other image.

This matching has the following advantages: It is easily computable and if the images compared are identical, the resulting distance is 0 because only identical regions are matched.

5.4.2 Quantized Hungarian Region Matching

Due to the greedy method, IRM does not always lead to the best result. Finding the best result is much more complicated, but with some quantization of the significances of the regions, it is possible to convert this matching problem into an assignment problem, which is computable in $\mathcal{O}(n^3)$, where n is the number of regions, using the Hungarian algorithm.

For the Hungarian algorithm to be applicable, a square distance matrix is required, but two images may have different numbers of regions. Also the Hungarian algorithm creates one-by-one alignments. As this constraint is not required and to create a square distance matrix the region descriptions the images are split such that each image is described by a

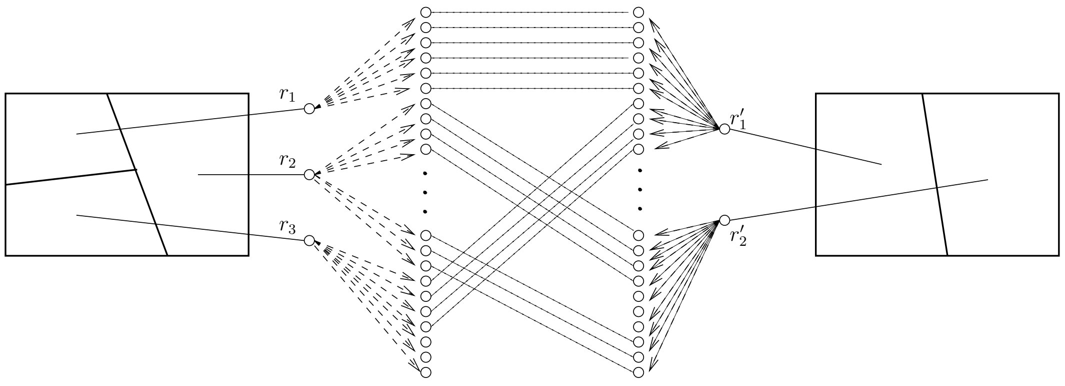


Figure 5.5: Region alignments for the quantized Hungarian region region matching.

total of 50 regions. In this step the region sizes (and thus the significances) are quantized to be multiple of 0.02. Thus, after the splitting, each image is described by a set of 50 region descriptions, where each description is of size 0.02.

Given these descriptions of two images, a 50×50 distance matrix D is created with $d_{i,j}$ is the distance between the regions r_i from image 1 and r'_j from image 2. To this matrix the Hungarian algorithm is applied resulting in the best possible one-by-one matching from the 50 regions from image 1 to the 50 regions from image 2. Intuitively, each of the initial regions from image one is partially mapped to one or more regions from image 2. This matching process is depicted in Figure 5.5. The image on the left hand is described by three regions. The image on the right hand is described by two regions. Each of these region sets is subdivided into 50 region descriptors and for the 50 region descriptors from image 1 the perfect matching to the 50 region descriptors from image 2 is found.

Given this matching, the quantized Hungarian region matchin distance is calculated by summing up the single distances as in the IRM

$$d_{qhrm}(R_1, R_2) = \sum_{i,j} s_{i,j} d_{i,j}$$

to obtain the total distance between image one and image two. Here, $s_{i,j}$ is set to 0.02.

5.5 Other Features

Up to here, we presented methods to compare histograms, images, region based descriptions, and local feature representations of images. Some features do not belong to any of these categories like PCA transformed features, invariant feature vectors and global texture features. To compare these features, we propose to use the Euclidean distance or the Mahalanobis distance to account for different domains of vector elements. Let Q be the N-dimensional vector for the query image and X be the N-dimensional vector for the database image to be compared, then the Mahalanobis distance between X and Q is calculated as the result from the quadratic form

$$d_M\left(Q, X'\right) = \sqrt{(Q-X)^T \cdot \Sigma^{-1} \cdot (Q-X)},$$

where Σ^{-1} is the inverted covariance matrix.

Chapter 6

Applications

This chapter gives an overview of the applications developed in this work and the techniques used. Since the aim of this work is to evaluate the quality of different features for image retrieval, we developed a large variety of feature extraction tools. Details about these tools can be found in Appendix A.

Here, we present the applications that use the features. Since the main goal was to evaluate features for image retrieval, the focus was put on the development of an image retrieval system capable of using all the features described in Chapter 4. Additionally, an application capable of clustering images according to similarities between certain features was developed.

6.1 Content-Based Image Retrieval

As the aim was to develop a content-based image retrieval system to test different types of features and dissimilarity measures, the main goal in the development was to create a highly flexible system capable of dealing with a large number of different features. To achieve these objectives the Flexible Image Retrieval Engine (FIRE) was developed. FIRE is a suite of two programs. A server and a web client which is to be run on a web-server. The server itself can be run on any computer.

If a query is performed from the web-interface the client sends the filename of the image to be queried to the server, the server loads this image and the features extracted for it and then compares these features to all the images from the database using the selected distance measures. Then the server sends the query result to the client, which presents the result to the user. In addition to the images resulting from the query some query performance measures are calculated and also presented to the user. Details about these performance measures are presented in Section 8.1.

After a query has been performed, the user may mark some of the results as “good” or “bad” and use these to query again. This is also known as *relevance feedback* and may improve the query result strongly [Müller & Müller$^+$]. Two versions of the query interface are available: one very simple interface which provides only the basic functions and a more sophisticated interface where nearly all settings of the server can be changed easily.

Apart from this interactive mode, several options to test the query performance of the image retrieval system have been implemented. For example, it is possible to query a database with each of its images in a leaving-one-out manner. That is, the database is queried with each image and the remaining images are used to calculate performance measures. The results are averaged over the whole database.

A screenshot of the web interface is given in Figure 6.1. At the top the settings for the query can be adjusted . It is a list of the features available for the database. For each feature the distance measure and the weight used can be modified. Also the number of results to be shown can be changed. In the Section “Results” the results are shown. The top-left image is the query image and the other images are its nearest neighbors. For each image some information is given: The image filename, the textual description, the class (if available), and a score. Also, the results may be marked as “good” or “bad” to allow for relevance feedback queries. Below the results a PR-graph is shown if available and some other performance evaluation measures are given as well. At the bottom random images are given to allow for new queries.

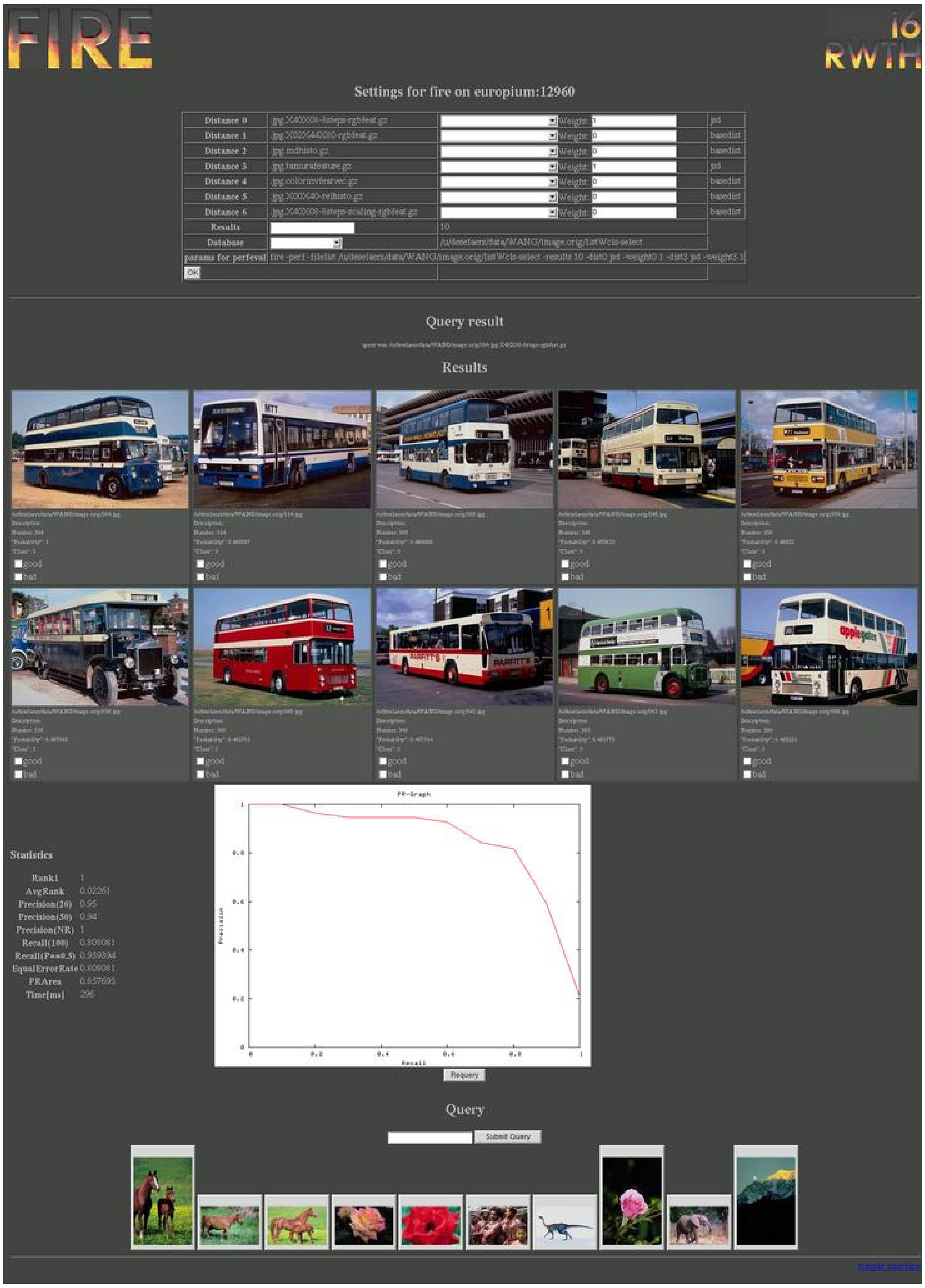
6.2 Grouping of Visually Similar Images

Image databases based on textual annotation are still quite common. A major drawback of this method is that annotations, the textual descriptions, are often ambiguous. Google recently offers a way to search for images based on textual information found in the context of images. For example, the search for “cookie” results in four different types of images: images of edible cookies or images of people eating cookies, screen-shots of programs dealing with cookies in the context of the Internet, and images not concerned with cookies at all. Even for words with less ambiguity nearly always two groups of images are returned: One group of images meeting the requirements and one group of images not suitable. To improve this situation we propose to use methods from computer vision to help the user reach his goals faster and more comfortably. Improvement can be achieved by using the features used for image retrieval examined in Chapter 4 in combination with clustering methods explained in Section 2.2.

An application providing these capabilities was developed, and additionally a small web interface for demonstration purposes has been created. The application provides access to nearly all features and distance functions introduced in Chapters 4 and 5 and LBG-clustering or k-means clustering can be chosen. In Figure 6.2 a screenshot of the application is given. At the top the used clusteralgorithm is selected. Below, the settings for the k-means and the LBG clustering algorithm can be selected. Also, the dissimilarity measure used and the database to be clustered can be selected. At the moment the 100 words Google image search was queried about are available here. Below this, the features to be taken into account can be selected. When all settings are chosen according to the user’s preferences, the clustering is started by hitting the “cluster” button. After the clustering process is finished the results are shown. Under the results the output of the clustering program is given for debugging purposes.

6.3 Classification

An extra application for classification is not necessary. Due to the strong connection between classification and image retrieval (cp. Section 8.1) all experiments concerning classification have been done using the FIRE framework in performance evaluation mode. The performance evaluation mode calculates the average precision for the first results over the performed queries $P(1)$ and this is exactly the recognition rate of a nearest neighbor classifier. From this follows that the error rate of a nearest neighbor classifier NN-ER can be calculated as NN-ER$= 1 - P(1)$. This allows for using all the features presented in Chapter 4 for classification.



Settings for fire on europium:12960

Distance 0	jpg.X40X08-8steps-rgbfeat.gz	Weight: 1		jsd
Distance 1	jpg.X02X44X00-rgbfeat.gz	Weight: 0		basedist
Distance 2	jpg.mdhisto.gz	Weight: 0		basedist
Distance 3	jpg.tamurafeature.gz	Weight: 1		jsd
Distance 4	jpg.colorinvfeatvec.gz	Weight: 0		basedist
Distance 5	jpg.X00X40-relhisto.gz	Weight: 0		basedist
Distance 6	jpg.X40X08-8steps-scaling-rgbfeat.gz	Weight: 0		basedist
Results		10		
Database		/u/deselaers/data/WANG/image.orig/listWcls-select		
params for perfeval	fire -perf -filelist /u/deselaers/data/WANG/image.orig/listWcls-select -results 10 -dist0 jsd -weight0 1 -dist3 jsd -weight3 1			

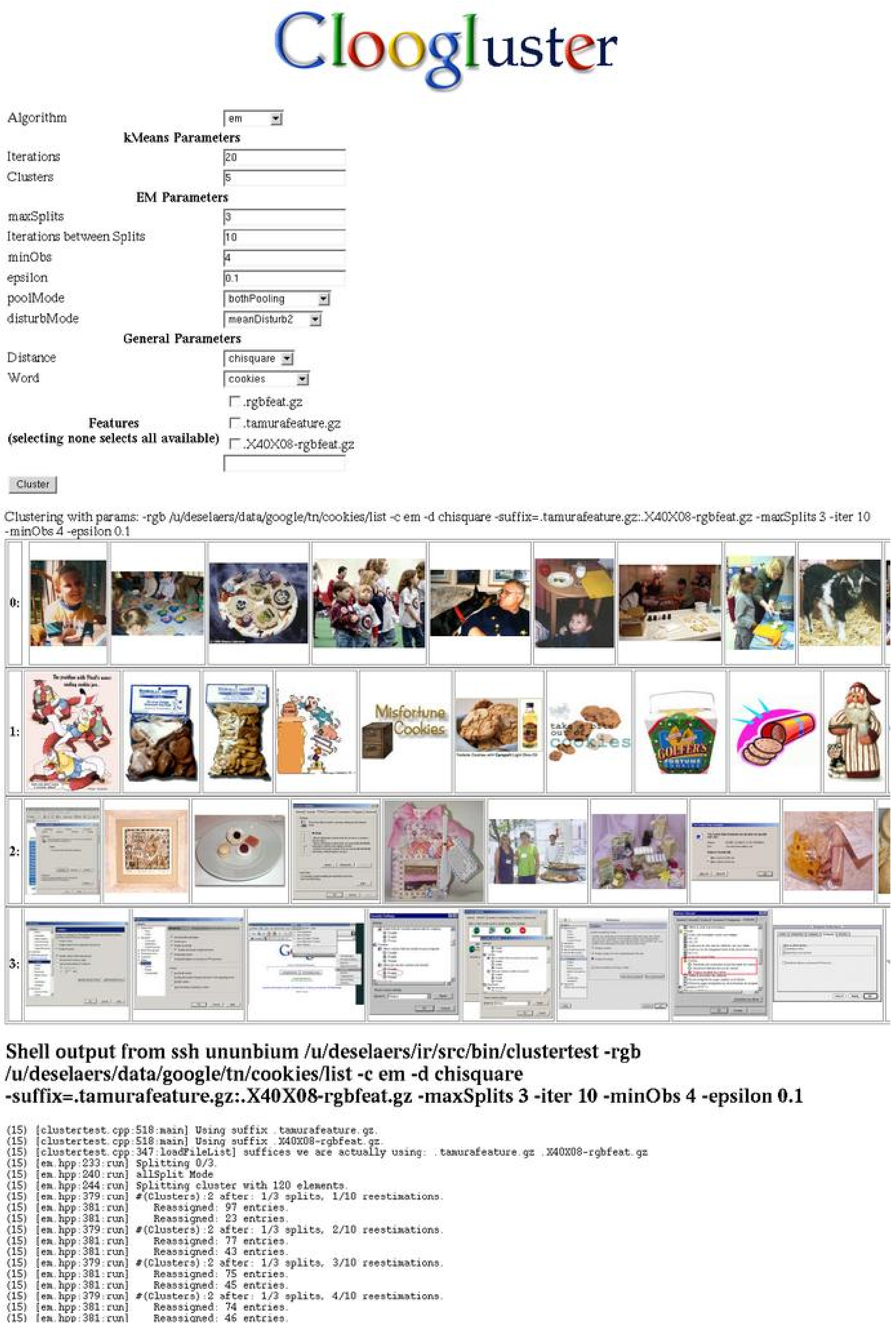
Query result

Results

Statistics

Rank1	1
AvgRank	0.02261
Precision(20)	0.95
Precision(50)	0.94
Precision(NR)	1
Recall(100)	0.808081
Recall(P==0.5)	0.989394
EqualErrorRate	0.808081
PRArea	0.857693
Time[ms]	296

Figure 6.1: Screenshot of FIRE.



Algorithm	em
kMeans Parameters	
Iterations	20
Clusters	5
EM Parameters	
maxSplits	3
Iterations between Splits	10
minObs	4
epsilon	0.1
poolMode	bothPooling
disturbMode	meanDisturb2
General Parameters	
Distance	chisquare
Word	cookies
Features (selecting none selects all available)	☐ .rgbfeat.gz ☐ .tamurafeature.gz ☐ .X40X08-rgbfeat.gz

Clustering with params: -rgb /u/deselaers/data/google/tn/cookies/list -c em -d chisquare -suffix=.tamurafeature.gz:.X40X08-rgbfeat.gz -maxSplits 3 -iter 10 -minObs 4 -epsilon 0.1

Shell output from ssh ununbium /u/deselaers/ir/src/bin/clustertest -rgb /u/deselaers/data/google/tn/cookies/list -c em -d chisquare -suffix=.tamurafeature.gz:.X40X08-rgbfeat.gz -maxSplits 3 -iter 10 -minObs 4 -epsilon 0.1

```
(15) [clustertest.cpp:518:main] Using suffix .tamurafeature.gz.
(15) [clustertest.cpp:518:main] Using suffix .X40X08-rgbfeat.gz.
(15) [clustertest.cpp:347:loadFileList] suffices we are actually using: .tamurafeature.gz .X40X08-rgbfeat.gz
(15) [em.hpp:233:run] Splitting 0/3.
(15) [em.hpp:240:run] allSplit Mode
(15) [em.hpp:244:run] Splitting cluster with 120 elements.
(15) [em.hpp:379:run] #(Clusters):2 after: 1/3 splits, 1/10 reestimations.
(15) [em.hpp:381:run]    Reassigned: 97 entries.
(15) [em.hpp:381:run]    Reassigned: 23 entries.
(15) [em.hpp:379:run] #(Clusters):2 after: 1/3 splits, 2/10 reestimations.
(15) [em.hpp:381:run]    Reassigned: 77 entries.
(15) [em.hpp:381:run]    Reassigned: 43 entries.
(15) [em.hpp:379:run] #(Clusters):2 after: 1/3 splits, 3/10 reestimations.
(15) [em.hpp:381:run]    Reassigned: 75 entries.
(15) [em.hpp:381:run]    Reassigned: 45 entries.
(15) [em.hpp:379:run] #(Clusters):2 after: 1/3 splits, 4/10 reestimations.
(15) [em.hpp:381:run]    Reassigned: 74 entries.
(15) [em.hpp:381:run]    Reassigned: 46 entries.
```

Figure 6.2: Screenshot of the clustering application.

Chapter 7

Databases

In this chapter we present the databases used to test the applications developed. Databases used for image retrieval as well as databases where clustering was applied and databases defining a classification task are introduced. Results other groups obtained using these databases are given along with the results we obtained on these databases in Chapter 8.

7.1 Corel

The Corel database is a large database of photographs of different scenes. It is widely used in the field of content-based image retrieval, but comparison based on this database is very difficult because the size of the database is enormous and different groups use different subsets of this database. The size of the database used varies between 30 000 in the BlobWorld system and 200 000 in the SIMPLIcity system. A subset of approximately 45 000 images was used for the experiments done during the course of this work. The images are 384×256 or 256×384 pixels each.

The database is annotated, but the annotation has been done by different persons and the quality of the annotation varies strongly between different images. Due to this, it is not possible to use any direct approach for performance evaluation. Some example images and their annotations are given in Figure 7.1. This figure shows that the annotations is questionable. In one image there is a mistake in the annotation ("perfomer" instead of "performer") and in another image there is one word written in two different ways ("grouper", "growper"). The annotation of the database consists of 184 988 words in total, 13 811 different words. That is, each image has 6.27 words of annotation in average, the maximum number of annotating words is 15 and the minimum is 1. That is, we have a corpus nearly 14 000 classes where each image is member of 6 classes in average.

7.2 WANG

The WANG database is a subset of the Corel database of 1000 images which have been manually selected to be a database of 10 classes of 100 images each. The images are subdivided into 10 classes such that it is almost sure that a user wants to find the other images from a class if the query is from one of these 10 classes. This is a major advantage of this database because due to the given classification it is possible to evaluate retrieval results. One example of each class can be seen in Figure 7.2. This database was also used for classification experiments.

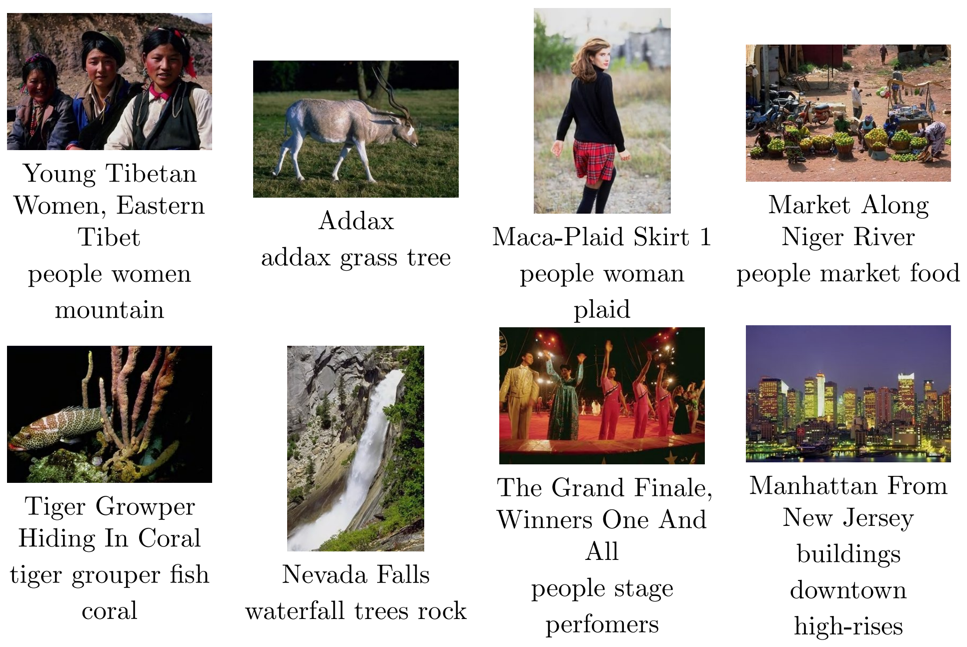


Figure 7.1: Example images from the Corel database.

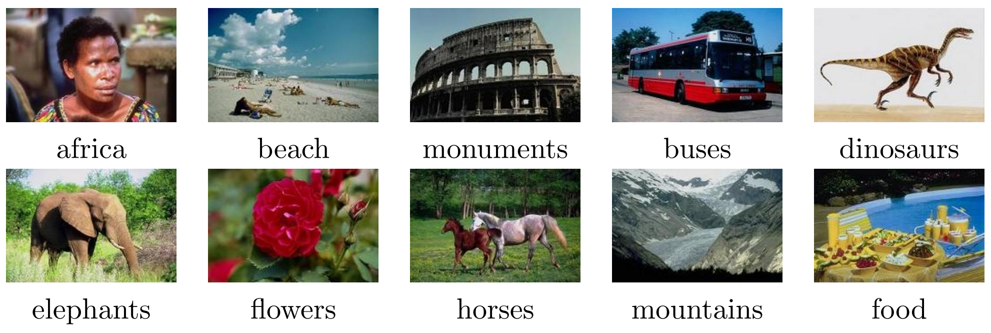


Figure 7.2: One example image from each of the 10 classes of the WANG database.

This database was used extensively to test the different features because the size of the database and the availability of class information allows for performance evaluation as can be seen in Section 8.1.

This database was created by the group of professor Wang from the Pennsylvania State University and is available for download[1]. Since this database is a subset of the Corel database, the images are of size 384×256 or 256×384 pixels as well.

7.3 Corel Subset

To have another corpus where performance evaluation is easily possible we created a database similar to the WANG database. 1000 images from 10 classes were selected from the Corel database. One example image for each class is shown in Figure 7.3.

[1]http://wang.ist.psu.edu/

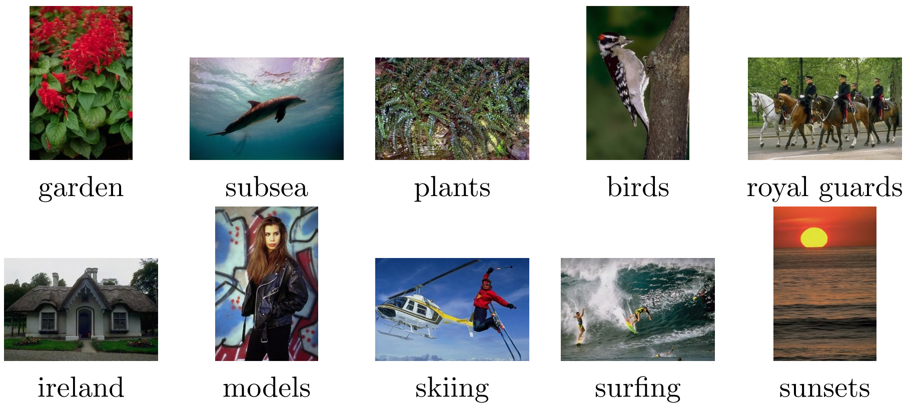


Figure 7.3: Example images from the Corel subset.

7.4 IRMA

IRMA (Image Retrieval in Medical Applications) is a database of currently 3879 medical radiographs, but it is still growing. In the future the database will consist of at least 10 000 images. The images were collected in a cooperation between the Department of Diagnostic Radiology, the Department of Medical Informatics, and the Lehrstuhl für Informatik VI of the Aachen University of Technology (RWTH Aachen) as part of the IRMA project[2]. For a long time the database consisted of 1617 images only and until now all experiments have been done using this older database. Here, we give all results for both of these databases to be able to compare them to older results, as this work is the first to experiment with the new 3879 images database. The 3879 images are divided into a train database of 2832 images and a test database consisting of 1016 images. Some images are left out since they are not classified. For classification, this database can be used either as a 26 classes task or as an 8 classes task. The 26 classes are a refinement of the 8 classes. The older 1617 images databases consisted of 6 classes. One example from each of these classes is shown in Figure 7.4. This database is basis of a very difficult task since images from the same class often are very different as can be seen in Figure 7.5. The images have been labelled by radiologists [Güld & Schubert$^+$ 03] and thus allow for performance evaluation in image retrieval tasks. The classes describe the body region of the image. The six classes are “abdomen”, “skull”, “chest”, “limbs”, “breast”, and “spine”. The 8 classes are “abdomen”, “skull”, “chest”, “lower limb”, “upper limb”, “pelvis”, “breast”, and “spine”. The 26 classes are a further refinement. Since this database is not freely available, only results from participants of this project are available.

7.5 CalTech Datasets

[Fergus & Perona$^+$ 03] use different datasets for unsupervised object training and recognition of objects. They classify whether an object is contained in the image or not. For this purpose they have several sets of images containing certain objects (motorbikes, airplanes, cats, cars, and leaves) and a set of arbitrary images not containing any of these objects. For performance evaluation of the image retrieval system some of their data sets were used. Most of the images

[2]http://www.irma-project.org

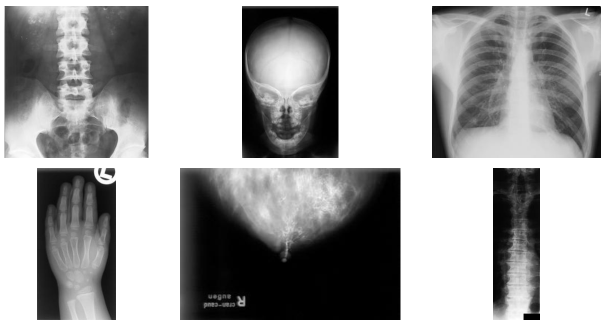


Figure 7.4: One image from each of the six IRMA-1617 classes: “abdomen”, “skull”, “chest”, “limbs”, “breast”, and “spine”.

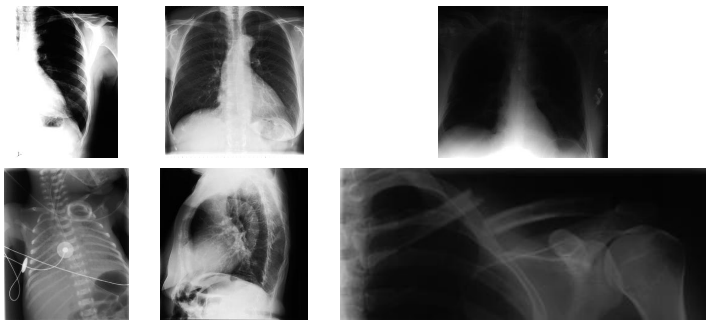


Figure 7.5: Several images from class “chest” from the IRMA-1617 database.

are actually color images but for the experiments the images are converted to gray scale. For classification, each of this datasets is split into disjunct training and a testing sets. The database is freely available online[3].

We use the motorbikes, the airplanes, and the faces, as well as the backgrounds. Example images are provided in Figure 7.6. For our experiments the given partition of the data into training and testing set was used. The training set was used as database in the image retrieval system and the testing set was used to query the system.

7.6 UW Database

The UW database consists of 1109 pictures. The images are photographs and have been created by the computer science department of the University of Washingten. The images are of different sizes, from 640×480 up to 883 × 589. There is no class information of the images but most of the images are annotated. The images which have not been annotated before were annotated in the course of this work. Additionally, the images are grouped in categories, e.g. “springflowers”, “barcelona”, and “iran”. In total there are 18 categories. Some example images with annotation are shown in Figure 7.7. The annotation consists of 6368 words from 353 different words. In average, each image has 5.74 words of annotation. The maximum number is 22, and the minumum is 1. The database is freely available[4].

[3]http://www.robots.ox.ac.uk/~vgg/data/index.html
[4]http://www.cs.washington.edu/research/imagedatabase/

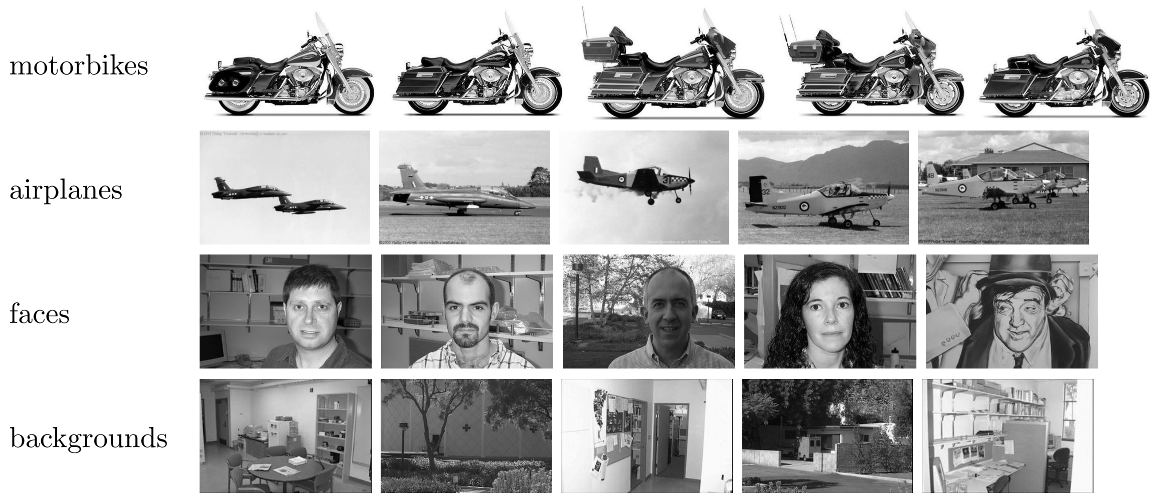


Figure 7.6: Examples of the CalTech datasets.

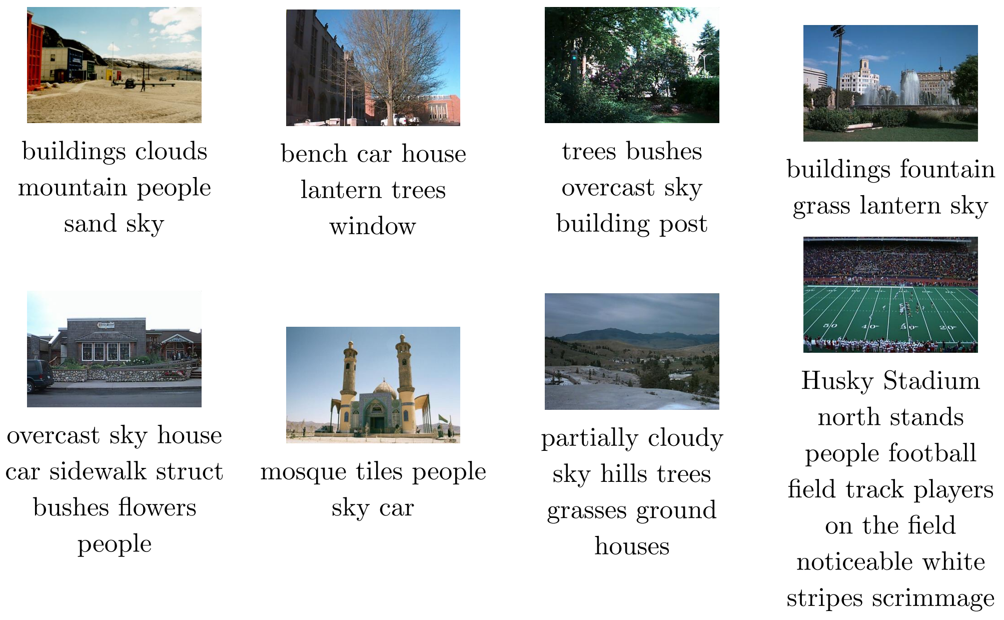
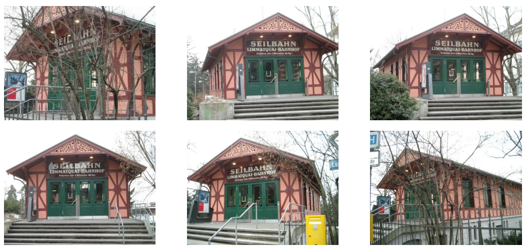

buildings clouds mountain people sand sky
bench car house lantern trees window
trees bushes overcast sky building post
buildings fountain grass lantern sky
overcast sky house car sidewalk struct bushes flowers people
mosque tiles people sky car
partially cloudy sky hills trees grasses ground houses
Husky Stadium north stands people football field track players on the field noticeable white stripes scrimmage

Figure 7.7: Examples from the UW database with annotation.

7.7 ZuBuD

The "Zurich Buildings Database for Image Based Recognition" (ZuBuD) is a database which has been created by the Swiss Federal Institute of Technology in Zürich and is described in more detail in [Shao & Svoboda+ 03b, Shao & Svoboda+ 03a].

The database consists of two parts. A main part consisting of 1005 images of 201 houses, 5 of each house and a query part of 115 images. Each of the query images contains one of the houses from the main part of the database. All pictures are of size 640 $\times$ 480. The pictures of each house are taken from different viewpoints and some of them are also taken under


(a)

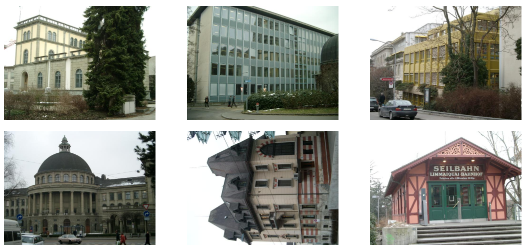

(b)

Figure 7.8: a) A query image and the 5 images from the same house in the ZuBuD-database b) 6 images of different houses in the ZuBuD-database.

Figure 7.9: Example images from the MPEG-7 test set.

different weather conditions and with two different cameras. To give a more precise idea of this database some example images are shown in Figure 7.8. The ZuBud database is freely available on the Internet[5].

7.8 MPEG-7 Test Set

The MPEG-7 Test set is a database of 2343 color images of size 376x256 or 256x376. It is not very widely used in image retrieval systems since it is not freely available. To our knowledge only the SIMBA system [Siggelkow & Schael$^+$ 01] uses this database. The database consists of photographs from various scenes. A large amount of the images are courtesy of the Department of Water Resources, California, USA. Some example images are depicted in Figure 7.9

In [Siggelkow 02] a set of 15 images was selected and for each of these images the relevant images have been selected manually from the database. For each of the 15 images between 3 and 14 images were selected as relevant. Quantitative results for queries using this knowledge about relevances are given and are compared to the methods presented here in Chapter 8.

7.9 Google

To test the clustering approach we created a new database of Google image search[6] results by querying Google image search with 100 English words and saving the first 120 thumbnails the search returned. This yielded a database of 12 000 images from 100 classes. In Figure 7.10

[5]http://www.vision.ee.ethz.ch/ZuBuD
[6]http://images.google.com

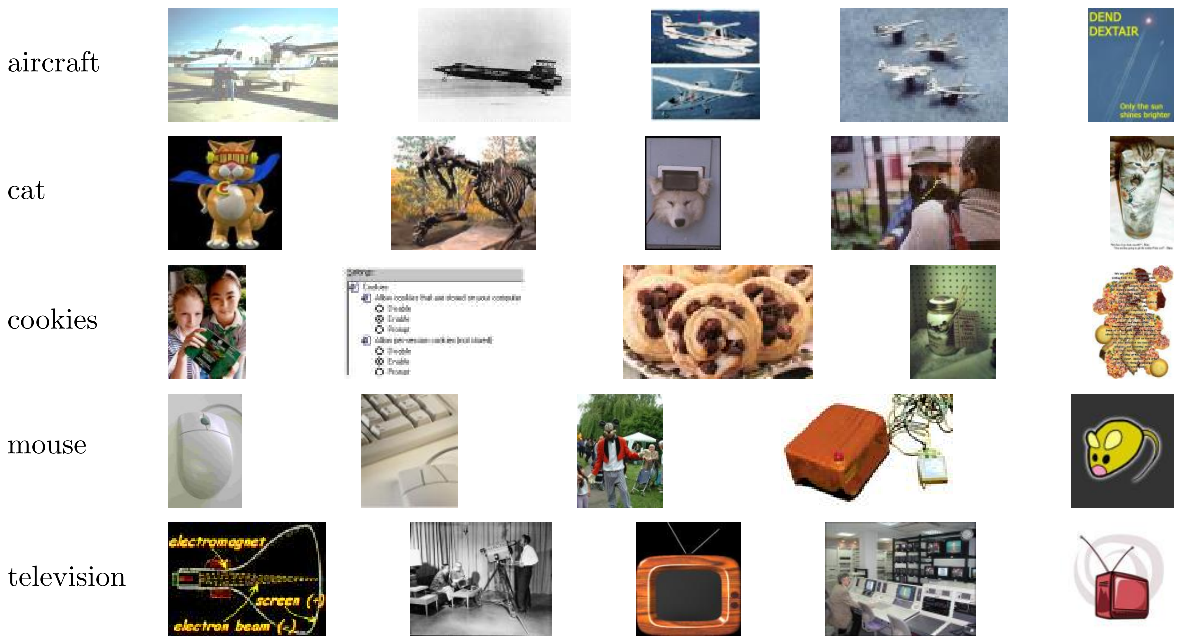


Figure 7.10: The first five results for five image searches on Google in March 2003.

the first five results for five different queries in google are presented. The saved images are of different sizes, the smallest image is 26×14 and the largest image is 150×150. The database was created in March 2003 and thus reflects the results from this date.

7.10 COIL

The Columbia Object Image Library (COIL) database is a well known database for image object recognition. Two different COIL databases are available: COIL-20 [Nene & Nayar$^+$b], which contains gray images from 20 objects, and COIL-100 [Nene & Nayar$^+$a] which contains color images from 100 objects. Both databases consist of images taken from different 3D-objects viewed from varying positions. Each image contains a single object subject to different illumination conditions. In COIL-100 there are 72 images from each object of size 128×128 pixels. All objects used for the COIL-100 database are depicted in Figure 7.12. The COIL databases are freely available[7].

To be able to compare our results to results obtained by [Käster & Wendt$^+$ 03], we use the subset of 20 objects of COIL-100 semantically matching the COIL-20 database. These 20 objects are depicted in Figure 7.12.

[7]http://www1.cs.columbia.edu/CAVE/research/softlib/

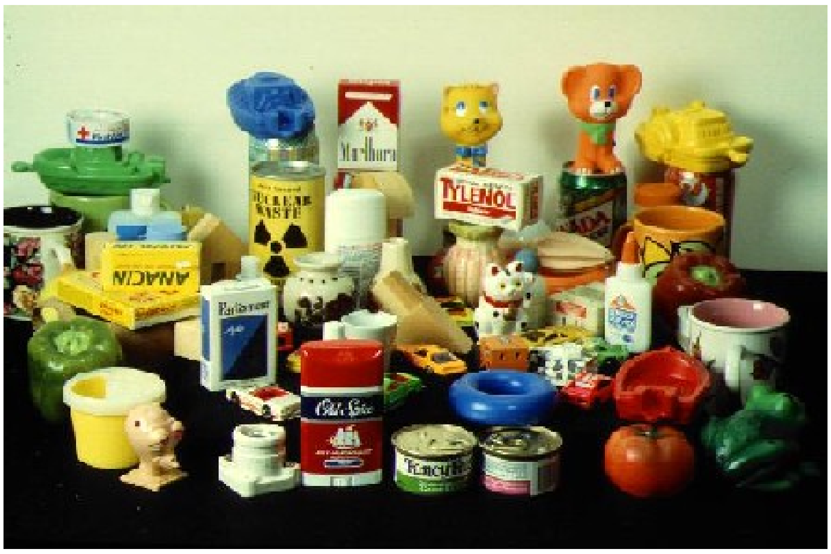


Figure 7.11: All objects from the COIL-100 database.

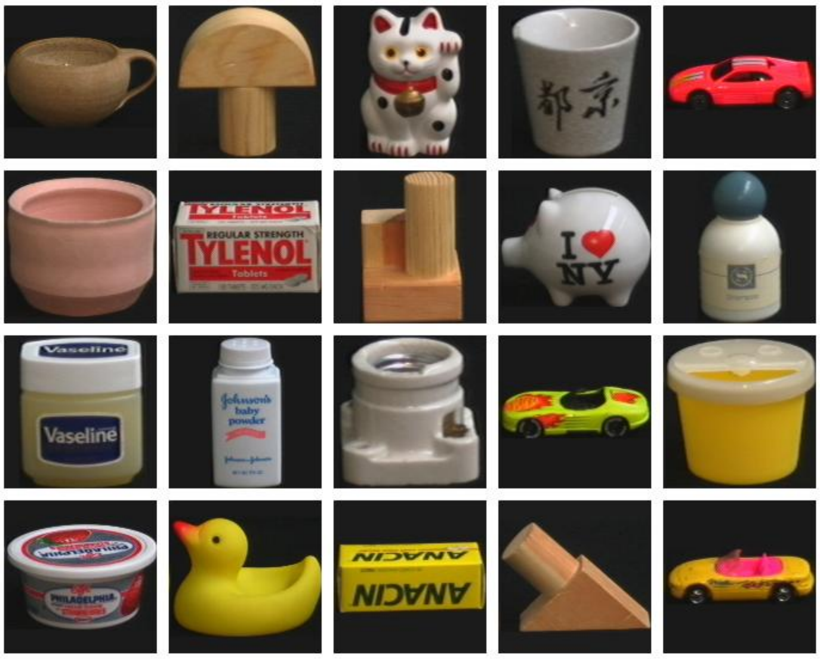


Figure 7.12: The 20 objects from the COIL-100 database corresponding to the objects of the COIL-20 database.

Chapter 8

Results

In this chapter the results for the image retrieval task, for the clustering task, and for classification tasks are presented. The results for image retrieval and classification are described in one section as these tasks are closely related. To be able to compare different image retrieval results, first a set of performance measures is introduced. For the tasks where comparable results from other works are available, these results are presented in this section and compared to the results obtained here.

8.1 Performance Evaluation for Content-Based Image Retrieval

In [Müller & Müller$^+$ 01] the authors investigate the possibilities of performance evaluation in content-based image retrieval. Since there is no common standard for performance evaluation for this task they propose a suitable set of performance measures for content-based image retrieval. Their work is based on the performance evaluation methods used at the Text REtrieval Conference (TREC). In this work this set of performance measures is adapted and extended by further performance measures. Also the correlation of different performance measures is analyzed.

The major problem in performance evaluation of content-based image retrieval systems is that neither a standard test database nor a standard performance measure is available. Thus, in early reports of content-based image retrieval systems, the results are often restricted to the presentation of retrieval results of one or more example queries, which is easily used to give a positive impression of the abilities of a system. Obviously, this is neither an objective nor a quantitative measure and it is impossible to compare systems based on example results alone. Apparently, an objective, quantitative performance measure is needed. The other problem is that no standard database is available for content-based image retrieval. That is, many image retrieval systems use different databases to present their results. Due to this it is impossible to compare the performance of different systems even if quantitative results are given.

In textual information retrieval, several performance measures are well established and it is reasonable to adapt some of them for content based image retrieval. Probably the most commonly used performance measures in information retrieval are the precision P and the recall R defined as

$$\begin{aligned} P &= \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}} \\ R &= \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}} \end{aligned}$$

These two values are often combined into a so called PR-graph. These figures show how many relevant and irrelevant documents are contained in the top ranked documents. PR-graphs are described e.g. in [Schäuble 97] and are created in several steps: In the first step it is assumed that the user inspects the n first documents of the ranked list. So for every $n = 1 \dots N$ precision P_n and the recall R_n is calculated. This results in a sawtooth curve because considering one additional image may raise the precision or lower it. In the next step this sawtooth curve is converted into a monotonic curve by setting

$$P_k := \max\{P_i \mid i \geq k\}.$$

This process can be interpreted as looking only at the locally optimal answer sets for which recall and precision cannot be improved simultaneously by inspecting further documents. In the last step the PR-graphs of different queries can be combined by calculating the arithmetic means of precision values corresponding to the same recall value. That is, some recall values are selected, the precision values corresponding to these values are calculated for each query and averaged. In the context of this work, the PR-graph is always evaluated for 11 recall values: $R = 0.0, 0.1, 0.2 \dots 0.9, 1.0$.

Since PR-graphs do not contain all the desired information and it is not clear how to compare image retrieval systems looking at graphs, [Müller & Müller$^+$ 01] propose to enhance the PR-graph with a set of other performance measures that are defined in the following: Rank_1, $\widetilde{\text{Rank}}$, $P(20), P(50), P(N_R)$, $R(P = 0.5), R(100)$. We propose to enlarge this set by some further performance measures: $P(P = R), P(1)$, NN-ER, and PR-area. In the following, the performance measures proposed are explained briefly:

$\mathbf{Rank_1}$. rank at which the first relevant image is retrieved

$\widetilde{\mathbf{Rank}}$. normalized average rank of relevant images

$$\widetilde{\text{Rank}} = \frac{1}{NN_R}\left(\sum_{n=1}^{N_R} R_i - \frac{N_R(N_R-1)}{2}\right)$$

where R_i is the rank of the ith relevant retrieved image and N_R is the total number of relevant images. This measure is 0 for perfect performance and approaches 1 as performance drops. For random retrieval its expected value is 0.5.

$\mathbf{P(20), P(50), P(N_R)}$. precision after 20, 50 and N_R images retrieved

$\mathbf{R(P = 0.5), R(100)}$. recall at Precision $P = 0.5$ and Recall after 100 images retrieved

$\mathbf{P(P = R)}$. precision where Recall equals Precision

PR-area. the area below the PR-graph

P (1). precision of the first image retrieved; Averaged over a whole database this is the same as the recognition rate of a nearest neighbor classifier using the same features and distance measures. Thus, the error rate of a nearest neighbor classifier can be calculated as NN-ER$= 1 - P\,(1)$.

Given this set of performance measures it is possible to compare content-based image retrieval systems quantitatively given a database where the relevance of the images is known with respect to some queries. Usually relevances of images are not known, thus it is not possible to calculate these measures for queries.

Databases where relevances are known are, for example, the IRMA database and the WANG database. In both cases, classes are given and we can assume that a user querying with an image from one class is looking for other images from the same class. For the WASH database, relevances are not directly clear and assumptions have to be made to extract relevances from the descriptions of the images. We assume that an image is relevant with respect to a query image if the descriptions of the images contain identical words. To make this relevance estimation more robust to inconsistencies in annotations the porter stemming algorithm [Porter 80] is used. This algorithm removes word endings leading to the same result e.g. for "house" and "houses". This stemming is important since annotation was done by several persons in most cases. After stemming is performed, the intersection of the sets of words is calculated and if there is one stemmed word in both of the descriptions, we assume that the images are relevant with respect to each other.

To be able to compare the performance of image retrieval system more easily it would be preferable to have a single performance measure. Since all the measures are supposed to measure the quality of the retrieval system in one way or another it is obvious to assume a strong correlation. However, to our knowledge so far no quantitative analysis for this assumption was made. Here we present such an analysis using the WANG and the IRMA databases. A correlation matrix for a selection of performance measures is given in Figure 8.1. This matrix shows that the performance measures are indeed strongly correlated and thus to compare image retrieval systems it should be sufficient to use only one performance measure. A very strong correlation can be seen between the neighboring values from the PR-graph. Values not so strongly correlated to the other measures are $R(P = 0.5)$ and $P(R = 1)$. This is due to the fact that both values do not reflect the quality of the first images retrieved but, especially $P(R = 1)$, consider the last images retrieved. For a user searching for images it is usually more important to find suitable images fast than to find the last suitable image. If a more detailed analysis is needed or a specified property of a system needs to be measured it is advisable to consider more or all of these performance measures. In this work, the error rate (ER) is selected as performance measure to compare the different settings as it is a well established performance measure in classification tasks and the average absolute correlation from $P(1)$ to the other performance measures is 0.83.

8.2 Results for Content-Based Image Retrieval

Since we investigated a lot of features and a lot of distance functions we are interested in the discrimination performances of these: Which features/dissimilarity measures lead to good results and which do not.

First, different distance measures for the different types of features are compared. Afterwards different features are compared using the best distance measures according to the

	P(1)	ER	P(50)	R(P=.5)	R(100)	Rank1	$\widetilde{\text{Rank}}$	P(R=P)	PR-area	P(R=0)	P(R=0.1)	P(R=0.5)	P(R=0.9)	P(R=1)
P(1)	100	-100	94	62	94	-91	-90	86	90	98	95	83	73	31
ER		100	-94	-62	-94	91	90	-86	-90	-98	-95	-83	-73	-31
P(50)			100	57	91	-88	-87	77	81	93	93	73	62	14
R(P=0.5)				100	48	-58	-54	55	58	61	48	55	52	21
R(100)					100	-96	-97	94	96	97	95	92	84	50
Rank1						100	99	-94	-95	-96	-86	-92	-85	-54
$\widetilde{\text{Rank}}$							100	-97	-97	-96	-88	-95	-89	-59
P(R=P)								100	99	92	85	99	95	72
PR-area									100	95	88	98	94	67
P(R=0)										100	94	89	81	42
P(R=0.1)											100	82	72	31
P(R=0.5)												100	96	76
P(R=0.9)													100	81
P(R=1)														100

Figure 8.1: Correlation of performance measures scaled to $[-100, 100]$.

experiments before. The experiments are done on the WANG and the IRMA-1617 corpus since the size and the given classification allow for extensive performance evaluation. The insights obtained on these databases are later transferred to other image retrieval tasks.

8.2.1 Comparison of Different Distance Functions

In this section different dissimilarity measures for the different types of features are compared experimentally and the performance for the different types of features is quantitatively measured. The experiments were done using the WANG and the IRMA-1617 corpus. First comparison measures for histograms, then the different distance measures for images, the different methods to compare images based on local features, and finally the different comparison measures for region based descriptions of images are analyzed and compared.

Different Comparison Measures for Histograms

Exemplary results (error rates) obtained using different comparison measures for histograms are shown in Table 8.1 for the WANG database and for the IRMA-1617 database. For the WANG database an invariant feature histogram with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$ was used and for the IRMA database a local feature histogram with 512 bins of 5×5 local features was used. Complete tables can be found in Appendix B in Tables B.4 and B.5. Corresponding PR graphs are shown in Figure 8.2. The results show that L_1 distance, χ^2 distance and Jensen Shannon divergence yield very similar results and clearly outperform the widely used Euclidean distance.

Since the time warp distance was not investigated before, the results obtained using the time warp distance with different parameters are presented here in more detail. The time warp distance has three parameters: different models for the deformation constraints are

Table 8.1: Error rates [%] on WANG and IRMA-1617 using different comparison measures. For the WANG database an invariant feature histogram with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$ and for the IRMA-1617 database a local feature histogram was used.

WANG		IRMA-1617	
Distance measure	ER[%]	Distance measure	ER[%]
JSD	15.9	L_1	8.3
χ^2	16.5	χ^2	9.1
L_1	18.4	JSD	9.3
Euclidean	28.3	Euclidean	14.2

Table 8.2: Error rates [%] using time warp distance on the IRMA-1617 database with an invariant feature histogram with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$. The last column gives the corresponding error rate without warping.

Distance	ER[%]	base ER[%]
L_1	34.7	35.6
L_2	41.1	42.4

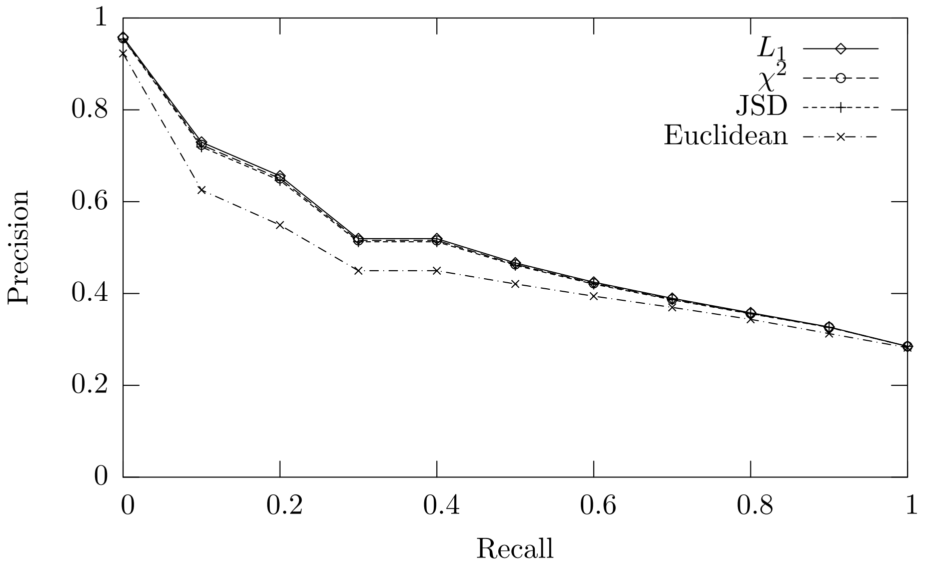
available, the underlying local distance can be varied, and the deformation penalty can be varied.

In Figure 8.3 the results for different parameter settings for the time warp distance are shown comparing invariant feature histograms with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$. This type of histogram does not lead to good results for the IRMA database but the time warp distance is not applicable to every type of histogram as it requires an ordering of the bins (cp. Section 5.1.2), e.g. the time warp distance is not applicable to local feature histograms. Figure 8.3(a) gives error rates for different penalties and different deformation constraints using an L_1 type distance and Figure 8.3(b) gives error rates using different penalties and different deformation constraints using an L_2 type distance. The results improve using the time warp distance. In both cases the choice of the deformation model is not very important but the penalty is a sensitive parameter. In both cases the deformation penalty has to be within a certain range to obtain good results. If the penalty is chosen too low (e.g. "penalty=0"), the results are quite bad, due to the fact that unsimilar histograms can be aligned too good. The fact that L_1 distance outperforms Euclidean distance was observed earlier in this work for histograms. Both graphs show that the time warp distance improves the results compared to the underlying base distance. The best error rates in comparison to the error rates obtained with the base distances are given in Table 8.2.

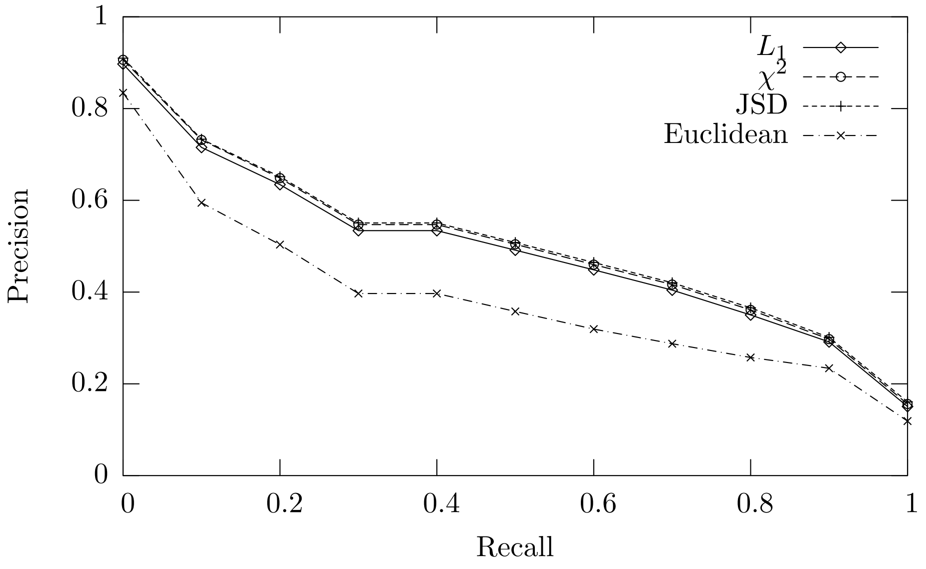
Different comparison measures for images

For comparing images directly, the Euclidean distance, the image distortion model, and the tangent distance were proposed. Table 8.3 shows results for these three comparison measures for the WANG and IRMA-1617 databases.

The results show that in both cases the image distortion model yields the best and the simple Euclidean distance the worst results. However, it can be seen clearly that taking the



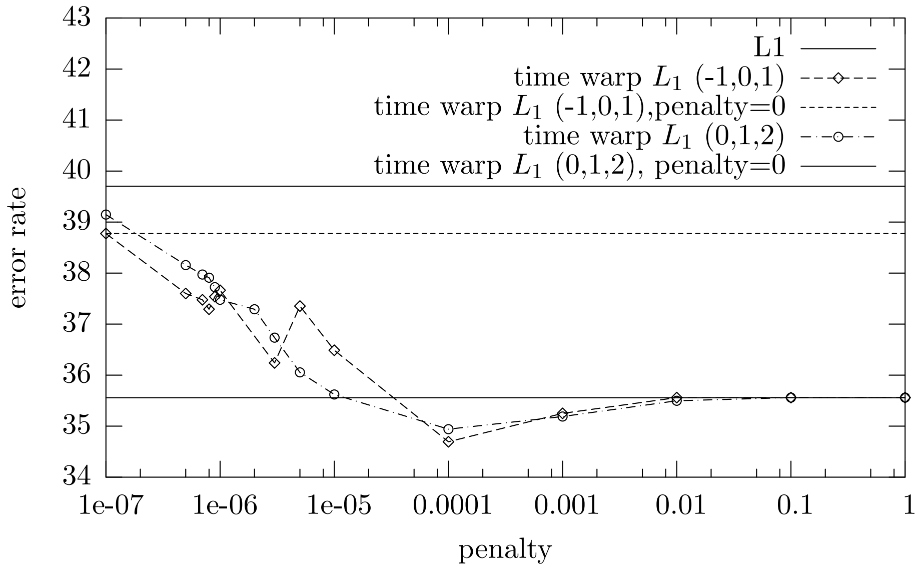
(a) PR graphs for IRMA-1617 using different distance measures.



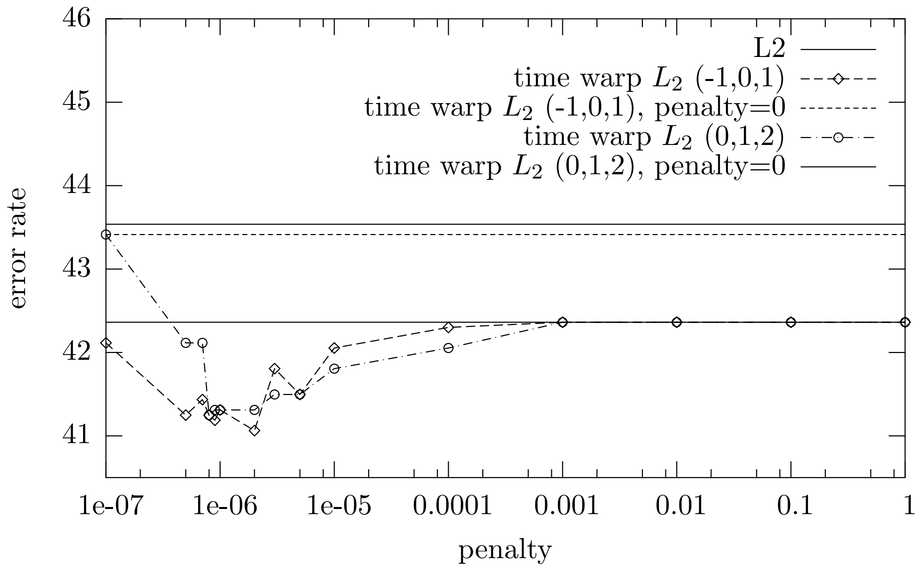
(b) PR graphs for WANG using different distance measures.

Figure 8.2: PR graphs for WANG and IRMA-1617 using different distance functions.

images into consideration for retrieval is advisable for the IRMA task but not advisable for the WANG task. This leads us to the conclusion that taking into account the images as features directly is suitable in more specialized tasks, like the IRMA task, whereas it is not suitable in less restricted tasks as the WANG task. We assume that this is due to the fact



(a) Error rate vs. penalty for L_1 base distance.



(b) Error rate vs. penalty for L_2 base distance.

Figure 8.3: Different parameters for the time warp distance on the IRMA-1617 corpus with an invariant feature histogram with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$.

that the IRMA images usually contain one object (e.g. arm, leg, skull) from a limited set of objects (body regions) whereas most images from the WANG corpus contain several objects with varying positions and the variation in the appearance of different objects from the same

Table 8.3: Error rates [%] for the IRMA-1617 database using different image comparison measures.

Method	IRMA	WANG
Euclidean	17.7	55.1
IDM with thresholding	6.7	[Gollan 03] 22.1
Tangent distance	[Keysers & Dahmen+ 03] 13.1	53.7
Tangent distance with thresholding	[Keysers & Dahmen+ 03] 11.1	-

Table 8.4: Error rates [%] using region based features and different comparison measures for the WANG database.

comparison measure	ER[%]
IRM	55.30
RM	49.80

type is very high. (e.g. old fashioned bus vs. modern red bus, children playing on the beach vs. beach without any person).

Different comparison measures for region based descriptions of images

In Section 5.4 we proposed two methods for comparing region based descriptions of images. Table 8.4 gives the results for these two methods using region based descriptions for the WANG database. The region matching method using the Hungarian algorithm and thus obtaining the perfect matching between the region descriptions is clearly better than the greedy integrated region matching. Nonetheless, these error rates are not satisfactory and probably are so bad due to the fact that segmentation still is an unsolved problem.

Different comparison measures for local features

In Section 5.3 we proposed three different methods for comparing images based on local features. Here we compare results for these methods and additionally compare the results to those obtained using the image distortion model with differently sized pixel windows.

Table 8.5 shows error rates for the IRMA database for different settings for local features as well as for the image distortion model. We consider this comparison interesting since in both cases subimages are compared. The image distortion model usually compares relatively small subimages and takes deformation constraints into account. The local feature based approaches usually compare relatively large subimages and no deformation constraints are considered. In the usual local feature based approaches even alignments across images of the same class are allowed. We present the results for histogram representation of local features (cp. Section 4.9). The results show that using stronger constraints the results get better with smaller subimages: the weakest constraints are in the method labeled "lf-1l1o". "lf-1l1o" is the common local feature approach without any constraints. For each local feature the nearest neighbor from the large set of database local features is searched and it is counted how many local features from which class are found. In the decision process it is not important from which image the local feature originates. The method labeled "glfd" incorporates slightly stronger constraints, which is the direct transfer from local features to image retrieval. Here,

Table 8.5: Comparison of error rates [%] on IRMA using local features and image distortion model.

	Method	19×19	5×5	3×3
local features	lf-l1o	13.0	28.8	31.7
	glfd	26.0	22.9	32.8
	lfd	26.8	23.1	28.4
image distortion model	IDM (sobel)	13.3	8.7	6.7
	IDM	11.5	11.4	12.7
local feature histograms	with 64 bins	30.2	11.6	14.3
	with 128 bins	28.0	10.1	12.6
	with 256 bins	24.6	9.5	29.9
	with 512 bins	-	9.3	17.7
	with 1024 bins	-	10.1	23.2

for each local feature its nearest neighbor is searched from the huge set of database local features and it is counted for each of the database images how many of its local features are found. In the end the image is classified to be from the same class as the image with the highest count. This method is equivalent to the common local feature approach if each image is considered to constitute its own class, since alignments within other images from the same class are not allowed anymore. Even stronger constraints are taken into account for the method labeled "lfd". Here, each local feature is forced to map to a local feature from the other image. That is, the comparison is image-wise. Comparing two images, for each local feature from the query image its nearest neighbor from the set of local features from this particular database image is searched. The strongest constraints are considered for the image distortion model (label "IDM"). Here the subimages from the query image are matched to subimages from a database image from a certain range of positions. That is, the subimages are not allowed to be matched to subimages too far away from the initial position in the images. The results show that dropping constraints about global relationships of images requires more information. That is, for the image distortion model each alignment is somehow related to the neighboring alignments and thus it is not necessary to regard large subimages whereas the local feature approach with an high amount of local information in each subimage works quite well without the global constraints. Intuitively, the local features need a high amount of local information to recover the lack of global information, or the missing knowledge about position in the local feature approach is recovered from a high amount of information of the neighborhoods.

8.2.2 Comparison of Different Features

In the last section we examined different comparison measures for the different types of features, in this section different features are compared using suitable comparison measures. Results for the WANG and the IRMA-1617 database are presented. The results are given as error rates and PR-graphs. The experiments were done in a leaving-one-out manner. That is, each image from the database was selected as query image and the nearest neighbor form

Table 8.6: Error rates [%] using different features for the WANG database.

Feature	Distance measure	ER[%]
inv. feat. histogram $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	JSD	15.9
color histogram	JSD	17.9
Tamura histogram	JSD	31.0
local feature histogram 256	JSD	32.5
image of size 32×32	Euclidean	55.1

Table 8.7: Error rates [%] using different features for the IRMA-1617 database.

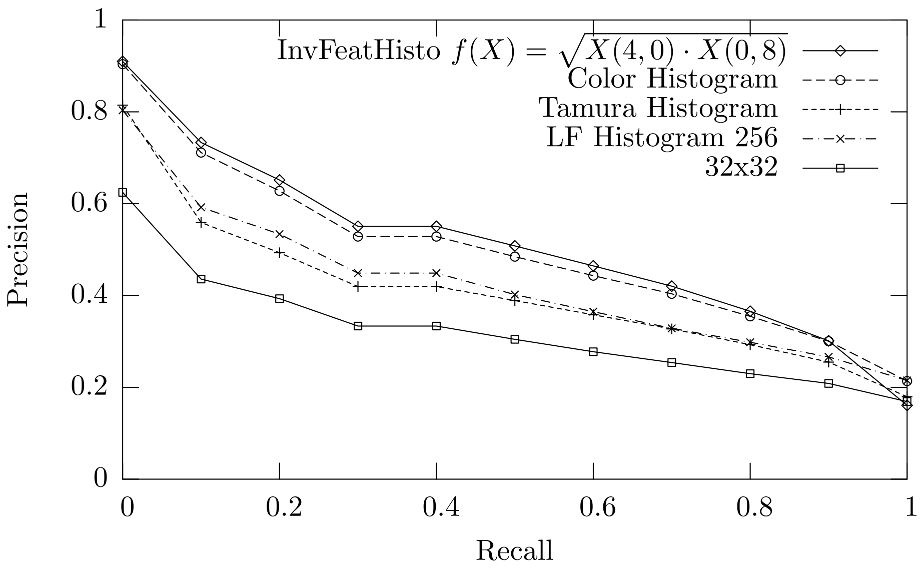
Feature	Distance Measure	ER[%]
image of size $32\times X$	IDM Sobel	6.7
local feature (5×5) histogram 512	JSD	9.3
32×32	Euclidean	17.7
Tamura histogram	JSD	19.3
inv. feat. histogram $f(X) = \mathrm{rel}(X(0,0) - X(0,4))$	JSD	22.6
inv. feat. histogram $f(X) = \sqrt{X(0,0) \cdot X(0,2)}$	JSD	29.2
Fourier Mellin feature	Euclidean	53.1

the remaining images was searched.

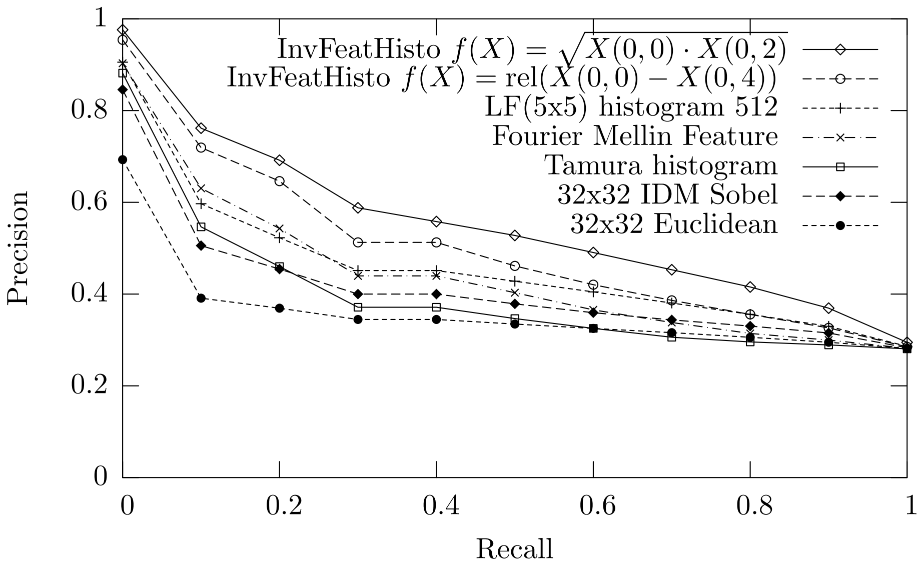
Exemplary results (error rates) we obtained using different features and suitable comparison measures are shown in Table 8.6 for the WANG database and Table 8.7 for the IRMA-1617 database. Complete tables can be found in Appendix B in Tables B.1 and B.2 for reference. The results show that different tasks require completely different features. For the IRMA database the best results are obtained using the image pixels as features as opposed to the WANG database where these features yield poor results. Vice versa, the invariant feature histograms yield very good results on WANG but perform poorly on IRMA. Figure 8.4 shows the corresponding PR graphs.

Since it is often desired to create features invariant with respect to a certain amount of transformation only, we implemented an "invariant" feature histogram considering only a sector of the circle for integration. It has to be noted that according to the theory this approach does not lead to partially rotation invariant features (cp Section 4.4, p. 4.4). Results using different sectors are given in Table 8.8 and approve the theory that it is not feasible to use these features. For these experiments invariant feature histograms with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$ were used and the tests were performed on the WANG database.

As it is well known that combinations of different methods lead to good results [Kittler 98], an objective is to combine the presented features. However, it is not obvious how to combine which features. To analyze the characteristics of features and which features have similar properties, we perform a correlation analysis as described in Section 4.12. The resulting graphs from multidimensional scaling are shown in Figure 8.5 and Figure 8.6 for WANG and IRMA-1617 respectively. The points in these graphs denote the different features. The distances between the points denote the correlations of the features. That is, points very close together denote features that are highly correlated and points farther away denote features with different characteristics.



(a) PR graphs for WANG database using different features.



(b) PR graphs for IRMA-1617 database using different features.

Figure 8.4: PR graphs for WANG and IRMA-1617 corresponding to error rates in Tables 8.6 and 8.7.

The graphs show that there is a clear clustering of features. Both graphs have a large cluster of invariant feature histograms with monomial kernel function. They result from invariant feature histograms using different monomial kernel functions and histogram settings.

Table 8.8: Error rates [%] using invariant feature histograms with partial rotation for the WANG database with $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$.

angle	ER[%]
360°	15.9
270°	16.1
180°	16.2
90°	16.6

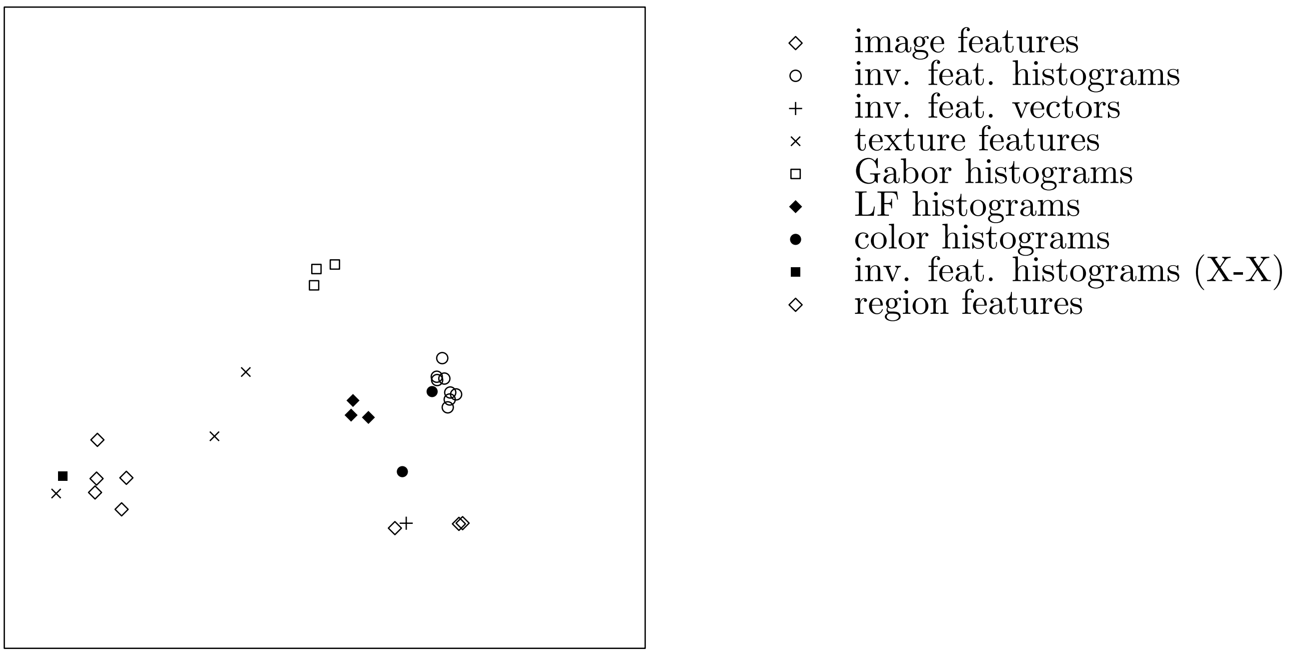


Figure 8.5: Results from Multi-dimensional scaling for WANG features.

Also, the graphs show clusters of local features, local feature histograms, and Gabor feature histograms. The different texture features do not form a cluster. This suggests that they describe different textural properties of the images and that it makes sense to use them in combination.

8.2.3 Image Retrieval Using Different Databases

In this section we present results obtained using weighted combinations of features for different databases. To obtain comparative results we optimize the settings on a given training database and test the settings using a given testing database.

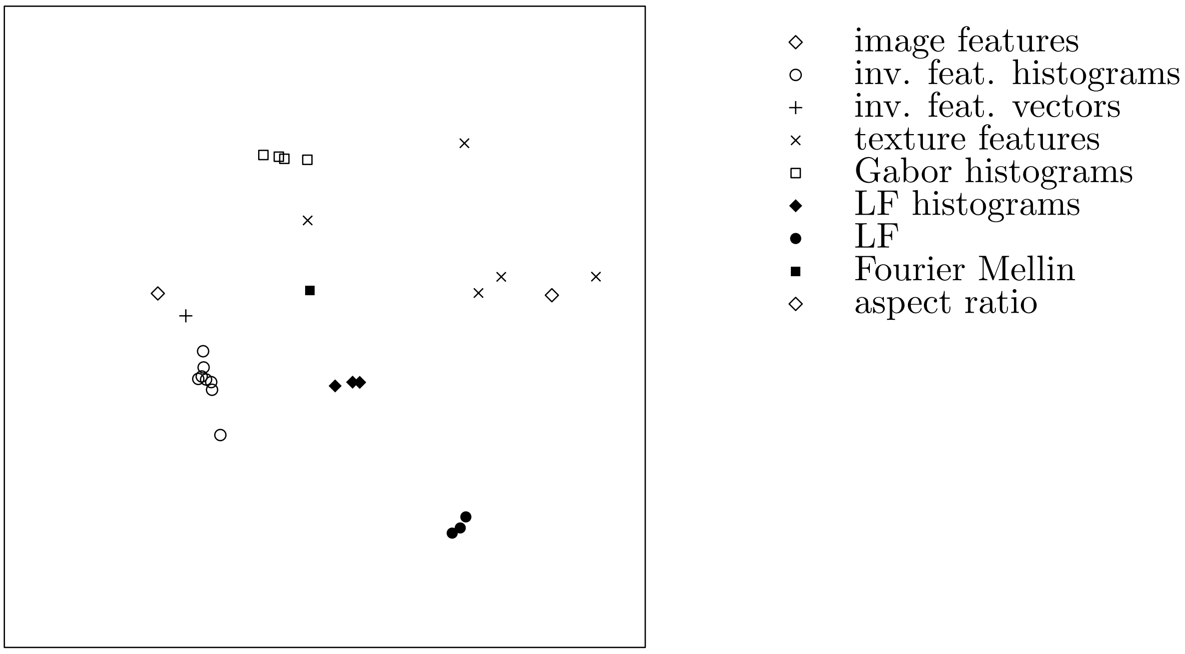


Figure 8.6: Results from Multi-dimensional scaling for IRMA features.

IRMA database

The IRMA-3879 corpus is a good example to show how parameters are optimized and tested: We optimize the settings on the training database using a leaving one out evaluation and use these settings to classify the test database. Table 8.9 presents results for different training and test situation. The first column gives a description of the used training method. Here "IDM Sobel" means that no training was done at all, but the only used features are images scaled to the height 32 and compared using the image distortion model with Sobel values. Also, for "$w_i = 1.0$" no training was done, but all extracted features are used with equal weights. In the remaining lines the settings were optimized on the training data using a greedy algorithm: "$w_i \in \{0, 1\}$" means that the features were weighted with either 0 or 1, this is also known as feature selection. "$w_i \in \{1, \ldots, 10\}$" offers more flexibility the weights can be any integer between 0 and 10. The second column gives error rates for the training data. The parameters were optimized with respect to this error rate. It is interesting that the image distortion model alone is better than a combination of all features. It can be seen that the performance increases as the degree of freedom increases. The third column gives error rates for the testing data. The fact that better error rates on the training data implicate better error rates on the testing data shows that the parameters are not overfitted to the training data. Where ranges are given several parameter settings obtained identical best results on the training data and the ranges denote the varying error rates on the testing data for these different settings. The table also shows that feature selection only is not sufficient to increase the recognition rate but more flexibility is needed here. In both cases where more flexibility was given, the results are improved in comparison to the base line result using the image distortion model alone.

Table 8.9: Results for different training and testing situations on the IRMA-3879 database. (Ranges in the last column denote different parameter settings with optimal result on training data.)

Training	L1O ER[%] on training data	ER[%] test data
IDM Sobel	7.5	7.2
$w_i = 1.0$	9.2	9.1
$w_i \in \{0, 1\}$	7.5	7.3-9.1
$w_i \in \{0, \ldots, 10\}$	6.6	6.3-7.0

Note that this base line result in fact is very good as exactly this method obtains nearly the best results for the smaller IRMA-1617 corpus.

For these experiments the following features were available: 32×32 thumbnails compared with Euclidean distance, $32\times X$ thumbnails compared with image distortion model distance, two different Tamura texture histograms compared with Jensen Shannon divergence, Tamura texture images compared with Euclidean distance, three different invariant feature histograms with monomial kernels compared with Jensen Shannon divergence, one invariant feature histogram with relational kernel compared with Jensen Shannon divergence, invariant feature vectors compared with Euclidean distance, global texture descriptors compared with Euclidean distance, and image sizes compared with Euclidean distance. In Table B.3 the error rates for all of these features are given for the 8 classes and for the 26 classes task.

To be able to compare these results to the results obtained earlier on the smaller IRMA-1617 database, Table 8.10 gives error-rates for this database together with results published in other works results obtained in this work are given in the lines labeled with "Feature selection". For these experiments a set of 36 different features for each image was used. The features used were selected in a greedy manner. That is, first each feature was tested apart, then the best of these features was taken, and all other features were tested together with this feature, and so forth.

Using this method to optimize the parameters for the leaving-one-out classification, leads to an error rate of 4.8% on the IRMA-1617 corpus which is the best obtained error rate so far. To check whether these parameter set is overfitted to the data, we also applied this method in an eight-fold cross validation experiment. That is, we trained the used feature set on seven eighth of the data and classified the remaining eighth using these parameters. The error rate of 6.1% shows that the parameters were overfitted to the data. Nonetheless, the result of 6.1% is an improvement in comparison to the error rate obtained with image distortion model alone. Bootstrap method [Efron & Tibshirani 93] showed that this is an improvement on the 11% level.

For these experiments the following features were available: four different Gabor histograms, twelve different local feature histograms, ten different invariant feature histograms with monomial kernel, one invariant feature vector, one invariant feature histogram with relational kernel, two different thumbnails of the image, Gabor features, two different types of local features, two different Tamura texture feature histogram, a Tamura texture image, a global texture descriptor, the image size, and a Fourier Mellin feature.

Further analysis of the results for the IRMA-1617 corpus has been done. Figure 8.7 shows the PR-graphs for the 6 IRMA-1617 classes apart. The graphs show that the retrieval

Table 8.10: Error rates [%] obtained on the IRMA-1617 database.

Method	ER[%]	
Pseudo 2D hidden Markov model	[Gollan 03]	5.3
Image distortion model	[Gollan 03]	6.7
Local features & tangent distance	[Kölsch 03]	7.4
Extended tangent distance	[Keysers 00]	8.0
Feature selection (cross validation)		**6.1**

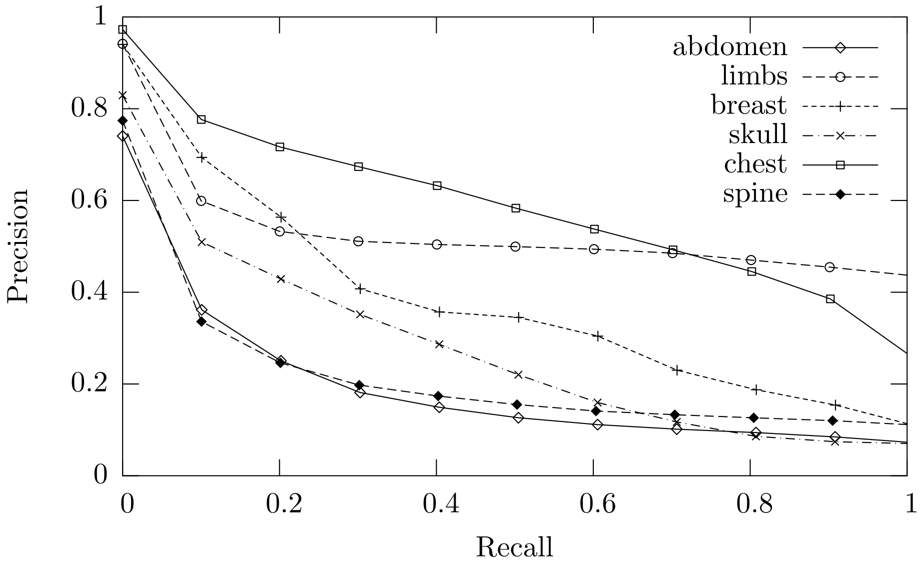


Figure 8.7: Class wise PR-graphs for the six IRMA classes.

performance for an image is class dependent. Different factor have an impact on this. First, different types of images may be differently hard to distinguish, e.g. the difference between a skull and a hand image image is quite clear whereas the difference between spine and an abdomen images is not so obvious. Another very important factor is the size of the classes. Given a query image from a larger class, the probability to find images from the same class is higher than for a smaller class. In this database the class "chest" contains 43% of the images, and the class "abdomen" contains only 7% of the images.

WANG database and UW database

Similar experiments to those performed on the IRMA databases were carried out using the WANG database and the UW database (cp. Sections 7.2 and 7.6). That is, we optimized the parameters for the one database and tested this setting on the second database. Results for these experiments are given in Table 8.11.

The lines in this table denote the parameter trainings performed. The first line "$w_i = 1.0$" means that all features available were used with equal weights. The second line means that the features used were selected from the MDS graph in Figure 8.5 such that the features are

Table 8.11: Results for different training situations for the WANG and UW databases.

Training on	ER[%] WANG	ER[%] UW
$w_i = 1.0$	12.7	12.2
MDS graph	16.0	13.9
WANG	(9.9)	13.5
WANG (sel)	(10.9)	13.8
UW database	15.1	(9.4)
UW database (sel)	16.2	(10.2)

as different as possible. In this case we selected 32×32 thumbnails, local feature histograms, invariant feature histogram with monomial and relational kernel, global texture descriptors, region features, and Tamura texture histograms. The lines "WANG" and "UW database" give the error rates for experiments where the parameters were optimized on the databases and the lines "WANG (sel)" and "UW database (sel)" give error rates for experiments where the parameters were optimized on the databases but the weights were restricted to 0 and 1.

The results show that in both cases (transfer from WANG to UW database and vice versa) the parameters are overfitted to the training data for all training cases. When optimizing the setting on the WANG database, either with selection or unrestricted, the error rate improves strongly compared to the error rate when just using all features with equal weights, but the error rate on the UW database deteriorates strongly. The same applies to the case when optimizing the parameters on the UW database and testing on the WANG database. This shows that this training overfits the parameters to the training data. This overfitting is so clearly due to the fact that these two databases are not very similar and thus need different settings.

For these experiments the following features were available for both databases: five invariant feature histograms with monomial kernel with different settings, one invariant feature histogram with relational kernel, an invariant feature vector, a 32×32 thumbnail, local features of size 19×19 from the 32×32 thumbnails, local feature histograms with 128 and 256 bins, the global texture descriptor, a Tamura texture histogram and a region description of the images.

WANG database and COREL subset

Since the databases WANG and UW database differ quite strong in their characteristics we also tried to transfer trained parameters from WANG to the COREL subset (cp. Section 7.3). Results for these experiments are given in Table 8.12. The line "$w_i = 1.0$" gives results where all available features were used with equal weights. The lines "WANG" and "Corel subset" give results for parameters optimized on these databases respectively. From these results it can be seen that it is possible to transfer the features and weightings for image retrieval from one database to another if the databases are adequately similar because for both training cases the results not only improve on the training database but also on the testing database

Table 8.12: Error rates [%] for different training situations for the WANG and Corel subset databases.

Training on	ER[%] WANG	ER[%] Corel subset
$w_i = 1.0$	13.3	21.7
WANG	(12.0)	21.6
Corel subset	13.2	(20.5)

Table 8.13: Error rates [%] for different training situations for the WANG subsets.

Training on	WANG even	WANG odd	query odd on even	query even on odd
$w_i = 1.0$	14.4	17.4	16.4	13.6
WANG even	(10.6)	16.4	14.8	
WANG odd	13.2	(13.2)		12.8

in comparison to the case where all features were used with equal weights.

For these experiments the following features were available: two different invariant feature histograms with monomial kernel, an invariant feature histogram with relational kernel, an invariant feature vector, and a Tamura feature histogram.

To emphasize the fact that the settings for image retrieval are transferable from one task to another if the databases are sufficiently similar, an experiment with even more similar databases was done. We subdivided the WANG database into 2 parts of 500 images each, one part consisting of all images with even numbers, the other consisting of images with odd numbers. Thus, each of these parts consisted of 10 classes with 50 images in each class. Results for different training situations with the WANG subsets are given in Table 8.13. The top line gives results for the case that all features are used with equal weights, the other two lines give the results for training of the weights on the two parts of the database. The first two columns show the error rates if leaving one out is used on the databases, the other two columns give the error rate if one database is used to query the other. That is, classifying one database using the other as training data. In all cases training of the parameters on one database improves the results for the other database and for both query experiments in comparison to the case where all features are used with equal weights. Thus it is suitable to optimize parameters for the database if the expected query images are similar to those contained in the database.

For these experiments the following features were available: five different invariant feature histograms with monomial kernel, an invariant feature histogram with relational kernel, an invariant feature vector, local feature histograms with 128 and 256 bins, 32×32 thumbnails of the images, a global texture descriptor, a Tamura texture feature histogram, and a region descriptor of the images.

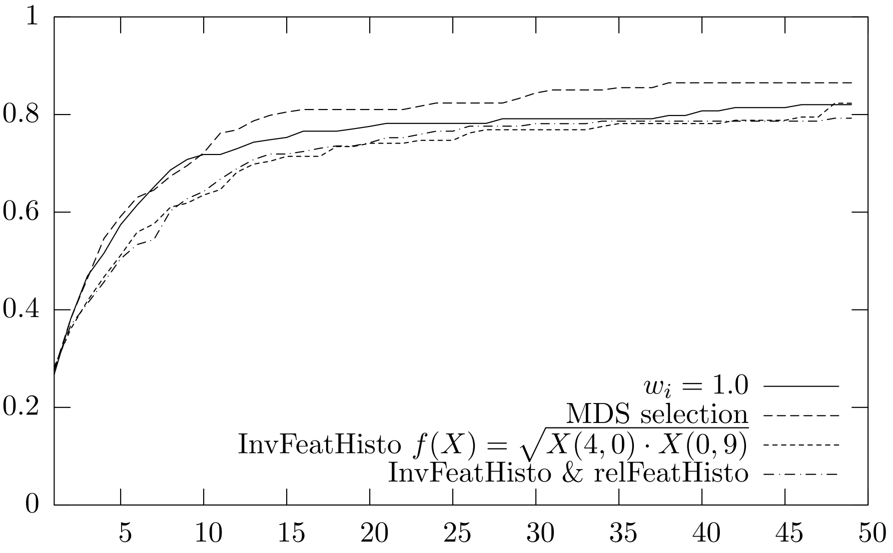


Figure 8.8: Graph of Recall vs N_R for the MPEG-7 database.

Table 8.14: Average normalized ranks $\widetilde{Rank}$ for the MPEG-7 database for different settings.

Method	$\widetilde{Rank}$
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	0.040
InvFeatHisto & relFeatHisto	0.034
$w_i = 1.0$	0.032
MDS selection	0.032

MPEG-7 database

For the MPEG-7 database a set of 15 images was selected in [Siggelkow 02] and for each of these images the set of relevant images was manually determined. For these experiments it does not make sense to give precision values as the low number of images from each group (between 3 and 14) lets the values decrease very fast. Also, to give error rates is not suitable for a test set of 15 images only. Additionally, the error rate is 0% due to the fact that the query images are contained in the database.

For comparison of different methods we give a graph of Recall plotted vs. the number of images returned in Figure 8.8 and average normalized ranks $\widetilde{Rank}$ in Table 8.14. The results show that the base method, using an invariant feature histogram with monomial kernel alone ("InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$"), can be improved by adding other features. To use the invariant feature histogram with relational kernel additionally ("InvFeatHisto & relFeatHisto") improves the results strongly. To use all available features ("$w_i = 1.0$") improves the result further and to use a reasonable feature set ("MDS selection") yields the best result. The experiments using only the invariant feature histograms were presented in [Siggelkow 02] and were repeated for this work to be able to compare them.

Table 8.15: Error rates [%] obtained on the ZuBuD database using different methods.

Method	L1O ER[%] on training data	ER[%]
[Shao & Svoboda+ 03a]		13.9
[Obdrzalek & Matas 03]		0.0
$w_i = 1.0$	7.3	15.7
Weights optimized on training data	(3.9)	10.4

For all these experiments, available features were: two invariant feature histograms with monomial kernels different settings, an invariant feature histogram with relational kernel, an invariant feature vector, the global texture descriptor, 32×32 thumbnails, and the Tamura feature histogram. For the MDS selection one invariant feature histogram with monomial kernel, the invariant feature histogram with relational kernel, the Tamura feature histogram, and the global texture descriptor were used. The thumbnails were not used as they did not obtain good results for the WANG database and the second invariant feature histogram was not used as one invariant feature histogram should be enough. Probably it would be possible to obtain even better results when training the weights on this data but this would result in a parameter set valid for exactly this task only.

ZuBuD database

As the ZuBuD database is subdivided into a training and a testing database, it is easily possible to train and test parameters. Comparison results are presented in [Shao & Svoboda+ 03a] using local invariant feature descriptors and in [Obdrzalek & Matas 03] using local affine frames. The comparison results are given together with our results in Table 8.15. The line "$w_i = 1.0$" gives error rates for the case that all features extracted are used with equal weights and the line "Weights optimized on training data" shows results where the feature weightings were optimized for the training data using a leaving-one-out approach. The results are better than those presented in [Shao & Svoboda+ 03a] but worse than those presented in [Obdrzalek & Matas 03]. In [Obdrzalek & Matas 03] the settings are optimized on this database as they tested various parameters and only one parameter setting obtains this results.

CalTech database

Another task which is closely related to content based image retrieval is the classification of complex scenes. One such task is the CalTech database (cp. Section 7.5). Here, three different tasks are considered, each consisting of a two class decision problem whether the object of interest is depicted in the image or not. In Table 8.16 we present results we obtained using the features presented in this work without incorporation of special domain knowledge or complex models like those proposed in [Fergus & Perona+ 03] and [Weber 00]. The results show that each of the three tasks can be solved better with a combination of simple features than with a complex model. This implies that the task can be regarded as "too easy" for complex recognition tasks as the global image similarity is already sufficient to obtain very good results.

Table 8.16: Equal error rates [%] on the CalTech database.

Method	airplanes	faces	motorbikes
32×32	24.0	15.0	17.4
[Weber 00]	32.0	6.0	16.0
[Fergus & Perona$^+$ 03]	9.8	3.6	7.5
Tamura feature	1.6	3.9	**7.4**
combination of features	**0.8**	**1.6**	8.5

The line "32×32" gives an absolute baseline error rate. This error rate was obtained using a nearest neighbor classifier for images scaled to 32×32 pixels. The line "Tamura feature" gives the error rate for a nearest neighbor classifier using only the Tamura texture histogram and the line "combination of features" uses a combination of Tamura features and invariant feature histograms.

In [Fei-Fei & Fergus$^+$ 03] experiments on the same data are made, yielding error rates slightly worse than those presented in [Fergus & Perona$^+$ 03] but the amount of used training data was reduced enormously. Here, the authors use one to five training images only.

8.3 Performance Evaluation for Clustering Visually Similar Images

Evaluation of clustering algorithms is a task which was addressed before. A common way to measure the quality of a partitioning of a data set is the Rand index [Jain & Dubes 88, Saporta & Youness 02]. It is based upon the number of pairs from the same class belonging to the same cluster. The Rand index is a number between 0 and 1 where 0 means that the clustering is bad, and 1 means that the clustering is identical to the original partition.

To calculate the Rand index, it is necessary to calculate the contingency table. That is, given two partitionings $P = \{p_1, \ldots, p_N\}$ and $P' = \{p'_1, \ldots, p'_M\}$ the $M \times N$ matrix C with $C_{nm} = |p_n \cap p'_m|$ is calculated. Given this matrix the Rand index R is computed as

$$R = \frac{\sum\limits_n \sum\limits_m \binom{C_{nm}}{2} - \left[1/\binom{C}{2}\right] \sum\limits_n \binom{C_{n\bullet}}{2} \sum\limits_m \binom{C_{\bullet m}}{2}}{\frac{1}{2}\left[\sum\limits_m \binom{C_{\bullet m}}{2} + \sum\limits_n \binom{C_{n\bullet}}{2}\right] - \left[1/\binom{C}{2}\right] \sum\limits_n \binom{C_{n\bullet}}{2} \sum\limits_m \binom{C_{\bullet m}}{2}}$$

with $C_{n\bullet} = \sum\limits_m C_{nm}$, $C_{\bullet m} = \sum\limits_n C_{nm}$, and C is the number of images clustered.

8.4 Results for Clustering Visually Similar Images

In this section we present the results we obtained with the clustering application. For the Google database we present some exemplary results. For this database it is not possible to give quantitative results as we do not have information which images have to be in a cluster together. To give quantitative and comparable results we also applied the algorithms to the WANG and the COIL database.

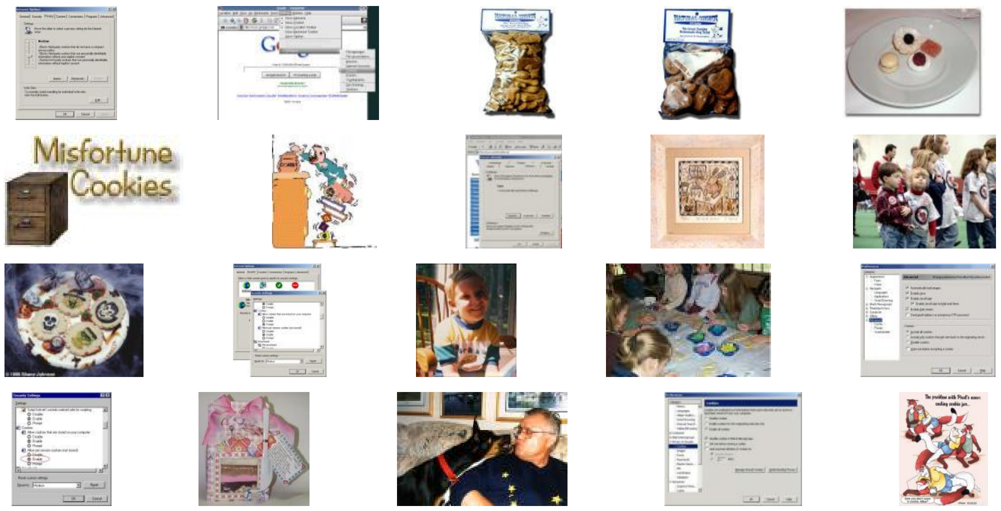


Figure 8.9: Results from Google image search for query “cookies”, unprocessed.

8.4.1 Clustering the Google Images

Figure 8.10 shows the result of clustering the 20 first results from Google image search queried with “cookie”. Cluster 1 contains only images with people, cluster 2 contains images of edible cookies and some drawings, cluster 3 contains images not fitting into any other cluster, and cluster 4 contains screenshots. Though the partitioning is not a perfect one, as cluster 3 contains images which could be from the other clusters, it is obviously an improvement over an unsorted display as depicted in Figure 8.9.

Figure 8.11 shows the result of clustering the first 25 results from Google image search queried with “aircraft”. Also here, visually similar images are together in one cluster.

8.4.2 Clustering the WANG & COIL Images

To be able to quantify the results from clustering images we chose the WANG database, where we know the reference partitioning into 10 clusters and the COIL-100 database which is divided into 100 classes. To compare our results to the results from other groups we also take a 20 class subset of COIL-100 database (cp. Section 7.10).

In both cases the results improve strongly by adding features. Another interesting results is that the results from the LBG clustering algorithm are better than those obtained using k-means clustering, though LBG clustering is provided with less information about the number of clusters. This is very clear for the complete COIL-100 database. We assume that the k-means algorithm is sensitive to the initial partition. Thus, the LBG clustering algorithm seems more robust when using a high number of clusters. Another remarkable result is the fact that our methods outperform the results presented in [Käster & Wendt$^+$ 03] though they tested a lot of parameter settings for each of their algorithms and the results cited are the best results obtained in their work. That is, the parameters used are trained on the testing data for their work. Whereas our methods were not specially optimized to the task. The only setting changed was the number of clusters for k-means, the number of splits for LBG clustering and the features used as shown in the tables. For the k-means algorithm the correct number

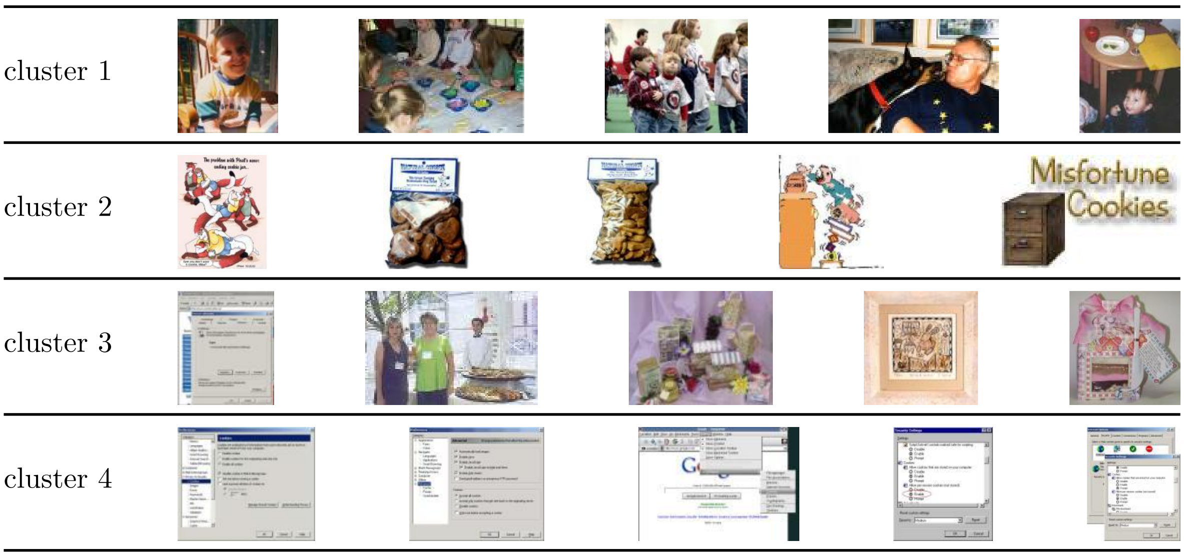


Figure 8.10: Results from Google image search for the query “cookies” clustered using the LBG clustering algorithm.

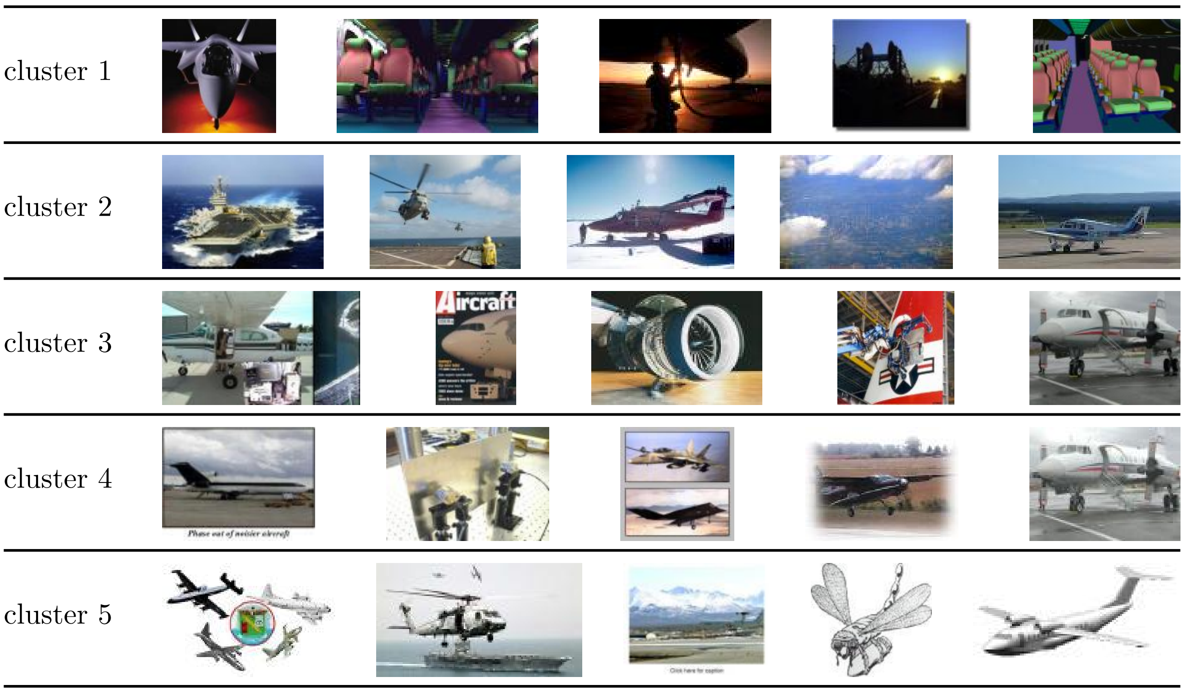


Figure 8.11: Results from Google image search for the query “aircraft” clustered using the k-means clustering algorithm.

of cluster centers was given, and for the LBG clustering the number of splits was set to the minimum allowing for the right number of clusters. These results leads us to the conclusion

Table 8.17: Results for clustering WANG.

Database	Method/Feature	Rand-Index LBG	Rand-Index k-means
WANG			
	(1) Invariant feature histogram	0.46	0.47
	(2) Relational invariant feature histogram	0.22	0.24
	(3) Tamura histogram	0.25	0.27
	(1) & (2)	0.43	0.53
	(1) & (3)	0.48	0.54
	(1) & (2) & (3)	**0.57**	**0.58**

Table 8.18: Results for clustering COIL.

Database	Method/Feature	Rand-Index	
COIL 20			
	Results from other works		
	[Käster & Wendt+ 03] k-Means		0.53
	[Käster & Wendt+ 03] CLARA		0.54
	[Käster & Wendt+ 03] PAM		0.55
	[Käster & Wendt+ 03] hierarchical		**0.62**
		LBG	k-means
	(1) Invariant feature histogram	0.64	0.67
	(2) Relational invariant feature histogram	0.53	0.63
	(3) Tamura histogram	0.45	0.63
	(1) & (2)	0.83	0.74
	(1) & (3)	**0.83**	0.76
	(1) & (2) & (3)	0.79	**0.82**
COIL 100			
	(1) Invariant feature histogram	0.53	0.67
	(2) Relational invariant feature histogram	0.48	0.63
	(3) Tamura histogram	0.51	0.63
	(1) & (2)	0.54	0.74
	(1) & (3)	**0.65**	0.76
	(1) & (2) & (3)	0.64	**0.82**

that the features presented are well suited to this task.

Chapter 9

Conclusion and Perspectives

Conclusion

In this work a broad variety of features for content-based image retrieval was presented, investigated, and experimentally evaluated. This work gives a review of features proposed for image retrieval and refines several of them. An emphasis was placed on the invariant feature histograms. All the features presented were used for content-based image retrieval on a wide variety of databases to evaluate the different discrimination performances of these features. For the first time, quantitative results are given for a wide variety of databases using different image retrieval methods. Experiments to determine the optimal set of features for different image retrieval tasks were carried out and the characteristics of the different features were analyzed using an empirical correlation analysis.

For the invariant features we investigated the extension to scaling and to partially rotation invariant features. Both extensions did not improve the retrieval performance, which is in accordance with the theoretical prediction.

To analyze the features various dissimilarity measures were implemented, refined, and tested, as different features require different comparison methods. As well as a large amount of features was introduced and presented, we gave a broad overview of different comparison measures for the different types of features.

An important part of this work was the implementation of a flexible image retrieval system, capable of managing large amounts of different features and distance measures. The system is a fully automatic system, that is, it is presented with a query image and without any further information the query is evaluated.

To measure the performance of the system an appropriate performance measure had to be determined. Several different performance measures were evaluated and compared using an empirical correlation analysis. This analysis showed that most of the performance measures are indeed highly correlated and thus the error rate was selected as performance measure to compare the different image retrieval setups since it is a well known method in object recognition and it is easily computable.

The experiments showed that the connection between image retrieval and image recognition is in fact a very close one. All the methods that improve the results in the one task also improve the results in the other task. On the one hand, the high flexibility of the image retrieval system implemented allowed for using many of the classification methods known for image retrieval. On the other hand, the features proposed for image retrieval were used for classification.

The experiments show that the selection of features for an image retrieval task strongly depends on the images involved. One important aspect is to use a selection of features accounting for the different properties of the images as there is no feature capable of covering all aspects of an image. The experiments show that for color images of a general type (e.g. photographs of arbitrary scenes) a combination of the following features is suitable:

- invariant feature histograms
- Tamura texture histogram
- local features or local feature histograms
- Gabor features

For gray images from a more limited domain (e.g. radiographs) the invariant features do not obtain good results but instead the pixel values of the images are very important.

Experiments to find a good feature set revealed that it is suitable to search for an optimal set of weightings and features if the training database is very similar to the testing database used for the actual retrieval task. However, this is not suitable to search for an optimal setting when the databases differ significantly. In this case it is best to calculate a reasonable set of features and use all of these with equal weight.

Application of the presented features is not limited to image retrieval tasks, but the insights are adaptable to other tasks as well, for example for classification. Another application where the features were used is clustering of visually similar images. Since many image search engines are based on text retrieval, these image search engines are not suitable for all tasks. With the methods presented, it is possible to enhance the results from text based image search engines. We presented an approach to improve text based image search engines using the proposed features and some clustering algorithms. That is, given a set of images, the features are extracted and a clustering algorithm is applied to present the user with a set of images that can be browsed more easily.

We investigated the connection between local features and the image distortion model and found that local features recover the lacking restriction with respect to deformation constraints by a higher amount of local information. This was proven experimentally for the local feature approach with very small local features and for the image distortion model with large subimages.

Perspective

At this point of the work, still some questions remain unanswered and many ideas remain untested and need to be investigated. It would be interesting to see the results of other image retrieval systems using the same databases. One step in this direction is already in progress as the authors of the GNU Image Finding Tools (GIFT) are experimenting with the IRMA database in the context of the MedGIFT project.

Another desired application is to use the methods presented to accelerate and ease the task of classification of images to be included into the IRMA database. Here, the images to be classified by radiologists are taken as query images to the already classified images and the nearest neighbors for those are returned together with their classification to allow the radiologist selecting the best fit and refine the classification if necessary.

As the image retrieval system as it is realized now has to keep all data necessary for the retrieval process in memory the size of the database to search is restricted by the amount of memory available. For the future, database support is desirable.

A future aim for content-based image retrieval surely is to go from appearance based image retrieval to semantics based image retrieval. That is, the images should be retrieved based on the objects contained in the scenes, the acts depicted or, if necessary, e.g. the names of the persons shown. For semantic retrieval a complete understanding of the images is necessary.

Similar demands for image understanding techniques are in the task of automatic annotation. Here the objective is to take a picture and automatically generate a textual description of its contents. To get a satisfactory result, image understanding is necessary as well.

From the fact that most of the works cited are from the last five years and image retrieval as it is understood now, is a quite new area of research it can be seen that content-based image retrieval is an active area of research. The still increasingly growing amount of digitally available pictures will enforce further research.

Another very important task is the creation of a well-documented standard test database for content-based image retrieval and fixed testing criteria to be able to compare different image retrieval systems in a quantitative way as it is already common in textual information retrieval for years. A first starting point for this is given in this work, but further research, larger databases, and better annotations and relevance criteria still have to be found.

For the analysis of complex scenes some of these features offer possibilities to be integrated into a holistic model as proposed in [Keysers & Dahmen^{+} 03].

Appendix A

Software Documentation

In this chapter we give an overview of the software developed in the context of this work. First a list of software which has been used to create the software, the data, and this document is given and then we introduce the software developed.

In the course of this work many freely available programs have been used: Linux as operating system, X and sawfish window manager as graphical desktop environment, gcc compilers and python to write programs, libpng and libjpeg for reading image files, libANN for efficient approximative nearest neighbor search, GetPot for parsing of command lines, qiv, xv, ImageMagick and The Gimp for viewing and manipulating images, Emacs, XEmacs and Vim for writing programs, scripts, and this thesis, and LaTeX and xfig for typesetting this thesis. In the following a list of the software developed in the context of this work is given together with a short description for each program.

A.1 FIRE Framework

FIRE (Flexible Image Retrieval Engine) is a framework for content based image retrieval. It consists of a server part implemented in C++ and a client part implemented in python. The server and the client communicate over network sockets by a simple line-oriented protocol.

fire

The server of fire can work in different modes. The most important mode is server mode. In server mode, a client is able to connect via network socket to the server and change its settings and retrieve images from a database. Other modes available are distance file mode, performance evaluation mode, and query performance evaluation mode.

In distance file mode a file with all distances between all image pairs from the database is created. This is necessary for efficient searching of the best parameter set. In performance evaluation mode, the system queries itself in a leaving one out manner for each image from the database and returns the results, calculates performance evaluation measures and averages those over all images. In query performance evaluation mode the retrieval performance for a given setting is measured with query images that are not contained in the database.

Invocation of fire is done as follows:
`fire (-s | -perf | -distanceFile <filename> | -qperf)` [options]
with

`-s [port]` starts fire in server mode. By default the socket `12960` is used, but a different port can be specified.

`-distanceFile <filename>` starts fire in distance file mode. The distance file is written to the specified file.

`-perf` starts fire in performance evaluation mode.

`-qperf` starts fire in query performance evaluation mode.

For each of these modes the following options are available. In some cases some options are obligatory:

`-filelist <filename>` specifies the file list describing the database to be loaded by fire (obligatory in distance file mode, performance evaluation mode, and query performance evaluation mode).

`-results <number>` specifies the number of results returned for each query (default: 9).

`-distN <distname>` sets the distance measure used for the N-th file (N starting with 0, obligatory in distance file mode, performance evaluation mode, and query performance evaluation mode for at least one N, otherwise no useful result is obtained). For a list of available distance measures see Table A.1.

`-weightN` set the distance measure used for the N-th file (N starting with 0, defaults to 1 if N-th distance is set, 0 otherwise).

`-queryFileList <filename>` to specify the file list describing which files to use in query performance evaluation mode (obligatory in query performance evaluation mode, ignored otherwise).

`-ROCoutfile <filename>` if specified, in query performance evaluation mode, a file with distances to nearest neighbors from each class is written for each query. This is necessary to obtain ROC (Receiver Operating Characteristic) curves. (ignored in any mode apart from query performance evaluation mode).

fire.py and fireadm.py

`fire.py` and `fireadm.py` are web front ends to the fire server. Both are implemented in python as cgi-scripts and communicate with the server using a TCP socket. The difference between `fire.py` and `fireadm.py` is that the latter allows for changing several parameters as the used distances, the used weights, the used database, and the number of results. Default server, default port, and directory for temporary data can be modified easily by changing some variables in the program.

A.2 Clustering Framework

The clustering framework is a set of two programs to cluster images into visually similar groups. There is a main program written in C++ and a small web front end written in python executing the main part.

clustertest

`clustertest` is the main clustering program. It reads images from a database and clusters them. `clustertest` takes the following parameters:

`-c <clusteralgorithm>` to select the cluster algorithm used. `em` and `kmeans` are available at the moment.

`-d <distancemeasure>` to select the distance measure used to compare images and to compare images with cluster centers. For a list of available distance measures see Table A.1.

`-h` to give a short help

`-dummy` if this parameter is given, no input data is read, but some data is generated and clustered. This is convenient to test cluster algorithms

`-jf <filename>` if this parameter is given, the specified text file containing the data is read. The file has to be an ASCII file of very simple format used at the Lehrstuhl für Informatik 6 for internal data.

`-rgb <filename>` if this parameter is given, the specified filelist is read. If additionally the `-suffix` option is given, only a subset of the available feature files is read.

`-suffix=<suffix1:suffix2:...:suffixN>` using this parameter it is possible to specify the features used from a database description file.

`-noAnalyse` by default, a set of cluster performance evaluation measures is computed. If this parameter is used, this analysis is omitted.

`-testAnalyser` if this parameter is specified, after clustering and giving the performance evaluation measures some test cases for the performance measures are generated and analyzed.

`-noRearrange` by default, after clustering the data is rearranged such that the data points are sorted by distance to the cluster center. If this parameter is specified this step is omitted.

Parameters only applicable if `-c em` is given

`-splitMode <split mode>` selects the split mode. Available options: `allSplit`(default), `largestSplit`, `varianceSplit`.

`-disturbMode <disturb mode>` selects the center disturb mode. Available options: `varianceDisturb` (default), `meanDisturb`, `meanDisturb2`, and `constDisturb`.

`-poolMode` selects the averaging mode for the cluster variances.
Available options: `clusterPooling`, `dimensionPooling`, and `noPooling` (default).

`-dontSplitBelow <number>` specifies the smallest cluster size which may be split. If a cluster has less members than this, it is not split (default: 10).

`-iter <number>` specifies the number of reestimation iterations between two splits (default: 10).

`-minObs <number>` specifies the number of data points which have to be in a cluster. If the cluster has less members it is deleted (default: 4).

`-maxSplit <number>` specifies the maximum number of splits (default: 4).

`-stopWithNClusters <number>` if this option is set, the algorithm stops splitting when the specified number of clusters is reached.

`-epsilon <value>` specifies the epsilon used for disturbing the cluster centers (default: 0.1).

Parameters only applicable if `-c kmeans` is given

`-nOfClusters <number>` specifies the number of clusters to be calculated (default: 10).

`-iterations <number>` specifies the number of reestimation iterations (default: 10).

cloogluster.py

`cloogluster.py` is a very simple web front end for `clustertest`. It reads a list of available databases and provides a graphical interface to `clustertest`. clustertest is started and the output is parsed to present the clusters to the user in a convenient way.

Nearly all options of `clustertest` can be accessed easily from this interface. Paths and environment setup can easily be adjusted.

A.3 Feature Extraction Tools

In this section the programs developed for feature extraction are briefly described. All the programs are able to read jpeg and png image files when images are loaded and read gzipped text files when text files are read.

extractColorHisto

`extractColorHisto` is a program to create simple color or grey histograms from images. `extractColorHisto` is invoked by:

```
extractColorHisto [-steps <number>]
        (-pseudocolorhisto|-greyhisto|-mdcolorhisto) filename
```

The options mean:

`-steps <number>` specifies the number of bins for the resulting histogram.

`-mdcolorhisto` specifies that the given image is read as color image and a multi dimensional histogram is created. The value of `-steps` specifies the number of steps per dimension here, i.e. `-steps 8` results in a 512 bin histogram from an RGB image.

`-pseudocolorhisto` specifies that the image is read as color image, and one histogram is created for each color layer. `-steps` specifies the number of bins per color layer here.

`-greyhisto` specifies that the image is read as gray value image. A gray value histogram with the number of bins specified by `-steps` is extracted.

The output is written to a file with the same name as the input file concatenated with `.pseudomdhisto`, `.greyhisto`, or `.mdhisto`, respectively.

Table A.1: Available dissimilarity measures.

Symbol	Description	Identifier
$d_2(\cdot,\cdot)$	Euclidean distance	`euclidean`
$d_1(\cdot,\cdot)$	L_1 distance	`l1`
$d_{JSD}(\cdot,\cdot)$	Jensen Shannon divergence	`jsd`
$d_{KLD}(\cdot,\cdot)$	Kullback-Leibler divergence	`kld`
$d_{\chi^2}(\cdot,\cdot)$	χ^2-distance	`chisquare`
$d_{his}(\cdot,\cdot)$	histogram intersection	`histogramintersection`
$d_{EMD}(\cdot,\cdot)$	earth movers distance	`emd`
$d_{tw}(\cdot,\cdot)$	time warp distance with Euclidean	`timeshift`
$d_{tw}(\cdot,\cdot)$	time warp distance with L_1	`timeshiftl1`
$d_{lf}(\cdot,\cdot)$	local feature distance	`localfeatures`
$d_t(\cdot,\cdot)$	tangent distance	`tangent`
$d_{rd}(\cdot,\cdot)$	relative deviation	`reldev`
$d_{rbd}(\cdot,\cdot)$	relative bin deviation	`relbindev`
$d_t(\cdot,\cdot)$	tangent distance for histograms	`histotangentdist`
$d_{qhrm}(\cdot,\cdot)$	quantized hungarian region matching	`regiondistv2`
$d_{irm}(\cdot,\cdot)$	greedy region matching	`irm`
	integrated region matching	
$d_f(\cdot,\cdot)$	fidelity	`fidelity`
$d_{\overline{F}}(\cdot,\cdot)$	fidelity based distance measure	`oneminusfidelity`
$d_{\sqrt{1-F}}(\cdot,\cdot)$	fidelity based distance measure	`sqrtoneminusfidelity`
$d_{\log(2-F)}(\cdot,\cdot)$	fidelity based distance measure	`log2minusfidelity`
$d_{\arccos F}(\cdot,\cdot)$	fidelity based distance measure	`arccosfidelity`
$d_{\sin F}(\cdot,\cdot)$	fidelity based distance measure	`sinoneminusfidelity`
$d_{idm}(\cdot,\cdot)$	image distortion model	`idmdistance`
$d_{idm}(\cdot,\cdot)$	image distortion model (with Sobel)	`idmsobel`
	vote counting for local features	`globallocalfeaturedistance`.

extractTamuraTextureFeature

`extractTamuraTextureFeature` extracts Tamura texture features as described in Section 4.6. It is invoked by

```
extractTamuraTextureFeature [options](-color|-grey) <filename>
```

Available options are:

`-suffix <suffix>` to specify a suffix for the output file (default: `.tamurafeature`)

`-saveTextureImages` to specify that the texture image is saved. By default it is not saved.

`-textureimagesuffix` to specify the suffix for the texture image file. (default: `.tamura.png`).

The `-color` and `-grey` switches select whether the input image is read as color or grey image. The output histogram is written to a file with the same name as the input file concatenated with the suffix.

extractaspectratio

`extractaspectratio` is a program to write the size of the image to a file. Invocation is done with

```
extractaspectratio [options] -file <filename>
```

Available options are:

`-suffix <suffix>` to specify a suffix for the output file (default: `.ar`)

The file given with `-file` is read as image and the size is written to a file with the same name concatenated with the given suffix.

extractglobaltexturefeature

`extractglobaltexturefeature` is a program to extract the texture feature described in Section 4.7. Invocation is done by

```
extractglobaltexturefeature [options] (-color|-grey) <filename>
```

Available options are:

`-suffix <suffix>` to specify a suffix for the output file (default: `.globtexturefeat`)

The file given by `<filename>` is read as color or gray image depending on the option and the output is written to a file with the same name concatenated with the suffix. The main part of this program is courtesy of the IRMA project and was developed by Boris Terhorst [Terhorst 03].

extractinvariantfourierfeature

`extractinvariantfourierfeature` is a program to extract a Fourier Mellin feature as described in Section 4.4.3. The experiments have not been done with this program, but with an implementation in MatLab offering higher flexibility. Invocation of this program is done by

```
extractinvariantfourierfeature -file <filename>
```

The given file is read as gray image, the Fourier Mellin transformation is applied and the result is written to a file with the same name concatenated with `.fouriermellin`.

extractinvfeathisto

`extractinvfeathisto` extracts invariant feature histograms as described in Section 4.4.1. Invocation is done by

```
extractinvfeathisto [options] (-frgb|-fgrey|-rel) <filename>
```

Available options are:

`-suffix <suffix>` to specify the suffix of the output file. (default: `.rgbfeat` if `-frgb`, `.greyfeat` if `-fgrey`, and `.relhisto` if `-rel`.

`-featurefunction <featurefunction>` to specify the feature function used. Available feature functions are

- `X01X20` uses $f(X) = \sqrt{X(0,1) \cdot X(2,0)}$ (default).
- `X01X100` uses $f(X) = \sqrt{X(0,1) \cdot X(10,0)}$.
- `X01-X100` uses $f(X) = X(0,1) - X(10,0)$.
- `hsvparse*:X1=`x_1`:Y1=`y_1`:X2=`x_2`:Y2=`y_2 uses $f(X) = \sqrt{X(x_1,y_1) \cdot X(x_2,y_2)}$ for HSV images.
- `parse*:X1=`x_1`:Y1=`y_1`:X2=`x_2`:Y2=`y_2 uses $f(X) = \sqrt{X(x_1,y_1) \cdot X(x_2,y_2)}$.
- `parse3x:X1=`x_1`:Y1=`y_1`:X2=`x_2`:Y2=`y_2`:X3=`x_3`:Y3=`y_3 uses $f(X) = \sqrt[3]{X(x_1,y_1) \cdot X(x_2,y_2) \cdot X(x_3,y_3)}$.
- `parse-:X1=`x_1`:Y1=`y_1`:X2=`x_2`:Y2=`y_2 uses $f(X) = X(x_1,y_1) - X(x_2,y_2)$.

`-steps <steps>` specifies the number of steps for the created histogram. This is the number of steps per dimension. That is, for a color image a histogram with `<steps>`^{3} bins is created and for gray images `<steps>` bins are created.

`-samples <samples>` specifies the number of samples taken for Monte Carlo integration. If not specified integration is not done by Monte Carlo integration but exact.

`-scalingStart <value>` specifies the smallest scale factor to use.
(default: 1.0=original size).

`-scalingMulti <value>` specifies the factor by which the scale factor is multiplied each iteration (default: 1.1).

`-scalingStop <value>` specifies the largest scale factor to use (default: 1.0=original size).

`-rotationAngle <value>` specifies the rotation angle for integration. (default: 360=full circle)

With `-frgb` the input image is read as color image and a multi dimensional histogram is created, with `-fgrey` the input image is read as gray image and a one dimensional histogram is created. With `-rel` color or gray does not matter and a three dimensional histogram is created of the relational histograms. To obtain scale invariant features, the image size and the feature function used have to be considered. It is necessary to choose the scaling parameters such that the smallest image considered is smaller than the support size of the function, and for the largest image considered it is necessary, that one pixel is larger than the support size of the function. Let X be an image of size 384×256, and let $f(x) = \sqrt{X(4,0) \cdot X(0,8)}$. The image has to be scaled from 8×8 up to the eightfold of its original size: 3072×2048.

extractinvfeatvec

`extractinvfeatvec` is a program to extract invariant feature vectors as described in Section 4.4.2. It is invoked by

```
extractinvfeatvec [options] (-cvec|-gvec) <filename>.
```

Available options are

`-suffix <suffix>` to specify a suffix for the output file
(default: `.colorinvfeatvec/.greyinvfeatvec`)

`-samples <samples>` specifies the number of samples taken for Monte Carlo integration. If not specified integration is not approximative but exact.

`-cvec` specifies that the image is read as color image, `-gvec` specifies that the image is read as gray image. Depending on this the resulting vector has a different number of entries: 44 for grey images and 132 for color images.

extractlfv2

`extractlfv2` is a program to extract local features from images. It is invoked by

```
extractlfv2 [options] (-color|-gray) -img <filename>
```

Available options are

`-winsize <number>` specifies the size of the extracted local features. A value of n results in local features of size $(2n + 1) \times (2n + 1)$.

`-threshold <value>` specifies the variance threshold to determine which local features are extracted.

`-nOfFeatures <number>` specifies the number of local features to be extracted. If this is specified, the option `-threshold` is ignored.

`-subsampling <number>` specifies the step size between two pixels considered. By default this is 1 which means, that each pixel is considered.

`-images` if this is set, the local features are additionally saved as PNG images (default: not set).

`-padding` if this is set, the image is padded to be large enough to allow local feature to be extracted from each pixel. If this is not set local features are only extracted from the center points (default: not set).

`-suffix <suffix>` this specifies the suffix for the output file containing the local features as text.

em-segmentation

`em-segmentation` is the program to create region features as described in Section 4.10. It is invoked by

```
em-segmentation [options] -file <filename>
```

Available options are

`-suffix <suffix>` to specify the suffix of the file where the region features are written to (default: `.regions`).

`-maxSplits <int>` to specify the number of splits done in LBG clustering (default: 3)

`-iter <int>` to specify the number of reestimation iterations between two splits (default: 10).

`-minObs <int>` to specify the minimal number of observations in a cluster. If a cluster contains less observations it is deleted (default: 4).

`-epsilon <double>` to specify the epsilon used for splitting (default: 0.1).

`-disturbMode <disturbMode>` to specify the way of disturbing means while splitting. Available modi are: `varianceDisturb` (default), `meanDisturb`, `meanDisturb2`, `constDisturb`.

`-poolMode <poolMode>` to specify if and how variances are pooled in the cluster process. Available options: `clusterPooling`, `dimensionPooling`, and `noPooling` (default).

`-d <distance>` to specify the used distance to compare pixel features (default: euclidean). For a list of available distance measures see Table A.1.

`-smoothRange <int>` to specify the size of the smoothing operator in post processing

If this program is started, a segmentation is created and some data about the segments is saved to an output file. The output file has the same name as the input image concatenated with the given suffix. Additionally an image showing the regions is saved with the same name concatenated with `.png`.

blobworld-matlab source

From http://elib.cs.berkeley.edu/photos/blobworld/ it is possible to download the sources of the BlobWorld feature extraction. This is a MatLab program and we used it to generate these features. Additionally a small program to convert the resulting MatLab files to a simple text-based format has been developed.

lf-pca

`lf-pca` is the program to apply the PCA dimensionality reduction to a set of files with local features or gabor features. It is invoked by

```
lf-pca [options] <filelist>
```

where `<filelist>` specifies the list of files containing local features to be processed. Available options are:

`-transform <filename>` this options specifies that the PCA matrix does not have to be estimated but is loaded from the specified file.

`-saveto <filename>` this options specifies, that the PCA matrix has to be estimated and afterwards is saved to the specified file. This cannot be used together with `-transform`.

`-outDim <number>` specifies the dimensionality of the output data.

When this program is started, normally all files from the file list are read, the mean and the covariance matrix from this data is estimated and transformed for PCA transformation using singular value decomposition (SVD). Then the data is read again and transformed using the matrix. The transformed data is saved into files with the same name as the input files appended with `.pca`.

dbpcatransform

`dbpcatransform` is a program to apply PCA dimensionality reduction to nearly any type of feature. It is invoked with

```
dbpcatransform -toDim <dim> -filelist <filelist>
```

When the program is started it reads all files specified in the file list, calculates the covariance matrices and the means for the different types of features, transforms these using SVD and transforms the specified files. The output files are written to files with the same names as the input files concatenated with `.pca`.

gaborize

`gaborize` is the program to extract Gabor features as described in Section 4.5. It is invoked by

```
gaborize [options] filename
```

Available options are

`-numpha <number>` to specify the number of different phases used (default: 5).

`-numfreq <number>` to specify the number of different frequencies used (default: 5).

`-threshold <value>` to specify the local variance threshold for feature extraction
(cp. `extractlfv2`)

`-winsize <number>` to specify the window size for local variance calculation
(cp. `extractlfv2`)

`-all` to extract all gabor features. If this is specified `-threshold`, `-winsize`, and
`-nOfFeatures` is ignored.

`-nOfFeatures <number>` to specify the number of gabor features extracted.
Overrides `-threshold`.

`-color or -grey` To specify whether the image is a color or a gray image.

When the program is started the specified image is read, and Fourier transformed. Then the Fourier transform is multiplied with different filters and retransformed. From these images the Gabor feature vectors are extracted and saved to a file with the same name as the input file with `.gaborfeatures` appended.

Appendix B

Additional Tables

Table B.1: Error rates for different features on WANG database. A selection from this table is presented and explained in Section 8.2.2.

Feature	Distance	ER
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	JSD	15.9
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$ scaling	JSD	16.1
InvFeatHisto $f(X) = \sqrt[3]{X(2,0) \cdot X(4,4) \cdot X(0,8)}$	JSD	16.4
InvFeatHisto $f(X) = \sqrt[3]{X(2,0) \cdot X(4,4) \cdot X(0,8)}$	L_1	16.4
Color Histogram	JSD	17.9
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	L_1	18.2
InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(2,0)}$	JSD	18.3
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	L_1	18.4
InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(10,0)}$	JSD	19.9
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	JSD	19.9
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	JSD	20.0
Color Histogram	L_1	21.0
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	L_1	21.3
InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(2,0)}$	L_1	23.2
InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(10,0)}$	L_1	25.1
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(30,30)}$	JSD	25.2
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	L_1	25.5
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(30,30)}$	L_1	27.8
PseudoMDColor Histogram	JSD	27.8
PseudoMDColor Histogram	L_1	29.0
Tamura Histogram	JSD	31.0
Tamura Histogram	L_1	32.1
LF Histogram 256	JSD	32.5
LF Histogram 256	L_1	34.2
LF Histogram 128	JSD	35.9
InvFeatHisto $f(X) = \sqrt{X(0,1) - X(10,0)}$	JSD	36.5
LF Histogram 128	L_1	37.0

continued on next page

Feature	Distance	ER
LF Histogram 64	JSD	37.2
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(4,0)}$	JSD	37.7
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(4,0)}$	L_1	38.9
InvFeatHisto PCA($f(X) = \sqrt{X(4,0) \cdot X(0,8)}$)	Euclidean	39.0
LF Histogram 64	L_1	40.7
PCA(Color Histogram)	Euclidean	43.7
ColorInvFeatVec	Euclidean	44.9
InvFeatHisto $f(X) = X(0,1) - X(10,0)$	L_1	46.9
PCA(InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(10,0)}$)	Euclidean	47.7
PCA(InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(2,0)}$)	Euclidean	47.8
PCA(InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$)	Euclidean	47.8
Gabor Histogram 256	JSD	48.2
PCA(InvFeatHisto $f(X) = \sqrt{X(0,0)X(30,30)}$)	Euclidean	49.3
Gabor Histogram 128	JSD	49.9
PCA(InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$)	Euclidean	50.2
PCA(InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$)	Euclidean	50.7
PCA(InvFeatHisto $f(X) = \sqrt{X(0,2) \cdot X(4,4) \cdot X(8,0)}$	Euclidean	51.0
Gabor Histogram 256	L_1	51.2
PCA(PseudoMDColor Histogram)	Euclidean	51.3
Global texture feature	Euclidean	51.4
Gabor Histogram 128	L_1	51.7
Gabor Histogram 64	L_1	51.8
Gabor Histogram 64	JSD	52.8
Regions (max 4, smooth 5)	irm	54.3
32x32	Euclidean	55.1
Regions (max 8, smooth 10)	irm	55.3
32x32, gray	Euclidean	56.1
Regions (max 4, smooth 10)	irm	56.1
PCA(Tamura histogram)	Euclidean	56.3
PCA(ColorInvFeatVec)	Euclidean	56.7
PCA(32x32)	Euclidean	62.5
PCA(InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(4,0)}$)	Euclidean	68.8
PCA(InvFeatHisto $f(X) = \sqrt{X(0,1) - X(10,0)}$)	Euclidean	71.5
PCA(Global texture feature)	Euclidean	73.3
local features	glfd	87.5

Table B.2: Error rates for different features on IRMA-1617 database. A selection from this table is presented and explained in Section 8.2.2.

Feature	Distance	ER
32x32	IDM Sobel	6.7
LF(5x5) histogram 512	JSD	9.3
LF(5x5) histogram 256	JSD	9.5
LF(5x5) histogram 128	JSD	10.1
LF(5x5) histogram 64	JSD	11.6
LF(3x3) histogram 256	JSD	11.7
LF(3x3) histogram 128	JSD	12.6
LF(19x19)	lf-l1o	13.0
LF(3x3) histogram 64	JSD	14.3
LF(3x3) histogram 512	JSD	17.7
32x32	Euclidean	17.7
Tamura histogram	JSD	19.3
Gabor features	glfd	20.6
Tamura histogram	L_1	20.9
32x32 Tamura histogram	L_1	21.3
InvFeatHisto $f(X) = \text{rel}(X(0,0) - X(0,4))$	JSD	22.6
LF(5x5)	glfd	22.9
LF(5x5)	lfd	23.1
LF(3x3) histogram 1024	JSD	23.2
Gabor histogram 256	JSD	24.4
LF(19x19) histogram 256	JSD	24.6
32x32 Tamura image	Euclidean	24.6
Gabor histogram 256	JSD	25.4
Gabor histogram 128	JSD	25.6
Gabor histogram 256	JSD	25.7
Gabor histogram 256	L_1	25.8
LF(19x19)	glfd	26.4
InvFeatHisto $f(X) = \text{rel}(X(0,0), X(4,0))$	L_1	26.9
Gabor histogram 128	L_1	27.0
LF(19x19) histogram 128	JSD	28.0
LF(3x3)	lfd	28.4
LF(5x5)	lf-l1o	28.8
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,2)}$	JSD	29.2
Gabor histogram 64	JSD	29.3
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,2)}$	L_1	29.9
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(4,0)}$	JSD	30.1
LF(19x19) histogram 64	JSD	30.2
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(4,0)}$	L_1	30.3
Image size	Euclidean	30.4
Gabor histogram 64	L_1	30.4

continued on next page

Feature	Distance	ER
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,8)}$	JSD	31.0
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,8)}$	L_1	31.2
LF(3x3)	lf-l1o	31.7
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	JSD	31.9
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,32)}$	L_1	32.1
InvFeatHisto $f(X) = \sqrt{X(4,0) \cdot X(0,8)}$	L_1	32.1
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,16)}$	JSD	32.2
LF(3x3)	glfd	32.8
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,16)}$	L_1	33.0
InvFeatHisto $f(X) = \sqrt{X(0,0) \cdot X(0,32)}$	JSD	33.9
InvFeatHisto $f(X) = \sqrt{X(0,1) \cdot X(2,0)}$	L_1	38.1
InvFeatHisto $f(X) = \sqrt[3]{X(0,2) \cdot X(4,4) \cdot X(8,0)}$	JSD	41.5
InvFeatHisto $f(X) = \sqrt[3]{X(0,2) \cdot X(4,4) \cdot X(8,0)}$	L_1	44.2
InvFeatVector	Euclidean	52.9
Fourier Mellin Feature	Euclidean	53.1

Table B.3: Error rates for training and classification task for different features on IRMA-3879 database.

Feature	Distance	L1O ER on training data		ER test with train	
		8 classes	26 classes	8 classes	26 classes
Image 32×32	Euclidean	16.7	17.9	20.8	21.7
RelFeatHisto	JSD	20.2	19.5	24.9	25.4
InvFeatHisto	JSD	39.7	38.6	44.2	43.3
InvFeatHisto (scaling)	JSD	39.5	39.0	43.8	44.0
Image size	JSD	33.2	57.3	38.8	63.8
InvFeatVec	Euclidean	82.4	82.3	98.9	98.8
Tamura Histogram	JSD	19.0	18.6	23.1	23.3
Tamura Image	Euclidean	23.8	25.5	26.4	27.1
small Tamura Histogram	JSD	21.5	22.9	25.4	27.0

Table B.4: Error rates for different dissimilarity measures on WANG database for invariant feature histogram with $f(X) = \sqrt{X(4,0) \cdot X(8,0)}$. A seleciton of this table is presented and explained in Section 8.2.1.

Disssimilarity measure	ER
$d_{\log(2-F)}(\cdot,\cdot)$	15.6
$d_{\sqrt{1-F}}(\cdot,\cdot)$	15.6
$d_{\sqrt{1-F}}(\cdot,\cdot)$	15.6
$d_{JSD}(\cdot,\cdot)$	15.9
$d_{\chi^2}(\cdot,\cdot)$	16.5
$d_{rbd}(\cdot,\cdot)$	17.4
$d_{his}(\cdot,\cdot)$	18.4
$d_1(\cdot,\cdot)$	18.4
$d_{rd}(\cdot,\cdot)$	25.6
$d_t(\cdot,\cdot)$	27.6
$d_2(\cdot,\cdot)$	28.3
$d_{\arccos F}(\cdot,\cdot)$	31.5
$d_{\sin F}(\cdot,\cdot)$	31.5
$d_{KLD}(\cdot,\cdot)$	78.4

Table B.5: Error rates for different dissimilarity measures on IRMA-1617 database for local feature histograms of 5×5 local features with 512 bins. A selection of this table is presented and explained in Section 8.2.1.

Dissimilarity measure	ER
$d_{his}(\cdot,\cdot)$	8.3
$d_1(\cdot,\cdot)$	8.3
$d_{\chi^2}(\cdot,\cdot)$	9.1
$d_{JSD}(\cdot,\cdot)$	9.3
$d_{\log(2-F)}(\cdot,\cdot)$	9.5
$d_{\sqrt{1-F}}(\cdot,\cdot)$	9.5
$d_{\sqrt{1-F}}(\cdot,\cdot)$	9.5
$d_{rd}(\cdot,\cdot)$	11.7
$d_{rbd}(\cdot,\cdot)$	12.0
$d_2(\cdot,\cdot)$	14.2
$d_{tw}(\cdot,\cdot)$	14.2
$d_{\arccos F}(\cdot,\cdot)$	45.8
$d_{\sin F}(\cdot,\cdot)$	45.8
$d_{KLD}(\cdot,\cdot)$	74.1

Bibliography

[Barnard & Duygulu$^+$ 01] K. Barnard, P. Duygulu, D. Forsyth. Clustering Art. Proc. *Computer Vision and Pattern Recognition*, Vol. 2, pp. 435–439, Kauai, Hawai, Dec. 2001.

[Barnard & Duygulu$^+$ 02] K. Barnard, P. Duygulu, D. Forsyth. Modeling the Statistics of Image Features and Associated Text. Proc. *Document Recognition and Retrieval*, Electronic Imaging, San Jose, CA, Jan. 2002.

[Barnard & Forsyth 01] K. Barnard, D. Forsyth. Learning the Semantics of Words and Pictures. Proc. *International Conference on Computer Vision*, Vol. 2, pp. 408–415, Vancouver, Canada, July 2001.

[Berkhin 02] P. Berkhin. Survey of Clustering Data Mining Techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[Brill & Barrett$^+$ 92] M. Brill, E. Barrett, P. Payton. *Projective Invariants for Curves in Two and Three Dimensions*, chapter 9 in Geometric Invariance in Computer Vision, pp. 193–214. MIT Press, 1992.

[Carson & Belongie$^+$ 02] C. Carson, S. Belongie, H. Greenspan, J. Malik. Blobworld: Image Segmentation Using Expectation-Maximization and its Application to Image Querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 8, pp. 1026–1038, Aug. 2002.

[Carson & Thomas$^+$ 99] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, J. Malik. Blobworld: A System for Region-Based Image Indexing and Retrieval. Proc. *International Conference on Visual Information Systems*, pp. 509–516, Amsterdam, The Netherlands, June 1999. Springer Verlag.

[Chen & Wang 02] Y. Chen, J. Z. Wang. Region-Based Fuzzy Feature Matching Approach to Content-Based Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 9, pp. 1252–1267, Sept. 2002.

[Dahmen & Hektor$^+$ 00] J. Dahmen, J. Hektor, R. Perrey, H. Ney. Automatic Classification of Red Blood Cells using Gaussian Mixture Densities. Proc. *Bildverarbeitung für die Medizin*, pp. 331–335, Munich, Germany, March 2000.

[Deselaers & Keysers$^+$ 03a] T. Deselaers, D. Keysers, H. Ney. Clustering Visually Similar Images to Improve Image Search Engines. Proc. *Informatiktage 2003 der Gesellschaft für Informatik*, to appear, Bad Schussenried, Germany, Nov. 2003.

[Deselaers & Keysers^{+} 03b] T. Deselaers, D. Keysers, R. Paredes, E. Vidal, H. Ney. Local Representations for Multi-Object Recognition. Proc. *DAGM 2003, Pattern Recognition, 25th DAGM Symposium*, Vol. 2781 of *Lecture Notes in Computer Science*, pp. 305–312, Magdeburg, Germany, Sept. 2003. Springer Verlag.

[Duda & Hart^{+} 01] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification.* John Wiley & Sons, New York, NY, 2nd edition, 2001.

[Duygulu & Barnard^{+} 02] P. Duygulu, K. Barnard, N. de Freitas, D. Forsyth. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. Proc. *European Conference on Computer Vision*, Vol. 9, pp. 97–112, Copenhagen, Denmark, May 2002.

[Efron & Tibshirani 93] B. Efron, R. J. Tibshirani. *An Introduction to the Bootstrap.* Chapman & Hall, New York, 1993.

[Faloutsos & Barber^{+} 94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, W. Equitz. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems*, Vol. 3, No. 3/4, pp. 231–262, July 1994.

[Fei-Fei & Fergus^{+} 03] L. Fei-Fei, R. Fergus, P. Perona. A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories. Proc. *International Conference on Computer Vision*, Vol. 1, pp. 1134–1141, Nice, France, Oct. 2003.

[Fend & Siu^{+} 03] D. Fend, W. Siu, H. Zhang, editors. *Multimedia Information Retrieval and Management – Technological Fundamentals and Applications.* Springer Verlag, 2003.

[Fergus & Perona^{+} 03] R. Fergus, P. Perona, A. Zissermann. Object Class Recognition by Unsupervised Scale-Invariant Learning. Proc. *Conference on Computer Vision and Pattern Recognition*, pp. 264–271, Blacksburg, VG, June 2003.

[Frey & Jojic 03] B. J. Frey, N. Jojic. Transformation-Invariant Clustering Using the EM Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 1, pp. 1–17, Jan. 2003.

[Güld & Schubert^{+} 03] M. Güld, H. Schubert, M. Leisten, B. Plodowski, B. Fischer, D. Keysers, T. Lehmann, B. Wein. Automatische Kategorisierung von medizinischem Bildmaterial in einen multi-axialen mono-hierarchischen Code. Proc. *Bildverarbeitung für die Medizin*, pp. 388–392, 2003.

[Gollan 03] C. Gollan. Nichtlineare Verformungsmodelle für die Bilderkennung. Diploma thesis, Lehrstuhl für Informatik VI, RWTH Aachen, Aachen, Germany, Sept. 2003.

[Graczyk 95] C. Graczyk. Vision Texture, March 1995.
http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.

[Gu & Duncan^{+} 89] Z. Q. Gu, C. N. Duncan, E. Renshaw, M. A. Mugglestone, C. F. N. Cowan, P. M. Grant. Comparison of Techniques for Measuring Cloud Texture in Remotely Sensed Satellite Meteorological Image Data. *Radar and Signal Processing*, Vol. 136, No. 5, pp. 236–248, Oct. 1989.

[Haberäcker 95] P. Haberäcker. *Praxis der Digitalen Bildverarbeitung und Mustererkennung.* Carl Hanser Verlag, München, Wien, 1995.

[Haralick & Shanmugam$^+$ 73] R. M. Haralick, B. Shanmugam, I. Dinstein. Texture Features for Image Classification. *IEEE Transaction on Systems, Man, and Cybernetics*, Vol. 3, No. 6, pp. 610–621, Nov. 1973.

[Iqbal & Aggarwal 99] Q. Iqbal, J. Aggarwal. Using Structure in Content-Based Image Retrieval. Proc. *International Conference Signal and Image Processing*, pp. 129–133, Nassau, Bahamas, Oct. 1999.

[Iqbal & Aggarwal 02] Q. Iqbal, J. Aggarwal. CIRES: A System for Content-Based Retrieval in Digital Image Libraries. Proc. *International Conference on Control, Automation, Robotics and Vision*, pp. 205–210, Singapore, Dec. 2002.

[Iyengar & Lippman 98] G. Iyengar, A. Lippman. Clustering Images Using Relative Entropy for Efficient Retrieval. Proc. *Workshop on Very Low bitrate Video Coding*, Urbana, IL, Oct. 1998.

[Jain & Dubes 88] A. K. Jain, R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[Jain & Murty$^+$ 99] A. K. Jain, M. N. Murty, P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264–323, March 1999.

[Keysers & Dahmen$^+$ 03] D. Keysers, J. Dahmen, H. Ney, B. Wein, T. Lehmann. Statistical Framework for Model-Based Image Retrieval in Medical Applications. *Journal of Electronic Imaging*, Vol. 12, No. 1, pp. 59–68, Jan. 2003.

[Keysers & Macherey$^+$ 01] D. Keysers, W. Macherey, J. Dahmen, H. Ney. Learning of Variability for Invariant Statistical Pattern Recognition. Proc. *European Conference on Machine Learning*, Vol. 2167 of *Lecture Notes in Computer Science*, pp. 263–275, Freiburg, Germany, Sept. 2001. Springer Verlag.

[Keysers & Motter$^+$ 03] D. Keysers, M. Motter, T. Deselaers, H. Ney. Training and Recognition of Complex Scenes using a Holistic Statistical Model. Proc. *DAGM 2003, Pattern Recognition, 25th DAGM Symposium*, Vol. 2781 of *Lecture Notes in Computer Science*, pp. 52–59, Magdeburg, Germany, Sept. 2003. Springer Verlag.

[Keysers & Och$^+$ 02] D. Keysers, F. J. Och, H. Ney. Efficient Maximum Entropy Training for Statistical Object Recognition. Proc. *Informatiktage 2002 der Gesellschaft für Informatik*, pp. 342–345, Bad Schussenried, Germany, Nov. 2002.

[Keysers 99] D. Keysers. Texturanalyse von Farbbildern mit Gaborfiltern. Studienarbeit, Institut für Medizinische Informatik, RWTH Aachen, Aachen, Germany, 1999.

[Keysers 00] D. Keysers. Approaches to Invariant Image Object Recognition. Diploma thesis, Lehrstuhl für Informatik VI, RWTH Aachen, Aachen, Germany, June 2000.

[Kittler 98] J. Kittler. On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 226–239, March 1998.

[Kölsch 03] T. Kölsch. Local Features for Image Classification. Diploma thesis, Lehrstuhl für Informatik VI, RWTH Aachen, Aachen, Germany, Nov. 2003.

[Käster & Wendt+ 03] T. Käster, V. Wendt, G. Sagerer. Comparing Clustering Methods for Database Categorization in Image Retrieval. Proc. *DAGM 2003, Pattern Recognition, 25th DAGM Symposium*, Vol. 2781 of *Lecture Notes in Computer Science*, pp. 228–235, Magdeburg, Germany, Sept. 2003. Springer Verlag.

[Lehmann & Güld+ 03] T. Lehmann, M. Güld, C. Thies, B. Fischer, K. Spitzer, D. Keysers, H. Ney, M. Kohnen, H. Schubert, B. Wein. The IRMA Project – A State of the Art Report on Content-Based Image Retrieval in Medical Applications. Proc. *Korea-Germany Joint Workshop on Advanced Medical Image Processing*, pp. 161–171, 2003.

[Linde & Buzo+ 80] Y. Linde, A. Buzo, R. Gray. An Algorithm for Vector Quantization Design. Proc. *IEEE Transactions on Communications*, Vol. 28, pp. 84–95, Jan. 1980.

[Markkula & Sormunen 98] M. Markkula, E. Sormunen. Searching for Photos - Journalists' Practices in Pictorial IR. Proc. *Electronic Workshops in Computing – Challenge of Image Retrieval*, pp. 1–13, Newcastle, UK, Feb. 1998.

[Müller & Müller+] H. Müller, W. Müller, S. Marchand-Maillet, D. M. Squire. Strategies for positive and negative Relevance Feedback in Image Retrieval.

[Müller & Müller+ 01] H. Müller, W. Müller, D. M. Squire, S. Marchand-Maillet, T. Pun. Performance Evaluation in Content-Based Image Retrieval: Overview and Proposals. *Pattern Recognition Letters (Special Issue on Image and Video Indexing)*, Vol. 22, No. 5, pp. 593–601, 2001. H. Bunke and X. Jiang Eds.

[Nene & Nayar+a] S. A. Nene, S. K. Nayar, H. Murase. Columbia Object Image Library (COIL-100). Technical Report 6, Department of Computer Science, Columbia University, New York, NY, 96.

[Nene & Nayar+b] S. A. Nene, S. K. Nayar, H. Murase. Columbia Object Image Library (COIL-20). Technical Report 5, Department of Computer Science, Columbia University, New York, NY, 96.

[Ney 99] H. Ney. Mustererkennung und Neuronale Netze. Script to the Lecture on Pattern Recognition and Neural Networks Held at RWTH Aachen, 1999.

[Nölle 03] M. Nölle. Distribution Distance Measures Applied to 3-D Object Recognition – A Case Study. Proc. *DAGM 2003, Pattern Recognition, 25th DAGM Symposium*, Vol. 2781 of *Lecture Notes in Computer Science*, pp. 84–91, Magdeburg, Germany, Sept. 2003. Springer Verlag.

[Obdrzalek & Matas 03] S. Obdrzalek, J. Matas. Image Retrieval Using Local Compact DCT-Based Representation. Proc. *DAGM 2003, Pattern Recognition, 25th DAGM Symposium*, Vol. 2781 of *Lecture Notes in Computer Science*, pp. 490–497, Magdeburg, Germany, Sept. 2003. Springer Verlag.

[Pal & Pal 93] N. R. Pal, S. K. Pal. A Review on Image Segmentation Techniques. *Pattern Recognition*, Vol. 26, No. 9, pp. 1277–1294, Nov. 1993.

[Palm & Keysers$^+$ 00] C. Palm, D. Keysers, T. Lehmann, K. Spitzer. Gabor Filtering of Complex Hue/Saturation Images for Color Texture Classification. Proc. *International Conference on Computer Vision, Pattern Recognition, and Image Processing*, Vol. 2, pp. 45–49, Atlantic City, NJ, Feb. 2000.

[Paredes & Perez-Cortes$^+$ 01] R. Paredes, J. Perez-Cortes, A. Juan, E. Vidal. Local Representations and a Direct Voting Scheme for Face Recognition. Proc. *Workshop on Pattern Recognition in Information Systems*, pp. 71–79, Setúbal, Portugal, July 2001.

[Park & Jin$^+$ 02] M. Park, J. S. Jin, L. S. Wilson. Fast Content-Based Image Retrieval Using Quasi-Gabor Filter and Reduction of Image Feature. Proc. *Southwest Symposium on Image Analysis and Interpretation*, pp. 178–182, Santa Fe, NM, April 2002.

[Petrakis & Faloutsos$^+$ 02] E. Petrakis, C. Faloutsos, K. Ip, D. Lin. ImageMap: An Image Indexing Method Based on Spatial Similarity. *IEEE Transactions on Knowledge and Data Engeneering*, Vol. 14, No. 5, pp. 979–987, Sept. 2002.

[Porter 80] M. Porter. *An Algorithm for Suffix Stripping.* Morgan Kaufmann, San Francisco, CA, 1980.

[Puzicha & Rubner$^+$ 99] J. Puzicha, Y. Rubner, C. Tomasi, J. Buhmann. Empirical Evaluation of Dissimilarity Measures for Color and Texture. Proc. *International Conference on Computer Vision*, Vol. 2, pp. 1165–1173, Corfu, Greece, Sept. 1999.

[Rao 90] A. R. Rao. *A Taxonomy for Texture Description and Identification.* Springer Verlag, 1990.

[Reddy & Chatterji 96] B. S. Reddy, B. Chatterji. An FFT-Based Technique for Translation, Rotation and Scale invariant Image Registration. *IEEE Transactions on Image Processing*, Vol. 5, Aug. 1996.

[Reinhold & Paulus$^+$ 01] M. Reinhold, D. Paulus, H. Niemann. Appearance-Based Statistical Object Recognition by Heterogenous Background and Occlusions. Proc. *DAGM 2001, Pattern Recognition, 23rd DAGM Symposium*, number 2191 in Lecture Notes in Computer Science, pp. 254–261, Munich, Germany, Sept. 2001. Springer Verlag.

[Rubner & Tomasi$^+$ 98] Y. Rubner, C. Tomasi, L. J. Guibas. A Metric for Distributions with Applications to Image Databases. Proc. *International Conference on Computer Vision*, pp. 59–66, Bombay, India, Jan. 1998.

[Rui & Huang$^+$ 99] Y. Rui, T. Huang, S. Chang. Image Retrieval: Current Techniques, Promising Directions and Open Issues. *Journal of Visual Communication and Image Representation*, Vol. 10, No. 4, pp. 39–62, April 1999.

[Saporta & Youness 02] G. Saporta, G. Youness. Comparing Two Partitions: Some Proposals and Experiments. Proc. *Conference on Computational Statistics*, Berlin, Germany, Aug. 2002.

[Saux & Boujemaa 02a] B. L. Saux, N. Boujemaa. Unsupervised Robust Clustering for Image Database Categorization. Proc. *International Conference on Pattern Recognition*, Vol. 1, pp. 259–263, Aug. 2002.

[Saux & Boujemaa 02b] B. L. Saux, N. Boujemaa. Unsupervised Categorization for Image Database Overview. Proc. *Recent Advances in Visual Information Systems / International Conference on Visual Information System*, Vol. 2314 of *Lecture Notes in Computer Science*, pp. 163–174, Hsin-Chu, Taiwan, March 2002. Springer Verlag.

[Schael 01] M. Schael. Texture Defect Detection Using Invariant Textural Features. Proc. *DAGM 2001, Pattern Recogntion, 23rd DAGM Symposium*, Lecture Notes in Computer Science, pp. 17–24, 2001.

[Schölkopf 97] B. Schölkopf. *Support Vector Learning.* Oldenbourg Verlag, Munich, Germany, 1997.

[Schäuble 97] P. Schäuble. *Multimedia Information Retrieval*, chapter 1.6 Evaluation Issues, pp. 22–32. Kluwer Academic Publishers, 1997.

[Schulz-Mirbach 95] H. Schulz-Mirbach. Invariant Features for Gray Scale Images. Proc. *DAGM – Symposium – "Mustererkennung"*, pp. 1–14, Sept. 1995.

[Shao & Svoboda$^+$ 03a] H. Shao, T. Svoboda, T. Tuytelaars, L. V. Gool. HPAT Indexing for Fast Object/Scene Recognition Based on Local Appearance. Proc. *Conference on Image and Video Retrieval*, pp. 71–80, Urbana-Champaign, IL, July 2003.

[Shao & Svoboda$^+$ 03b] H. Shao, T. Svoboda, L. van Gool. ZuBuD – Zurich Buildings Database for Image Based Recognition. Technical Report Technical Report No. 260, Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland, Zurich, Switzerland, April 2003.

[Siggelkow & Burkhardt 97] S. Siggelkow, H. Burkhardt. Local Invariant Feature Histograms for Texture Classification. Technical Report 3, University of Freiburg, Institute for Computer Science, 1997.

[Siggelkow & Schael 99] S. Siggelkow, M. Schael. Fast Estimation of Invariant Features. Proc. *DAGM – Symposium – Mustererkennung*, Informatik aktuell, pp. 181–188, Bonn, Germany, Sept. 1999.

[Siggelkow & Schael$^+$ 01] S. Siggelkow, M. Schael, H. Burkhardt. SIMBA — Search IMages By Appearance. Proc. *DAGM 2001, Pattern Recognition, 23rd DAGM Symposium*, Vol. 2191 of *Lecture Notes in Computer Science*, pp. 9–17, Munich, Germany, Sept. 2001. Springer Verlag.

[Siggelkow 02] S. Siggelkow. *Feature Histograms for Content-Based Image Retrieval.* Ph.D. thesis, University of Freiburg, Institute for Computer Science, Freiburg, Germany, 2002.

[Smeulders & Worring$^+$ 00] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain. Content-Based Image Retrieval: The End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 12, pp. 1349–1380, Dec. 2000.

[Squire & Caelli 00] D. M. Squire, T. M. Caelli. Invariance Signatures: Characterizing Contours by Their Departures from Invariance. *Computer Vision and Image Understanding*, Vol. 77, No. 3, pp. 284–316, March 2000.

[Squire & Müller$^+$ 99] D. M. Squire, W. Müller, H. Müller, J. Raki. Content-Based Query of Image Databases, Inspirations from Text Retrieval: Inverted Files, Frequency-Based Weights and Relevance Feedback. Proc. *Scandinavian Conference on Image Analysis*, pp. 143–149, Kangerlussuaq, Greenland, June 1999.

[Starik & Selding$^+$ 03] S. Starik, Y. Selding, M. Wermann. Unsupervised Clustering of Images Using their Joint Segmentation. Technical Report 37, Leibniz Center for Research in Computer Science, Jerusalem, Israel, June 2003.

[Subrahmanian & Tripathi 98] V. Subrahmanian, S. Tripathi, editors. *Multimedia Information Systems*, chapter 5, Image Databases. Kluwer, July 1998.

[Swain & Ballard 91] M. J. Swain, D. H. Ballard. Color Indexing. *International Journal of Computer Vision*, Vol. 7, No. 1, pp. 11–32, Nov. 1991.

[Tamura & Mori$^+$ 78] H. Tamura, S. Mori, T. Yamawaki. Textural Features Corresponding to Visual Perception. *IEEE Transaction on Systems, Man, and Cybernetcs*, Vol. SMC-8, No. 6, pp. 460–472, June 1978.

[Terhorst 03] B. Terhorst. Texturanalyse zur globalen Bildinhaltsbeschreibung radiologischer Aufnahmen. Research project, RWTH Aachen, Institut für Medizinische Informatik, Aachen, Germany, June 2003.

[Wang & Li$^+$ 01] J. Z. Wang, J. Li, G. Wiederhold. SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture LIbraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 9, pp. 947–963, Sept. 2001.

[Weber 00] M. Weber. *Unsupervised Learning of Models for Object Recognition.* Ph.D. thesis, California Institute of Technology, Pasadena, CA, 2000.

[Zhang & Wong$^+$ 00] D. Zhang, A. Wong, M. Indrawan, G. Lu. Content-Based Image Retrieval Using Gabor Texture Features. Proc. *Pacific-Rim Conference on Multimedia*, pp. 392–395, Sydney, Australia, Dec. 2000.

[Zucker & Terzopoulos 80] S. W. Zucker, D. Terzopoulos. Finding Structure in Co-Occurrence Matrices for Texture Analysis. *Computer Graphics and Image Processing*, Vol. 12, pp. 286–308, Dec. 1980.

Index