Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: wolf@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-61 May 2000

Content based Image Retrieval using Interest Points and Texture Features

Christian Wolf [1]

Abstract

Content based image retrieval is the task of searching images from a database, which are visually similar to a given example image. Since there is no general definition for visual similarity, there are different possible ways to query for visual content. In this work we present methods for content based image retrieval based on texture similarity using interest points and Gabor features. Interest point detectors are used in computer vision to detect image points with special properties, which can be geometric (corners) or non-geometric (contrast etc.). Gabor functions and Gabor filters are regarded as excellent tools for texture feature extraction and texture segmentation. We present methods how to combine these methods for content based image retrieval and to generate a texture description of images. Special emphasis is devoted to distance measures for the texture descriptions. Experimental results of the query system on different test image databases are given.

[1]This work was supported by the Austrian Science Foundation under S7002-MAT

Contents

List of Figures

Chapter 1

Introduction

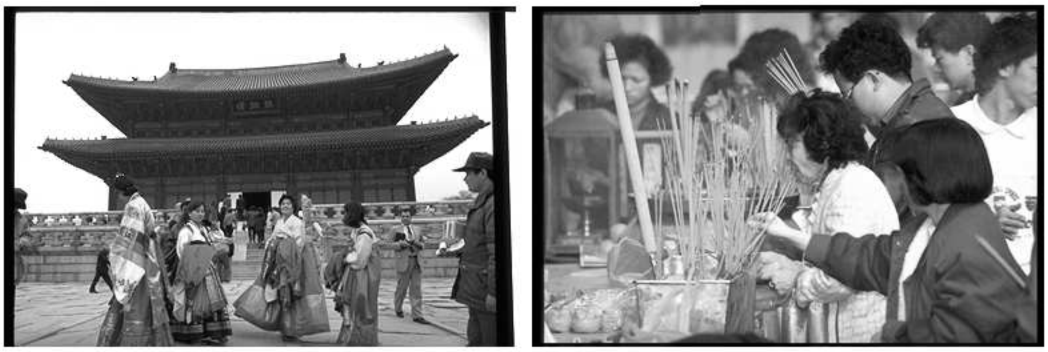
1.1 Motivation

Since its invention photography has rapidly conquered the world. No wonder, it was photography which made it possible to transfer and store visual impressions and experiences. People began to collect photographs and soon companies like newspaper publishers had large archives. However, finding specific pictures in these vast quantities of information was time consuming manual work.

The invention of the computer changed a lot about the ways information can be stored, but did not solve all problems. The equivalents of photographs — digital images — found their way into the information society as soon as the computers became powerful enough to process, display and transfer them. At the time this document is written the average desktop PC (already called "Multimedia PC") has enough memory and processing power to store large image and video databases. The hardware necessary to create and capture images is available, cheap and easy to use. Scanners, digital cameras and even digital video cameras able to connect to computers are flooding the markets and found their way into the average home. The latest technology discovered by the public is the Internet and the world wide web — an almost unlimited source of information.

The problem of finding specific pieces of information in this large archives remains unresolved. Solutions for text information have been found. Databases based on keywords or full text search are used successfully all over the world. The principle of keyword based search techniques has been applied to image databases as well. However, it is not possible to describe images with words to the same extent as contents in text form, since the information visible on images can be seen from different viewpoints or aspects.

Figure 1.1 shows two examples. How do we describe the contents of these images? People looking at Figure 1.1.a could be interested in the people or the fashion in the foreground, the architecture in the background or any other part of the image. The scene displayed in Figure 1.1.b is equally difficult to describe.



(a) (b)

Figure 1.1: How do we describe visual information?

The equivalent of a full text search for an image database is a content based image retrieval system using query by example as "query language". The user specifies a query image or parts of it. The system answers with images similar to this query image. The key question of this approach is, how do we define similarity between images? We distinguish two different approaches[27]:

- Systems based on *attentive scan* use conscious processes and high level reasoning to compare information. Similarity is derived from the knowledge about the semantic contents of the images. Humans use this concept to search for specific documents (or images) in archives.

- Systems based on *pre-attentive scan* use unconscious processes to compare information. Applying image processing algorithms they extract features of different nature: Colour, texture, structure and contour features have been used in former approaches for image descriptions.

An ideal content based query system should be based on attentive scan. Only by understanding the contents of archives *and* the desires of the user it is possible to deliver exactly the required information. However, the general image understanding problem has not yet been solved, i.e. computers and their software are not yet sophisticated enough to actually understand the semantic meaning of the contents of documents, and it is subject to scientific discussion until now if artificial processes will ever reach this degree of intelligence. Up to now all methods have been pre-attentive and the work described in this document is based on this approach as well.

The way and method how an image retrieval system is choosing and delivering its results depends on the purpose of the application. There is no general measure for visual similarity so there are no general purpose image retrieval systems for all kind of images, which always deliver correct results according to the user's wishes. The known approaches can be classified into systems more or less specialised to specific tasks and systems trying to implement a general purpose retrieval system working for all kinds of images. Examples for the former are medical systems like databases of X-ray images, applications specialised to facial images etc.

An application for a general purpose retrieval software can be found on the web: The Internet search engine Altavista [1] allows to search for text, images, video and audio contents of the web by keywords. Once an image is found the user is able to search what is called a *visual similar*. Obviously a system like this designed for general purposes will not work in all cases, since there is no feedback from the user which kind of information she desires. However, for the non business critical usage of the average user they can be useful enough. The demand of powerful image retrieval systems is not yet satisfied by existing applications. Not only the vast quantities of information available on the Internet need to be handled by search engines. Industries like TV broadcasting stations, magazines, newspapers, advertisement agencies, hospitals and even governments store huge amounts of visual information. Without being able to search for specific documents this knowledge cannot be used to the full extent.

The work described in this document introduces a new image retrieval system based on pre-attentive scan. The aim was to develop an approach based on texture similarity without performing a full texture segmentation of the images. Instead we are using local texture features gathered from a representative set of image pixels to describe images. The system can be outlined as follows:

- We select a fixed number of image pixels (called interest points) using interest point operators. The criteria for the selection of the points is how well they represent the contents of the image.

- Around each interest point we extract windows of fixed sizes from the image. From these windows, which we call interest regions, we extract texture features by applying a Gabor filter bank. The output of the Gabor filter bank is used to create different image representations.

- Finally we developed different distance measures applicable to our image representations.

The system itself has been tested on different test image databases and proved to perform well. Future work will integrate it into a bigger application (Section 2.3) using different types of similarity (colour, texture, structure and contour) and user feedback.

1.2 Structure of this Document

Chapter 2 gives an overview of the principal components of a content based image query system based on pre-attentive scan. We give a short overview of the former

[1] http://www.altavista.com

approaches and the state of the art of image retrieval including different types of features.

Chapter 3 describes different interest point detectors — We used some of these detectors in our system.

Chapter 4 explains how texture information can be gathered from digital images and how Gabor functions and a Gabor filter bank — the heart of our indexation algorithm — can be used for this purpose.

Chapter 5 gives an overview about the process of collecting Gabor features on interest points and the ways to interpret the results of this process.

Chapter 6 introduces an image description and methods based on the results of the Gabor filters for indexation purposes: Collections of feature vectors where one vector corresponds to an interest point.

Chapter 7 describes histogram based indexation methods. The histograms are built from the same Gabor filter outputs explained in section 5. Different image descriptions and different distance measures are compared.

Chapter 8 describes the environment used to develop and test the system. Two different databases with different types of images have been used. A short overview of the applications of the system is given.

Chapter 9 shows the performance results of the query algorithms introduced in this document on our test environment. We explain in great detail the methods used to measure query performance.

Chapter 10 draws conclusions and gives some outlook on future research and improvements of the methods introduced.

Chapter 2

State of the Art of Content Based Image Retrieval

The requirements for a content based image retrieval system depend on its application and the wishes of the user. Let's imagine a user who wishes to search an image which resembles her query image. This picture contains a beach of white sand, light blue sea, dark blue sky and a red car in the foreground. She provides her query image to an image indexation system which in our case responds to this query by different pictures showing beaches and skies with the same colours. But actually our user wanted some more photographs of the car model shown in the foreground. What went wrong? The user did not specify which type of similarity she desired for her response images. The first image indexing approaches used global colour information to calculate the similarity of images [33]. Since the dominant colour information in our fictitious example is the background — beach, sea and sky — and these areas have been the same at the results, the system returned these entries. A system responding with the desired results would need to know which parts of the image are relevant and which kind of similarity is desired. Colour based similarity would not return blue cars if the car in the query image is red. On the other hand for some applications colours are a good choice to use.

Since research on image indexation began, lots of different algorithms have been developed to satisfy these different types of queries. These algorithms are based on various kinds of image descriptions and methods to compare them. However, there are some parts which stay the same. We will describe the components of indexation systems and the associated terms in this chapter as well as the different approaches.

2.1 Components of an indexation system

Figure 2.1 shows schematically of the principal components of a content based image retrieval system. Two different processes need to be supported by these sys-

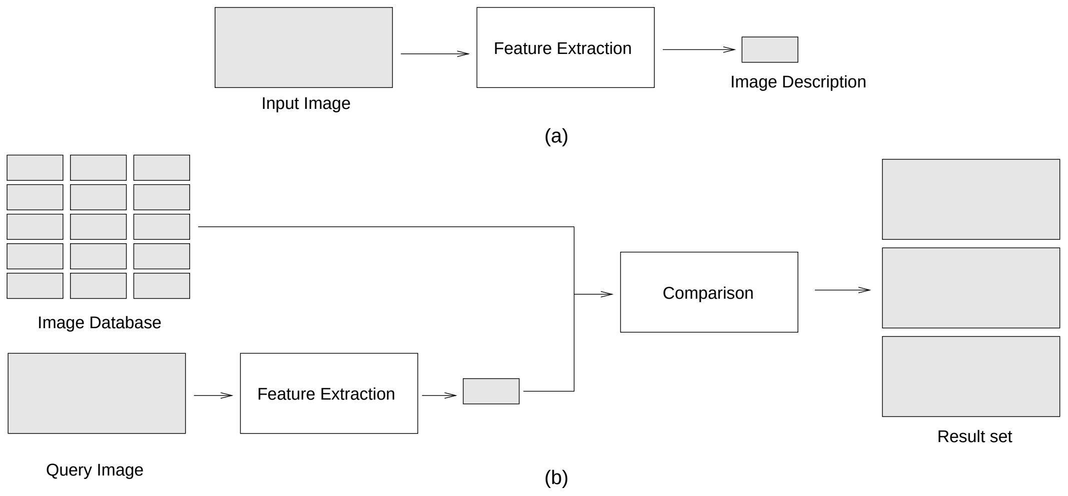


Figure 2.1: Components of an image retrieval system: (a) Building the index (b) Comparing images using the similarity function

tems: Indexing (uploading) images and querying the database. The main function for the user is to query the database providing an example image, which means that this image has to be compared with the images in the database. Since image processing algorithms tend to be computational expensive the comparison is not done using the images themselves but pre-computed image descriptions. The step to create these descriptions is called indexation. For the contents of the database it can be performed off-line and if necessary parallel for different images on different machines, so the performance demands are not as high as for the query step. However, the index for the query image needs to be created on-line during the query process unless the query image is taken from the database itself. The process can be described by three steps:

Pre-processing The first step before creating the image description is to prepare the image data. The operation being performed depends on the type of features taken in the next step. Usual pre-processing steps are applications of filters (e.g. Gaussian, median filter), segmentation of the image into regions of homogeneous colour or texture, or selection of special regions in the image.

Feature extraction The feature extraction step is gathering the actual information needed to describe the image. What kind of information needs to be extracted depends on the type of queries that shall be performed. We will describe different feature types (colour, texture, shape and structure) in this section.

Encoding The encoding step takes the extracted feature information and produces compact coded byte sequences which can be stored in the database.

The result of the indexation process is an image key, i.e. a byte sequence, for each image in the database together with a link to its source image. During the query process the systems uses a distance function to compare these keys with the query image. The distance function (or similarity function) compares two images and returns the distance of the two images in feature space. The database images whose features have minimum distance to the features of the query image are returned to the user as result of the query. It follows from this algorithm, that the image features and the distance function are depending on each other. The following properties of features and distance function are desirable:

Distinction This property means, that descriptions for images with different contents should be different, i.e. their distance should be big, while descriptions of images with similar contents should have a small distance. To satisfy these conditions the features should reflect the user requirements. E.g. if the desired result images should resemble the query image in colour, then the features should also be distinctive in colour. The property of distinction is hard to measure, since there is no general measure of visual similarity.

Compact size The obvious advantage of compact keys are the small space requirements for the database. The difference in disk space for an image database which stores millions of images is considerable, even if the sizes of the single keys can be decreased by small values.

Fast distance calculation During the query process the distance function is called often. Hence a fast calculation of this distance value is essential and determines the query speed. Before comparing two image descriptions they are read from the database and decoded by the query process. Then a data structure is built. The requirements for fast decoding of the stored image descriptions often conflicts with the desire of small key sizes.

2.2 Previous Work

This section describes previous work done in the field of image indexation. Although the algorithms developed in this work are based on texture similarity we will give here an overall overview about indexing algorithms including brief descriptions of other similarity types like colour and structure. More specific information about texture and texture features will be given in chapter 4.

2.2.1 Colour Based Image Retrieval

Research in the development of image features usable for indexation purposes began with the usage of global features. Global features are influenced by the whole image whereas local features are calculated by parts of the image or even a few pixels of it. An early indexing system on colour basis had been introduced by Swain and Ballard [33]. They filled global histograms with the RGB colour values of the input images and used the L_1 and the L_2 histogram distances to compare the histograms (See Section 7.3 on histogram distances). The algorithms were simple but performed surprisingly well. Improvements can be found in [18].

The main disadvantage of this method were the missing spatial constraints. The colour distribution stored in the histograms was calculated from the whole image area. Stricker et al. [32] developed an improvement by including limited spatial coherence. They introduced five partly overlapping, regions - one in the centre and one at each corner of the image, as additional criteria for the matching process. The contribution of each of the regions to the matching process can be adjusted by the user via weighting parameters, so the user himself can stress specific regions of the query image as being important and exclude other regions. Furthermore they filled 3-dimensional histograms with colour distributions with the first three moments (mean colour values, weighted variance and skewness) instead of the raw RGB values.

One step further towards of local features is the solution of Matas et. al [19] — a colour based approach as well. Their image descriptions consists of a color adjacency graph which to build after fully segmenting the image into regions of homogeneous colour. Nodes in this graph represent clusters of chromatic components which are connected by edges if they are spatial neighbours. Searching parts of an image in the database can be implemented by sub graph matching. However, the power of this method — the segmentation — is also a disadvantage. The performance of the query depends on the correct segmentation of the images. If this first process is not robust enough the quality of the response decreases.

2.2.2 Local invariant features

A different approach of feature extraction can be followed by using local features. Unlike global features they are not calculated from the whole image, but from small spatial areas or even single pixels instead. But calculating local features on every pixel of the images creates too much data, so not all pixels usually result in feature information. Instead the features are extracted from previously chosen regions of interest. One of the basic questions is how these regions of interest are selected, and what kind of special properties do they have. One idea is to select special points in the image, called *interest points*, and to extract features on areas around these points. Since we followed this idea in our work as well there is a chapter on interest points (Chapter 3), where different types and implementations of interest operators are discussed.

In [28] Schmidt and Mohr present an image retrieval method based on local features extracted on interest points. They choose the Harris corner detector (see section 3.2) to select key points of the image. For each key point they compute a vector with values invariant to similarity transformations, like the average luminance, the square of the gradient magnitude, the Laplacian etc. These feature vectors characterise the neighborhood of the key points they were collected on. The similarity function necessary to compute the distance values for two feature vectors is the Mahalanobis distance. This distance takes into account the different magnitudes as well as the different second-order statistical distributions of the elements of the feature vectors. Distances between two images are based on the distances of their feature vectors, since images are represented as sets of feature vectors. A voting algorithm decides about the image of the database most similar to the query image (See Section 6.4.1). Performing a query the set of feature vectors is created for the query image, and every vector is compared with the pre-computed features of the database images. If the distance is below a threshold t, then the involved database image gets a vote. The images having maximum votes are returned to the user.

The authors refine the indexing algorithm by adding multi scale representation. The pre-computed feature vectors of the database images are calculated at several scales. This can be achieved by different quantities of smoothing of the Gaussian derivatives which are basis for the computation of the invariants. The feature vectors of the query image are still computed at one scale only, but they are compared to the feature vectors of the database, which are extracted at several scales. This adds invariance to scale to the retrieval algorithm at the cost of a higher possibility of wrong matches. To compensate for that, semi local constraints are introduced. The constraints increase the requirements for a match by demanding not only a match of the two feature vectors but also of 50% of their neighbor vectors. An additional geometric constraint further decreases the possibility of wrong matches. Angles between key points have to remain consistent (e.g. the angles α_1 and α_2 in Figure 2.2).

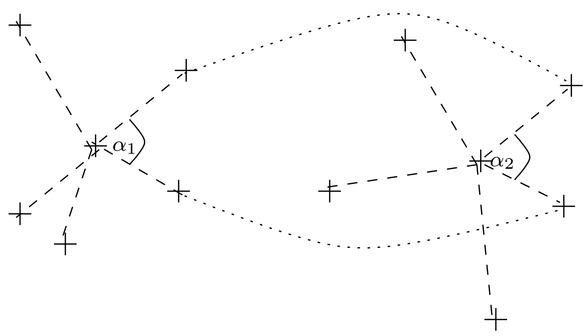


Figure 2.2: Semilocal constraints for the matching conditions of feature vectors

Siggelkow and Burkhard also use local invariant features [29] gathered on what they call key points, which "are distinguished and differ from the other points by some very structured neighbourhood". Basically they use rotation invariant histograms, whose features are built by taking integrals over all possible rotation functions applied to the key point. Remarkable is their method to built the histograms from the features. One feature does not correspond to a single histogram bin, but to a circular area of bins which is centred at the destination bin. Each bin receives a weighted input of this features, where the weights are smaller at the borders of the area. This approach makes histogram comparisons less sensible to small shifts in the pixel values.

For his object recognition algorithm Lowe [17] uses features on interest points as well, applying a method which he calls SIFT (*Scaled Invariant Feature Transform*). The basic idea is similar to the methods for image retrieval based on interest points: An object is recognised in an image if sufficient number of SIFT keys of a region, i.e. the features extracted on these interest points, match the objects keys. The locations are determined by searching maxima and minima of a difference of Gaussian function in an image pyramid. According to the author the operator detects points at regions and scales of high variations.

To each interest point a feature vector (called SIFT key vector) is created. To make the feature data invariant to changes of rotation and scale, canonical orientation and scale values are assigned to each vector. The feature data is stored relative to these values. The canonical scale can be determined by the pyramid level at which the key was detected. The canonical orientation R_{ij} for a pixel A_{ij} is computed by pixel differences $R_{ij} = atan(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij})$. To make the orientation more robust it is computed by using the peak values of a histogram of local orientations.

A SIFT key vector contains gradient values separated into different orientation planes, where each plane contains only gradients which correspond to its orientation. The orientations of the planes are computed relative to the canonical orientation of the vector. The regarded pixels are locations that fall in a circle around the key location. These orientation planes are computed for two levels: The level at which the interest point location was detected and the neighbouring level one octave higher.

According to the author these SIFT keys give enough measurements for high specifity, whereas they are invariant to scale, orientation and small variations in illumination.

2.2.3 Texture Based Algorithms

Rubner and Tomasi present in [25] an image retrieval algorithm based on texture features. By applying a dictionary of Gabor filters to each image they get a cloud of points in a multi dimensional space, where each point belongs to an image location. They post process this distribution using a clustering algorithm. The results are cluster centres plus their corresponding weights, which are inserted into a texture signature. The *Earth Mover's distance* is used to compare the signatures representing an image. This distance has been chosen for its interesting properties like robustness to small shifts in the histogram/signature distribution.

Similar to our approach Bhattacharjee uses in [1] a method based on interest points and texture features collected on regions around these feature points. See section 3.5 for a description of this interest point detector based on Morlet wavelets. An interest region is created around each interest point and a filter bank applied to it. Bhattacharjee uses three different filter types: the first-, second- and third-order derivatives of Gaussians. Each filter type is applied in different directions to the interest region. The maximum response of each filter is picked. This results in an n-dimensional vector for a filter bank of n filters. In analogy to text based retrieval systems he calls these vectors *tokens*. However, instead of comparing these token sets directly, an indexing vocabulary consisting of indexing terms is built. The vector space created by the n-dimensional tokens creates an n dimensional hyper cube. In this hyper cube an n-dimensional grid is created by partitioning the axis into intervals. The set of indexing terms consists of all grid points in this cube.

Each image is represented as a vector of weights based on these indexing terms. The weighting is either binary (1 - the term has been influenced by at least one token of the image, 0 - else) or more sophisticated using the significance of the indexing terms. The significance of an indexing term is controlled by two measures:

- The distance of the tokens to the indexing term. The bigger the distance the more likely the indexing term is a noisy version of the token
- The number of times the term appears in the whole database. A term that appears in many images is not very useful for retrieval

The similarity measure between two images is defined by the inner product of their weights vectors.

2.2.4 Methods Based on Structure

Image databases based on structural similarity take into account geometric descriptions of images, thus are aimed to index primarily drawings or other images showing strong geometric properties. For this reason Huet and Hancock implemented structure features to index databases of cartographic material in [11]. They developed their image descriptions to fit the contents of their database — aerial images of cities. The features were based on line segments. Hence the pre-processing phase includes an edge detection step and a segmentation process to divide the edges of the image into straight line segments. A following step computes attributes from pairs of line segments (Figure 2.3). They use features invariant to scale and rotation by extracting relative measures:

The relative orientation between the lines:

$$\phi_{ab,cd} = \min[(\theta_{ab} - \theta_{cd}), (\theta_{cd} - \theta_{ab})]$$

The ratio of line-segment length:

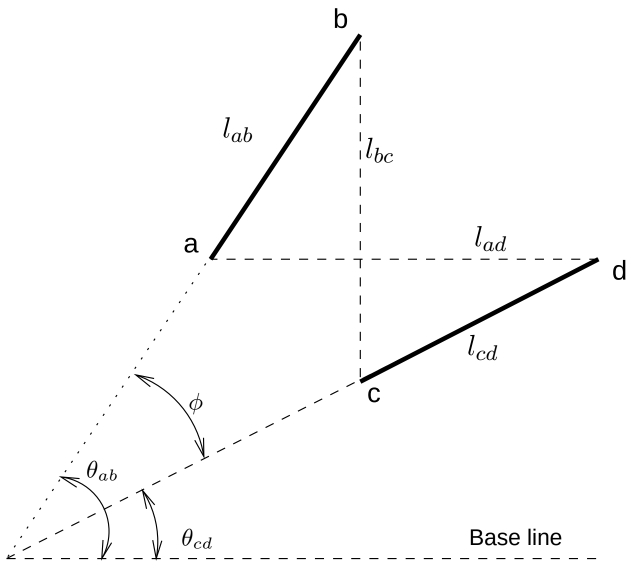


Figure 2.3: Attributes computed from line segments

$$r_{ab,cd} = \frac{\min[l_{ab}, l_{cd}]}{\max[l_{ab}, l_{cd}]}$$

The line-segment projection cross-ratio:

$$xr_{ab,cd} = \frac{\min[l_{ad}, l_{bc}]}{\max[l_{ad}, l_{bc}]}$$

where l_{ab} is the length of the line-segment and θ_{ab} is its orientation. These attributes are used to increase the respective histograms, i.e. different tests using angle histograms, length ratio histograms and cross ratio histograms have been performed. However, experiments using the angle histograms delivered the best results, the line-segment projection cross-ratio was not as discriminative, and the length ratio was not discriminative at all. Huet and Hancock also applied different histogram distances (L_1, L_2, *Bhattacharyya, Matusita, Divergence* — see section 7.3 on histogram distances) and recommended the Bhattacharyya distance as the most suitable for this kind of indexation.

The main disadvantage of this solution is its specialisation to aerial images by segmenting them into straight lines segments, which makes it less usable for other scenes. Popescu adopts in [23] the pairwise histograms of Huet and Hancock to hold interest point locations, thus extending the domain of usage to natural images. Instead of edge detection and segmentation an interest point detection step is applied to pre-process the images. He uses the multi resolution contrast based interest operator of Jolion (See section 3.6 for details on this operator). To create the pairwise histograms only the location of the points is used, no more information is taken from the image. The following steps are performed:

- Application of the interest operator to get the point locations.
- For each interest point calculate the spatially n-nearest neighbouring interest points.
- For each interest point calculate the set T of all possible triples of points, where the first point is the interest point itself, and the remaining two points are neighbors, which are immediately following each other in distance ranking:
$$T = \{(P, N_i, N_{i+1}) \in N \mid i \in [1, n]\}$$
where N is the set of the spatially n-nearest neighbouring points and the index i of the N_i denotes the ranking of the neighbouring points regarding the original point P, i.e. the nearest neighbour is specified by N_1, the second-nearest by N_2 etc.
- For each triple of T increase a histogram bin of the two dimensional feature histogram. The bin is determined by the neighborhood ranking of the neighbor points (first coordinate) and the angle between lines connecting the interest point with the neighbors.

Figure 2.4 shows a configuration of points which leads to an increment of the feature histogram. The nearest neighbor search for the interest point P finds the locations of the neighbors N_1, N_2 and N_3. The set of tripels can be given with $T = \{(P, N_1, N_2), (P, N_2, N_3)\}$. In our example the triple (P, N_2, N_3) is examined: The properties to determine the histogram bin which needs to be increased are the neighborhood ranking of the neighbor $N_2 = 2$ and the angle α between the two lines connecting the point P with its neighbors N_2 and N_3.

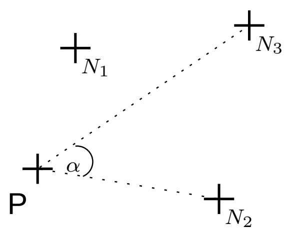


Figure 2.4: Indexing interest point locations

The two dimensional histograms created in the indexation step are compared by calculating the Bhattacharya distance during the query phase. Results on a database of images of various kinds proved, that the algorithms gives good results on drawings as well as on natural images.

2.2.5 Shape based methods

Mokhtarian et al. present in [21] an algorithm retrieving images through shape features. The basic principle of their method is the representation of images as curves. For this reason the images in their test database contain only one object per image. The representation is based on the curvature scale space theory: A closed planar curve can be given as

$$\Gamma = \{(x(u), y(u)) | u \in [0, 1]\}$$

where u is the normalised arc length parameter and $x(u)$ and $y(u)$ are the coordinate functions. The starting point of the curve is chosen randomly. By applying a one-dimensional Gaussian smoothing filter to the coordinate functions, the number curvature zero crossings decreases with increasing width σ of the Gaussian kernel, until the curve becomes convex and there are no more curvature zero crossings. This process can be modelled in a curvature scale space image (CSS image) as shown in Figure 2.5. The x-axes shows the arc length parameter u and the y-axis shows the width σ of the Gaussian kernel. The curves in the CSS image show the progress of concavities and convexities in the image. The original curve is represented by the maximas of the curves in the CSS image.

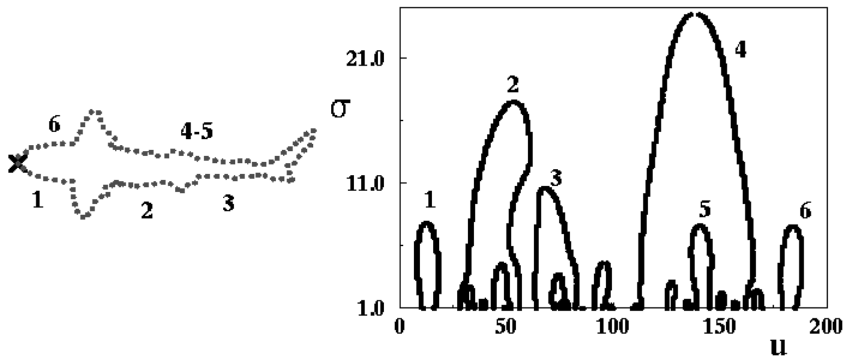


Figure 2.5: The CSS image of an example curve

The distance function compares the maxima locations of the two CSS images. Since the starting point of the curves has been chosen randomly, the curves have to be matched first. This can be done by choosing one point of each CSS image and shifting one image so that these two points match. This is done with several times and the best match is taken. The final distance is the sum of the straight line distances of the matched maxima pairs plus the y coordinates of the unmatched maximas.

The performance of the system has been evaluated on database of marine animals. The results sets of the query algorithm have been compared with the results sets created by different humans evaluators.

2.2.6 Commercial Systems

Content based image retrieval is a young domain, but there are already some successful commercial systems. One of the first presented is the system QBIC developed by the company IBM [6]. This query application includes a graphical query tool which not only allows to use example images to query the system, but also to create a query specification by drawing, sketching and selecting colours etc.

The query methods support colour, texture and shape features. The user may choose the type of query or the weights for the similarity measures. The features are global. For colour queries 3-dimensional colour histograms are taken. Texture oriented queries compare coarseness, contrast and directionally features and queries measuring shape similarity compare measures of area, circularity, eccentricity, major-axis direction etc.

In order to decrease query time several indexation techniques are applied. Fast filtering techniques eliminate a large number of candidates. For low dimensional features as 3D-histograms the database images are indexed using conventional index technologies known from database theory. Higher dimensional features are reduced to lower dimensions using the principal component transform.

A similar application has been developed by the company Virage, Inc.[9]: The system VIR (*Visual Information Retrieval*) also powers the content based image retrieval search function of the web search engine altavista[1]. Like QBIC, VIR supports different feature types like colour, texture, structure and shape. The features are either calculated globally or locally for smaller regions of the image.

The Virage core image indexing engine includes a programmers interface which makes it possible to invoke the functionality from different applications. The engine operates stateless and performs operations like the creation of feature vectors for images and the comparison. The applications are responsible for the storage of the images and the feature data. The system also supports query refinement by re-calculating the weights for the query methods according to the desires of the user (See section 2.3).

2.2.7 Summary

A summary of the methods for content based image query classified by feature type can be found in Table 2.1.

2.3 Environment of this work

The algorithms and methods described in this work have been developed at the Laboratoire Reconnaissance de Formes et Vision (*RFV*) of the Institute National de Science Appliquee de Lyon and the Pattern Recognition and Image Processing

[1] http://www.altavista.com

Author(s)	Feature Type	Description
Swain and Ballard [33]	Colour	Colour histograms
Stricker et al. [32]	Colour	Colour histograms for fixed regions
Matas et. al [19]	Colour	Color adjacency graph
Schmidt and Mohr [28]	Invariant features	Sets of feature vectors with invariant features. A voting algorithm calculates the distance
Siggelkow and Burkhard [29]	Invariant features	Histograms with invariant features
Lowe [17]	Invariant features	Object recognition using feature vectors with invariant features.
Rubner and Tomasi [25]	Texture	Gabor filters and Histograms
Bhattacharjee [1]	Texture	Texture features on interest regions. The features are distances of texture tokens to indexing terms
Huet and Hancock [11]	Structure	Pairwise histograms filled with geometric attributes of line segments
Popescu [23]	Structure	Pairwise histograms filled with geometric attributes taken from the locations of interest points
Mokhtarian et al. [21]	Shape	Features taken from the curvature scale space
IBM	Colour, texture, shape	The system uses global colour histograms, global texture features and global shape features.
Virage	Colour, texture, shape	Feature are the dominant colour and the colour variation, variations in colour and global shape features.

Table 2.1: A summary of the methods described in this chapter

Group (*PRIP*) at Vienna University of Technology. The aim was to create a content based image retrieval system based on texture similarity, which should — after further development — be part of an indexation algorithm based on user feedback, which is under development at RFV. Figure 2.6 shows the scheme of a query system including user feedback. The process starts with the first query, which presents its results to the user, who chooses the images corresponding the most to her wishes as well as the ones having the least (subjective) similarity. Based on this subset of the results the system adjusts its parameters and re-runs the query, thus achieving a refinement of the original query results. The essential requirement to ensure the increasing quality of the query results is to translate the user's digest of the results into the right adjustments of the system parameters.

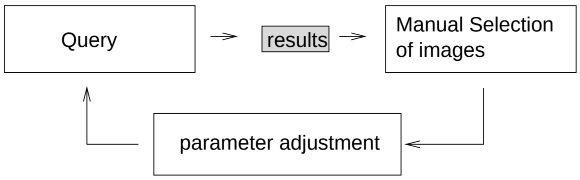


Figure 2.6: Image retrieval system with user feedback and parameter adjustment

A variant of this user feedback method is the multi feature retrieval system developed at RFV, where the adjustable parameters are weights for different query engines. Figure 2.7 shows a schematic overview of this system. The basic components are query engines based on colour, texture, structure and shape similarity. Starting a query each of these engines retrieves a result set of images out of the common database, which includes images of maximum similarity to the query images according to its respective similarity measure. All result sets are combined to one final result set using individual weights for each query method. These initial weights could be set to default values gathered from empiric studies of visual perception or they could simply be historic values preferred by the users.

The combined results of this first stage are presented to the user, who chooses her preferred images. According to this subset of the results the weights for the query engines are adjusted. This is done by checking which similarity types have been favoured by the user. Query engines whose images have been chosen preferably are rewarded by increased weights, whereas engines whose images did not manage to get in the users digest are punished by decreased weights. This way the image database adopts to the users wishes and delivers images according to the similarity desired by the user. One drawback of this method is that these weights differ from image to image. In one query texture is important, in another colour etc.

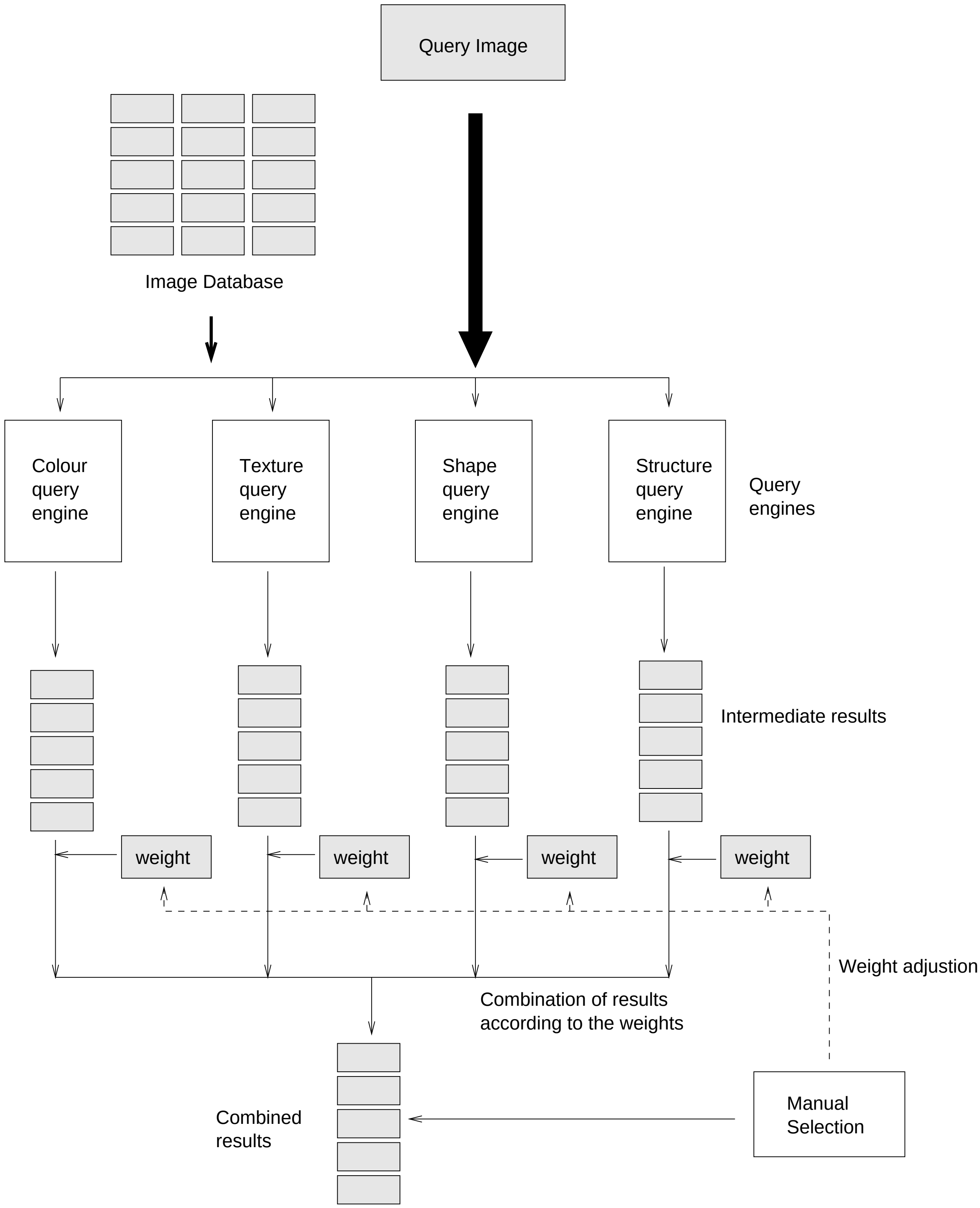


Figure 2.7: Multi feature retrieval system with user feed back and weight adjustment

Chapter 3

Interest Points

Image processing is a domain which depends on high computational power. Computers are getting more powerful, but still the task of storing and comparing large amounts of image and especially video data needs high capacities of both memory and CPU. Today a typical image contains 512×512 pixels, i.e. an indexation algorithm has to describe features for 260,000 pixels per image. Performance problems are not the only issue resulting from this large quantities of information. Most of this information is redundant. Pixels of homogeneous regions contain similar information, so adding them to an image description will not increase its quality considerably. In the area of image indexation much research was done to decrease redundancy and the amount of information necessary to describe images. Different attempts have been made:

Global features Global image features like colour histograms [33] describe the whole image only. However, the costs are rather high. Global features are sensible to outliers and matching only parts of images is not possible.

Segmentation Some algorithms perform a segmentation step in a pre-processing phase [19] to segment the image into regions which are homogeneous in the desired property (e.g. colour or texture). Features are extracted on each component.

Detection of interest points Instead of performing a full segmentation an interest operator detects points with specific properties. Features are extracted on these points or on regions around these points.

In our approach we decided to use an interest operator to locate key points in the input images. These points of special interest should hold the main information of an image both using their location and texture information gathered on their local surrounding region. This chapter will describe requirements for interest detectors, their historical development, and will explain some types of operators putting the emphasis on those we used in this work.

3.1 Overview

The first interest point detectors have been developed for 3D vision and robotics [22],[10] to extract corners. They have been used — and are still used — for motion detection or tracking systems, where it is necessary to have stabil features which remain unchanged in a sequence of video frames. The definition which points in an image are corners differs. Moravec and Harris defined corners as points, where a shift of rectangular window in all 4 directions changes the intensities of this window significantly (See section 3.2). Smith and Brady detect corners by applying a circular mask to each pixel and evaluating the differnces of the grayvalues in this mask (See section 3.3). But in all cases corner detectors rely on information of geometric nature whereas not all image features used for image indexation are geometric. Because the requirements for interest operators are not the same as those for corner detectors, more recently operators have been developed explicitly for indexation purposes. We will try to state the most common requirements for interest operators:

Stability An interest operator determines the locations in the image where features are extracted to create the image descriptions. Therefore it is important that for two similar images the interest points are detected at similar locations. Especially for algorithms using weak features where the location itself is used for indexation this property is of uttermost importance.

Descriptiveness The interest points need to be located at areas where the gathered features are most descriptive, i.e. the descriptiveness of a point location is dependent on the type of features used. However, most interest point operators define this descriptiveness as the amount of *"signal variation"* that can be found at that location. These interest points are also sometimes called *"salient points"*.

Invariance to rotation and scale Most image similarity measures demand robustness to changes of rotation and scale, which also requires the interest operator to have this property.

Robustness to JPEG and MPEG codation Images coded in the JPEG format or extracted from MPEG coded video streams suffer from artifacts due to information loss in the compression step. Especially at high compression rates the quality of the images decreases fast. The JPEG algorithm compresses images in blocks of 8×8 pixels using the discrete cosine transformation (DCT). At high compression rates, where less DCT coefficients are used for reconstruction, neighbouring blocks do not fit together, producing square shaped artifacts. These squares do not only disturb the visual impression of the image, they also pose problems to image processing algorithms, thus also corner and interest point detectors. However, since already large quantities of available image and video material are stored in the JPEG and MPEG formats an interest operator is required to be robust against the types of artifacts they produce.

3.2 The Harris Corner Detector

The corner detector described by Harris and Stephens [10] is also known as *Plessey corner detector*. The algorithm is based on the detector of Moravec [22], which uses a small window and the changes in image intensity when shifting the window in different directions. Moravec describes the change E as:

$$E_{x,y} = \sum_{u,v} w_{u,v} \mid I_{x+u,y+v} - I_{u,v} \mid^2$$

Depending on the grey-level distribution of the image under the window the following cases are possible:

1. If the image intensities are almost constant, then the changes after shifting the windows will be small
2. If the window is crossing an edge, then a shift parallel to the edge will cause small changes, whereas a shift perpendicular to the edge will cause a large change
3. If the window is crossing a corner, then shifts in every direction will cause a significant change in image intensity

Harris and Stephens proposed some improvements to Moravecs detector. First a Taylor expansion about the shift origin was performed:

$$E_{x,y} = \sum_{u,v} w_{u,v} \mid x\frac{\partial I}{\partial x} + y\frac{\partial I}{\partial y} + O(x^2, y^2) \mid^2$$

$$E_{x,y} = Ax^2 + 2Cxy + By^2$$

where

$$A = \frac{\partial I}{\partial x} \otimes w$$

$$B = \frac{\partial I}{\partial y} \otimes w$$

$$C = \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial y} \otimes w$$

Furthermore they changed the rectangular window for a Gaussian window:

$$w_{u,v} = \frac{1}{\sigma^2} e^{-(u^2+v^2)/2}$$

The change E can be rewritten as

$$E_{x,y} = (x, y)\, M\, (x, y)^T$$

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

On edge pixels one of the two eigenvalues α and β of M will be large and the other one stay small, whereas on corner pixels both eigenvalues will be large. Hence, using α and β a rotationally invariant response function can be created:

$$R = Det(M) \,-\, k\,(\alpha + \beta)^2$$

The actual interest points are created by a non local maxima suppression of this function.

The behaviour and the performance of the detector thus depends on 5 parameters. In this work we use two different implementations of the Harris operator with different parameters. The first version (referred to as *Harris 1* from now on) was implemented by Nicu Sebe [1], the second one (referred to as *Harris 2*) by Jean-Michel Jolion [2]. The parameters are the following:

- The parameter k of the response function. This parameter is the same for both versions: $k = 0.04$.
- The variance σ of the Gaussian function. Harris 1 uses a filter kernel of [1 2 1], whereas Harris 2 uses a kernel of [1 3 3 1].
- The kernel of the derivative function. The kernel for Harris 1 is [-1 8 0 -8 1]. Harris 2 uses a recursive derivative function (Deriche).
- The size of the window of the algorithm to extract the local maxima. Harris 1 uses a window of 3×3, Harris 2 uses 5×5.
- A final threshold to get the desired number of points. This value can be chosen by the user.

[1] Leiden Imaging and Multimedia Group

[2] Laboratoire Reconnaissance des Formes et Vision, INSA Lyon

3.3 SUSAN

The SUSAN corner detector is part of the low level image processing approach introduced by Smith and Brady [30] which combines edge detection, corner detection and noise reduction using the same basic principle. The basic feature of the SUSAN algorithms is a circular mask applied to each pixel of the image. The pixel in the center of the mask is called the nucleus. All other pixels can be classified into two classes: Pixels having the same (or similar) gray value as the nucleus and pixels having a different gray value. The area of pixels having the same gray value as the nucleus is called USAN (*"Univalue Segment Assimilating Nucleus"*).

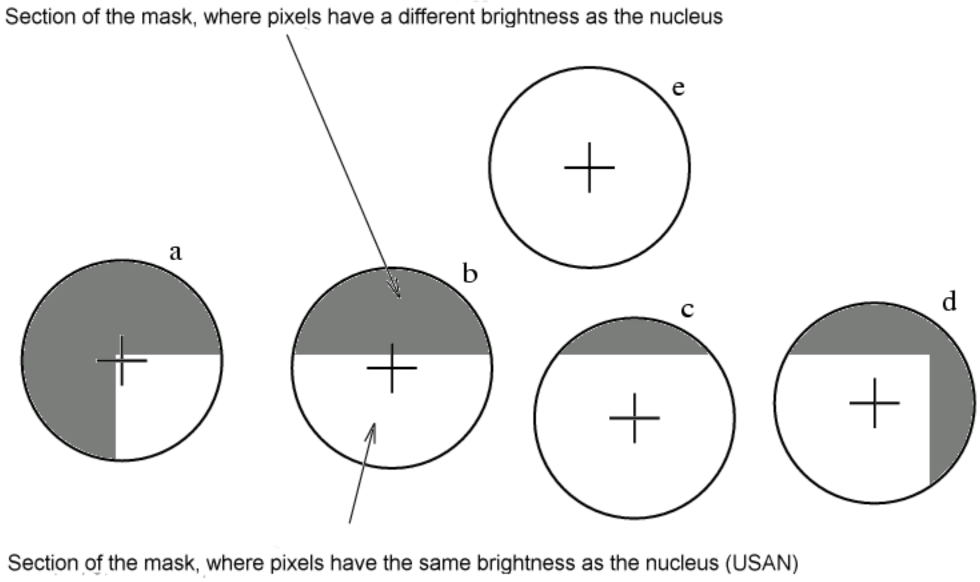


Figure 3.1: The principle of the SUSAN corner detector

Figure 3.1 shows 5 masks applied on different locations to an image containing a white rectangle on a dark gray background. As can be seen in the figure, the size of the USAN relative to the size of the whole mask contains information about the structure of the image. If the mask is placed on a homogeneous area then the USAN takes all the area of the mask (Figure 3.1.e). Approaching edges the size of the USAN decreases until it only covers 50% (Figure 3.1.b), whereas on corners it decreases even further (Figure 3.1.a). The SUSAN corner detection algorithm takes advantage of this principle. The following steps are necessary to compute the corner response of a pixel:

- Application of the circular mask and determination of the gray value of the nucleus (= the pixel whose corner response shall be computed).

- Determination of the pixels having the same or similar gray values as the nucleus. But instead of thresholding the gray value difference, the following function is used:

$$c(\vec{r}, \vec{r_0}) = e^{-(\frac{I(\vec{r})-I(\vec{r_0})}{t})^6}$$

  where $I(\vec{r})$ is the gray value of the nucleus and $I(\vec{r_0})$ is the gray value of the point to process. The function evaluates to 1 for small differences, i.e. pixels whose gray values are similar to the nucleus, 0 for big differences, and is smoothly descending near the "threshold value" t. The size of the USAN area on pixel r_0, can be calculated by the sum for all pixels in the mask:

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0})$$

- Threshold the size of USAN and subtract the size from the threshold value g to get the corner response:

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}) & \text{if } n(\vec{r_0}) < g \\ 0 & \text{otherwise} \end{cases}$$

  where $R(\vec{r_0})$ is the corner response and g is the geometric threshold, which determines which types of corners shall be detected. The lower this value, the sharper a corner has to be in order to be detected by the algorithm.

- Detection of false positives is necessary in certain cases, e.g. when the nucleus is set on stripes of thin lines. The size of the USAN will be small and eventually below the threshold g because of the small stripe of gray values similar to nucleus. However, the corner response is not justified in this case. To recognize situations like this the centroid of the USAN is computed and compared with the position of the nucleus. In our example they will be almost identical whereas in the case of a real corner the positions will be quite different. An additional constraint is added to the algorithm: Corner responses are removed, if not all pixels on the line between the nucleus and the centroid of the USAN belong to the USAN. This constraint removes false positives in images with lots of noise.

- A final step of non local maxima suppression delivers the local maxima as corners.

The algorithm is controlled by two parameters: The "quality" parameter g, which controls the shape of the corners, and the "quantity" parameter t, which affects the number of corners the system delivers.

3.4 Interest Point Extraction Using Haar Wavelets

The interest point detector of Loupias et al. [16] is based on the non standard decomposition of Haar and Daubechie wavelet transforms. Primers to wavelets can be found in [31], [8]. Generally Wavelets are used to hierarchically decompose functions. Similar to the windowed Fourier transform, wavelets allow to analyse functions in the frequency domain. The Fourier analysis transforms a periodic function into its frequency representation by representing it as sums of sine and cosine functions. For each frequency a coefficient is calculated, the "energy" of this frequency in the signal. However, the location of this energy cannot be found easily. The windowed Fourier transform is used for non periodic functions. The signal is cut into windows of fixed size, and the Fourier transform applied to each window. This allows to localise the frequency coefficients more or less precisely according to the window size.

One of the biggest disadvantages of the windowed Fourier transform is the fixed size of its windows. The perfect windows size is dependent on the frequencies which shall be localised. Low frequencies need bigger windows. If the window is too small, the period of the signal is not covered by the windows. High frequencies need smaller windows. If the window size is too big the localisation of the signal is not precise enough. Wavelets use windows of variable size. The basis functions are derived from a mother wavelet function. The basis functions for high frequencies are short compressed versions of the mother function, whereas the basis functions for the low frequencies are long, stretched versions of the mother wavelet.

Applied to image processing wavelets are used to represent images at different resolutions with different levels of details. A special type of wavelets are Gabor wavelets, which use Gabor functions as mother wavelet function. In our work we use a bank of filters built from these wavelet functions for texture feature extraction. In section 4.2 we give further details on Gabor functions and filters.

The interest point detector of Loupias uses the Haar and the Daubechie wavelet. The perhaps most simple wavelet is the Haar Wavelet, whose functionality will be explained in the next section.

3.4.1 The Haar Transform

As an example the gray values of an image are given in Figure 3.2.

1	7	6	2
9	7	1	3
4	5	6	2
3	4	9	3

Figure 3.2: Example image for a Haar wavelet

We are calculating the second level of a multi resolution representation by computing the average values of the pixels, which gives us an image half the size of the original image (Figure 3.3).

6	3
4	5

Figure 3.3: Average values after filter application

However, information is lost on this step. To be able to reconstruct the original image, the differences of the average values to the original values have to be calculated and stored as well. This results in a matrix of the original size, with the average values in the upper left quadrant and the detail coefficients in the other quadrants (Figure 3.4).

6	3	-4	2
4	5	-2	10
-8	4	-8	6
2	-4	0	-2

Figure 3.4: The Haar wavelet after computation of the second level

This process can be applied recursively to the upper left quadrant containing the average values. An example of this decomposition applied to the example image *cameraman* (figure 3.6a) can be seen in figure 3.6b. Note, that the gray values in the three other quadrants contain differential values of different filters (Figure 3.5). Hence the Haar transform is a multi resolution representation of the frequency coefficients of an image. High coefficients correspond to high variations in the image, thus salient points. These points can be extracted on different levels of the wavelet.

+	+	+	-
+	+	+	-
+	+	+	-
-	-	-	+

Figure 3.5: The filters applied on each level

Many other wavelet functions are suitable for this algorithm. The second form of the Loupias operator is based on the Daubechie wavelet[16], which is basically a more general version of the Haar wavelet. Loupias used for his interest point detector a filter size of 4×4. The coefficients of the filter are derived from the

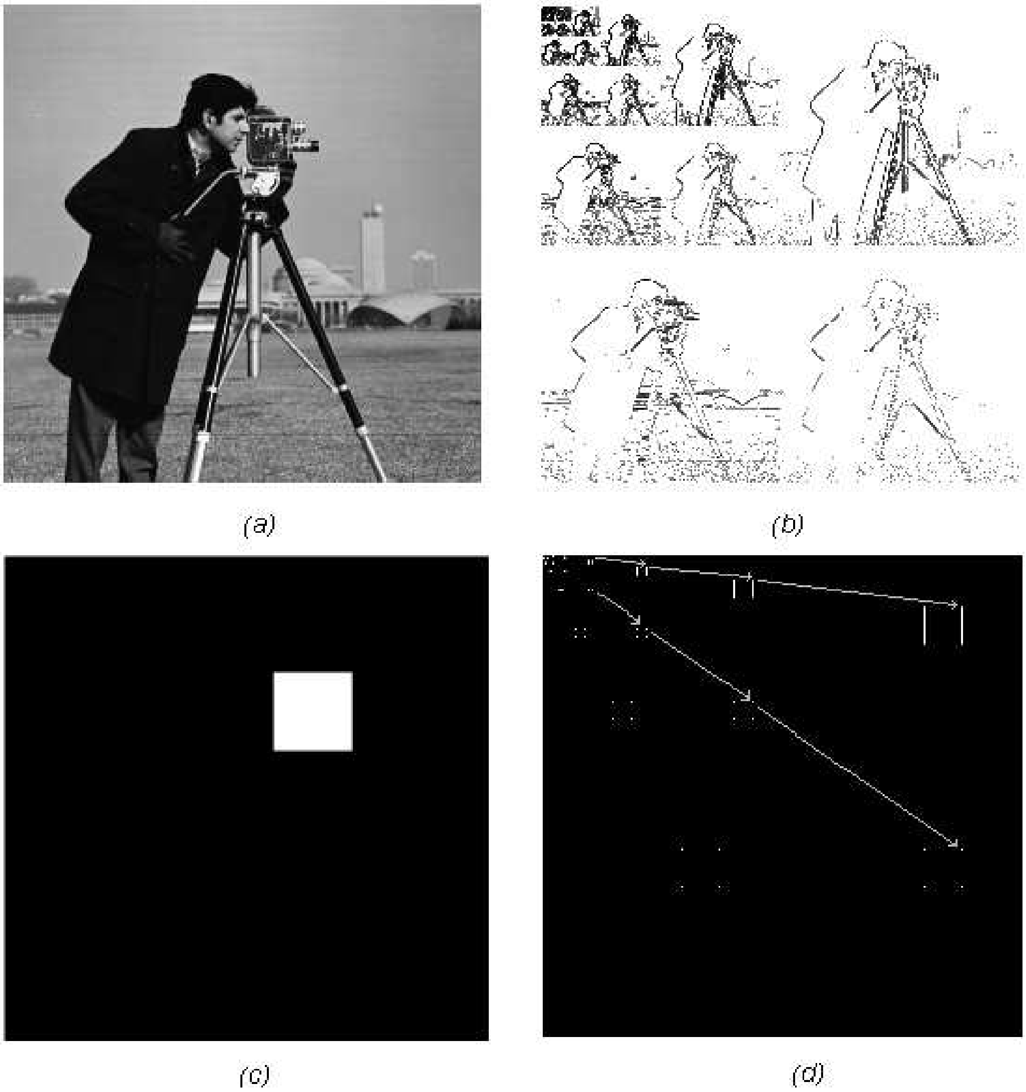


Figure 3.6: The Haar wavelet based interest operator(a) Example picture "cameraman" (b) The coefficients after Haar wavelet transform for example "cameraman"(c) Example picture "square" (d) Tracing an interest point trough the different levels for example "square"

constraint of orthogonality of the basis functions. Like with the Haar wavelet the size of a higher level is half the the size of the parent level. However, Since the mask size is 4×4, the masks are overlapping.

3.4.2 Extraction of the Interest Points

Each point in the transformed image corresponds to a saliency on it's respective level. If we go down one level in the hierarchy this point corresponds to 4 points in the lower level, i.e. these 4 child points have been used to calculate the original value. Of these 4 points the one with the highest coefficient can be seen as the one contributing the most to the value of the upper level, i.e. we can consider this point as the "origin" of the saliency.

The basic idea is to trace the saliency from level to level until we reach the lowest level. Each level of the pyramid contains the applications of the 3 differential filters in quadrants 2, 3 and 4 (Figure 3.5). The process starts at all levels of the transformed image and for all 3 different filter directions. For each point in the current level the maximum coefficients are traced down until the lowest level is reached. To each path the sum of it's coefficients can be assigned, which gives the value of it's saliency. Figure 3.6d shows an example of such a trace for the simple image of a square. Paths are searched starting on every level of the image. However, paths started from lower levels can be parts of paths started from higher levels. In this case the longest route will be taken. Finally, the locations of all traced points in the lowest level are returned to the user, including their assigned values. A threshold can be applied to return the desired number of interest points.

The results obtained by these detectors are different from the results of Harris and Susan, which assume that key points are corners. The points extracted by the wavelets detector are not corners, they are points were "something" happens at various levels of the image. They correspond to areas of high variations in the image.

3.5 Interest Points based on Morlet Wavelets

Bhattacharjee describes in [1] a solution for image retrieval based on structure using interest points and local texture features. His interest point detector is based on a continuous Morlet wavelet, whose mother wavelet function in frequency domain is defined by

$$\widehat{ES}_1(u,v) = \pi i u e^{-(\frac{u^2+v^2}{2})} e^{-(\frac{u^2+(v-y_1)^2}{2})}$$

The algorithm to extract the interest points works as follows:

1. Filter the image with the ES_1 wavelet at 18 different orientations. Only one resolution is used. The image is filtered with a "suitable" scaled version of

the mother wavelet.

2. For each pixel store the maximum of all filter responses in a *maxima image*.
3. Search local maxima. The local maxima are returned as feature points.

According to the author the interest points are usually placed on the end points of linear structures, similar to the response of *end-stopped cells* found in the mammalian cortex.

3.6 Contrast Based Interest Points

Jolion and Bres introduced the multi resolution contrast based detection of interest points in [2]. Like the Haar wavelet approach of Loupias this detector uses a multi resolution representation of an image to calculate the key points on different levels. The method is based on a contrast pyramid.

3.6.1 The Contrast Pyramid

The contrast pyramid built from an image is defined by

$$C_k(P) \equiv \frac{G_k(P)}{B_k(P)} \quad for \quad 0 \leq k \leq N-1 \quad and \quad C_N(P) \equiv 1$$

where $G(P)$ is an intensity pyramid and $B(K)$ is a background pyramid. N is the size of the pyramid given an image of the size $2^N \times 2^N$ pixels. The intensity pyramid is modelled by

$$G_k(P) = \sum_{M \in sons(P)} w(M) \cdot G_{k-1}(M)$$

The background pyramid is defined by

$$B_k(P) = \sum_{Q \in fathers(P)} W(M) \cdot G_{k+1}(Q) \qquad = Expand[G_{k+1}](P)$$

Hence, the background is calculated using the luminance of the father pixels in the upper level of the pyramid. A weight function $W(M)$, which takes into account the way the pyramid is built, is applied to the sum of the luminance of the father pixels. The ratio of the contrast is changed slightly from $\frac{G_k(P)}{B_k(P)}$ to equation 3.1 in order to get a symmetrical contrast measure which is not dependent on low or high intensity situations:

$$C_k^*(P) = min\left(\frac{|G_k(P) - B_k(P)|}{B_k(P)}, \frac{|G_k(P) - B_k(P)|}{255 - B_k(P)}\right) \tag{3.1}$$

An example for the contrast pyramid defined above is depicted in Figure 3.7b.

3.6.2 Extraction of the Interest Points

First on each level the local maxima are extracted and summed up, resulting in a new pyramid:

$$C_k^{\sim}(P) = \begin{cases} C_k^*(P) & \text{if P is a local maximum} \\ 0 & \text{otherwise} \end{cases}$$

This creates a pyramid with interest points extracted on every level. To collapse this pyramid into an image the following top down scheme is applied:

$$E_k^{\sim}(P) = \frac{C_k^{\sim}(P)}{k+1} + Expand[E_{k+1}^{\sim}](P) \quad for \quad k = N-1, \ldots, 0 \quad and \quad E_N^{\sim} = 0$$

The result is an energy map with higher values for key points with high contrast. An example for this energy map is shown in Figure 3.7c. To extract the desired number of interest points a process of non maxima suppression and thresholding is necessary.

Like the Haar wavelet detector, the multi resolution contrast approach does not return corners, since it is not based on any *a priori* model. According to the authors the detected interest points are less sensitive to discretisation and to jpeg coding noise than the Harris and the Susan detectors. Currently research is done using this interest operator for motion analysis in video indexing applications [4].

3.7 Other operators

Zitova et al. propose in [35] an operator which they call *feature point detector*. Feature points are points which belong to crossings of two edges of a specific angle. This property is calculated by counting the sign changes traversing a circle around each point, where this circle holds the differences of the gray value to a local mean. According to the authors the principal property of their operator is rotation invariance and robustness to blurred images.

3.8 Comparison and Remarks

In our image indexation algorithm described in this document we also use interest point detectors to determine interest regions. However, the methods described in this work do not rely on a specific detector. Instead we performed experiments combining our texture feature descriptions with 4 different interest operators, which from now on will be referred to as *Harris* (The Harris corner detector), *Haar* (The Haar-wavelet based operator of Loupias), *Daubechie* (The Daubechie-wavelet based operator of Loupias), *Contrast* (The multi-resolution contrast based interest points of Jolion). For the Harris corner detector both implementations of Sebe and Jolion (section 3.2) have been compared.

Figure 3.8 shows a comparison of these interest operators applied to two different test images. One is an artificial image, the other one the well known natural image *"Lenna"*. A difference which is visible at the first sight is the performance of the Harris operator compared to the other detectors. Looking at the artificial image it is immediately clear that the Harris operator detects corners. In Figure 3.8.a 300 points are detected. This setting is too high for this image, i.e. the algorithms have to detect more points than there are corners in the image. However, the Harris operator is designed to detect corners, not edges. Unlike the other interest point detectors, which also deliver points on edges (E.g. on the border of the rectangle), the Harris operator needs to have high gradients in at least two directions. Harris sometimes returns points on uniform areas. On the other hand the salient point detectors (Harris, Daubechie and Jolion) almost never return points on rather uniform areas. If the number of points required rises they tend to detect points on edges instead.

A look at the results on the image "Lenna" confirms these observations. The points detected by the Harris operator are on corners (windows, eyes, mouth, the crossings of hair etc.) and curves. The results of the wavelet based operators are different. The distribution of the points is more spread out, i.e. there are more points on the background of the image.

We believe that a performance evaluation of these different interest operators can only be performed for a specific application. For content based retrieval systems there is a strong interplay between interest operator, features and indexing methods. For our indexing algorithm based on texture features we performed experiments using all 4 operators. Details and results can be found in chapter 9.

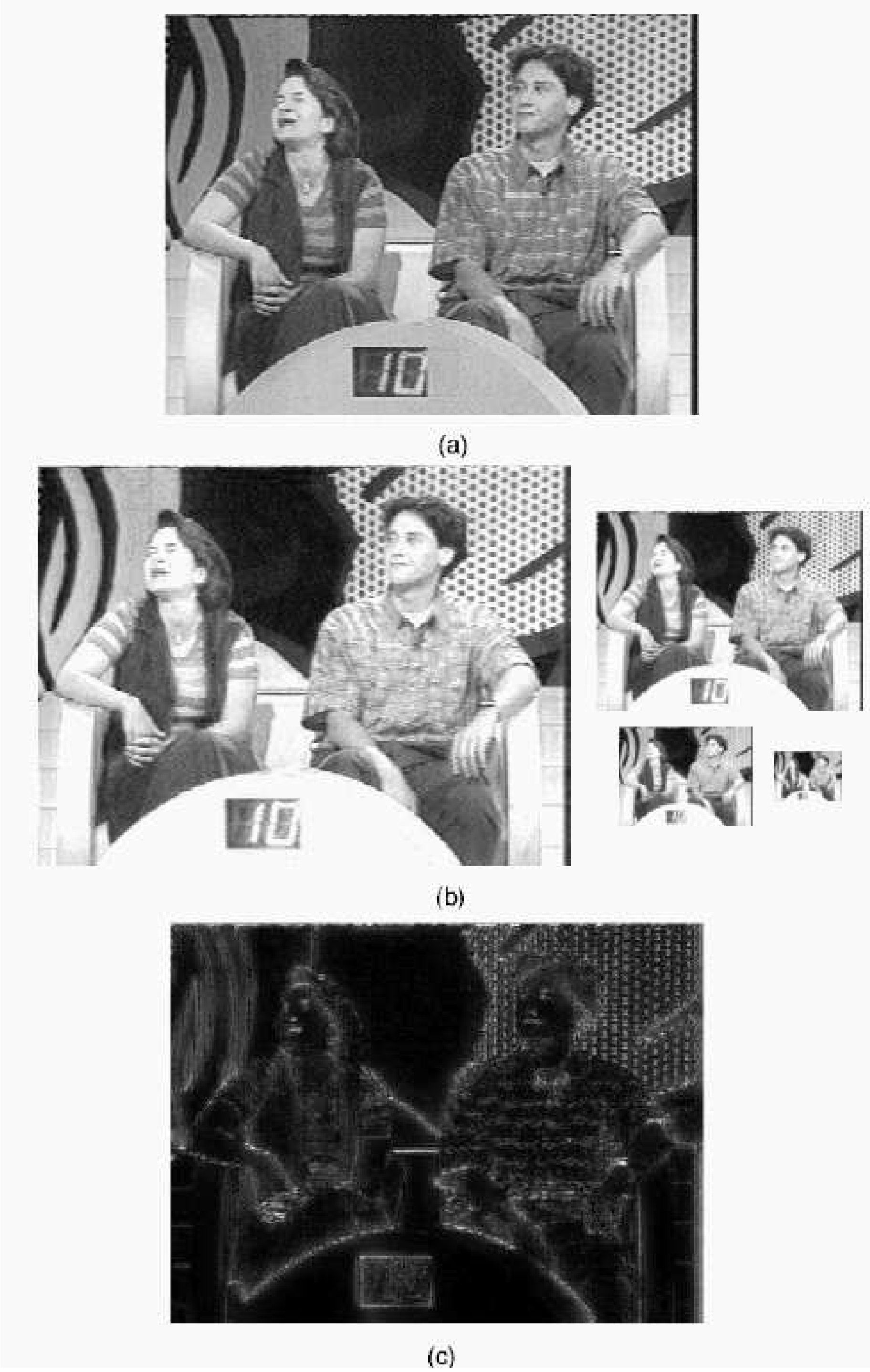


Figure 3.7: The multiresolution contrast based interest points (a) input image (b) The contrast pyramide (c) The collapsed energy map.

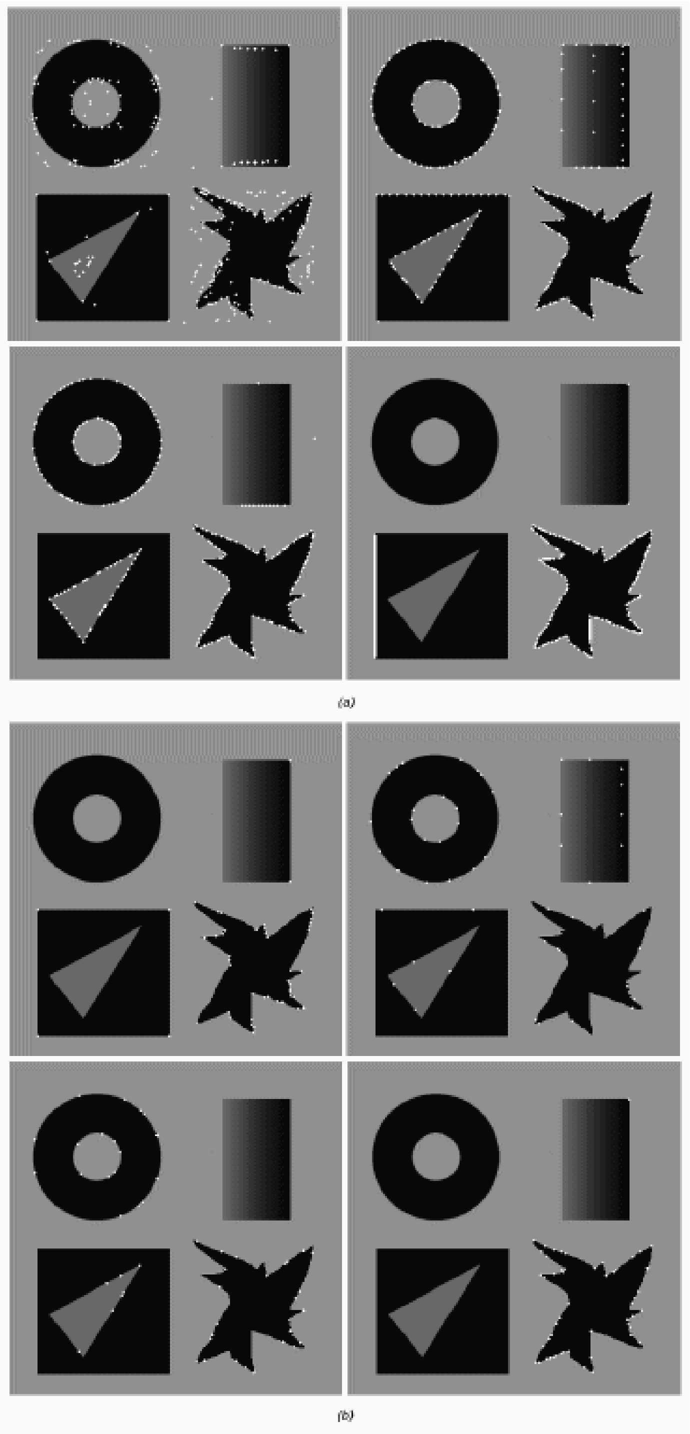


Figure 3.8: Comparison of the interest point detectors applied to an artificial image: Harris (upper left), Haar-wavelet (upper right), Daubechie-wavelet (lower left) and multi resolution contrast based points (lower right) (a) 300 points detected (b) 50 points detected.

Figure 3.9: Comparison of the interest point detectors applied to the Lenna test image: Harris (upper left), Haar-wavelet (upper right), Daubechie-wavelet (lower left) and multi resolution contrast based points (lower right) (a) 300 points detected (b) 50 points detected.

Chapter 4

Texture Features

The similarity measure described in this work is based on texture features. Textures are especially important for natural images, let's think about images of wood, grass, trees, sand, marbles, but also of metallic objects. Unlike colour, texture is not a property which can be extracted from a point, but of areas instead. A single pixel does not have a texture. A common definition is the repetition of basic texture elements, where the type of this repetition differs from case to case. Real periodic textures can be found in artificial images only, whereas natural textures are quasi periodic or random in nature. Figure 4.1 shows some examples for textures.

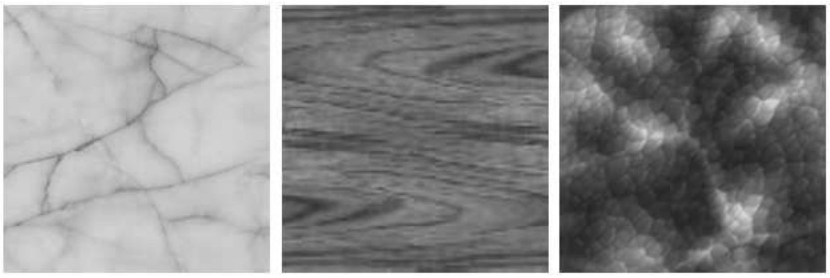


Figure 4.1: Examples for texture

To be able to describe or to classify textures we need to define their properties. Commonly used properties are frequency, direction, phase etc. However, these properties are dependent on the scale the image analysed. A nearly constant area could be textured with a big constant texture or a very fine (infinite) texture. A pullover could show textures on two scales: The printed pattern of squares at a higher scale and the structure of the wool at a lower scale.

Several techniques have been used during the past time for texture description and classification. Statistical methods like autocorrelation functions, co-occurrence matrices, gray level run length statistics have been presented as well as methods of the signal processing domain, e.g. filtering techniques. Our work is concentrated on the filtering techniques. A survey of the statistical methods can be found in [34], a comparative study of the filtering techniques has been done by Randen and Husoy in [24].

4.1 Filtering for Texture Feature Extraction

A common characteristics of signal processing approaches to texture analysis is the filtering step. This step is justified by the periodic nature of texture information. Randen and Husoy give an introduction into filtering for texture classification and an overview of the different filter types used [24]. As we stated before, a main characteristic of textures is the frequency and the directional information. A common method to determine the frequency of a signal is to filter it with band pass filters which pass different frequency bands, a so-called filter bank, and to measure the energy of the filter responses. For a signal of high frequency the filter passing high frequencies will have high energy output, whereas the other filters will show significant lower responses. These responses need to be measured using an estimation function. We can summarize the necessary practical steps to filter an image as follows:

- Transformation of the image or the region into the frequency domain. This is done because most filters are faster and easier to apply in the Fourier domain.
- Application of the filter bank.
- Estimation of the energy of the responses.

Different types of filters have been used for texture analysis. Laws [15] was one of the first to introduce filters. He proposed 25 separable two-dimensional filters which were constructed from 5 one-dimensional ones. The one-dimensional filters consisted of one low pass, one high pass and 3 band pass filters responding to different frequency bands.

Coggins and Jain introduced a filter bank designed to extract frequency and directional information [3]. They used Gaussian ring and wedge shaped filters. The ring filters extracted frequency information and the wedge filters extracted directional information.

For the work described in this document we chose a filtering approach based on Gabor filters, which are already widely used for texture analysis [12].

4.2 Gabor Functions and Gabor Filters

A Gabor filter bank is based on filters derived from Gabor functions, which have been introduced by Denise Gabor in his *"Theory of communications"* [7]. He proposed a family of functions which can be used to decompose arbitrary functions - similar to the sinus and cosinus waves for the Fourier transform. The aim he had in mind was to send coefficients of this decomposition in communication applications. Only recently these Gabor functions have been re-discovered and a new research domain called *Gabor Analysis* has been formed [5].

One of the main driving forces was the search for a signal representation which should be situated between the time (spatial) representation and the frequency (Fourier) representation. Both representations have advantages and disadvantages: Using the time series of a signal we are able to follow the order of the amplitudes through a temporal sequence. But we do not know which frequencies are used in the signal. Analysing the Fourier transform of the signal we can immediately see the spectrum of the frequencies which are part of the signal. However, localising one of the signals is difficult, since this information is hidden in the phase information of the representation. Gabor proposed to compose a signal not into sinus and cosinus waves but into a series of functions which are derived from one single fixed function as follows:

$$f(t) = \sum_{n,m} c_{m,n} gm, n(t).$$

$$g_{m,n}(t) = g(t - na)e^{2\pi imbt}$$

$$m, n \in Z$$

where the function $f(t)$ is represented as a series of the functions $g_{m,n}$. The functions $g_{m,n}$ are created by shifting the function $g(t)$ in time and frequency, where a and b are the shift parameters. For the function $g(t)$ Gabor proposed the Gaussian function. The image representation described above is called *Gabor wavelet*. More general wavelets hierarchically decompose functions into sums of elementary functions. The elementary functions are compressed or stretched version of a *"mother wavelet"* function. In section 3.4 we described an interest point operator introduced by Loupias [16] which uses Haar and Daubechie wavelets to represent images.

A Gabor filter bank consists of several filters based on the family of Gabor functions described above: Several Gaussian filters which are obtained by dilating and rotating the mother function. The two-dimensional functions used for image processing and their Fourier transform can be written as follows:

$$h_{\lambda,\alpha}(x, y) = h_0 \exp\left\{-\frac{1}{2}\frac{x'^2 + y'^2}{\lambda^2\sigma^2}\right\} \exp^{2\pi Ix'/\lambda}$$

$$x' = x\cos\alpha - y\sin\alpha$$

$$y' = x\sin\alpha + y\cos\alpha$$

$$\widehat{h_{\lambda,\alpha}}(u,v) = \exp\left\{-\frac{1}{2}(\lambda u - 2\pi)^2\sigma^2\right\}\exp\left\{-\frac{1}{2}\lambda^2 v^2\sigma^2\right\}$$

where

- α is the orientation of the filter.
- σ is the size of the filter (the variance of the Gaussian function).
- λ is the spatial period of the sinusoid plane wave (the *"frequency band"* of the filter).
- $h_0 = \left(\frac{2\pi}{\sigma\lambda}\right)^2$ is a normalisation parameter.

In this work we adapted the parameter settings used by Megret in [20], which were justified by Farrokhina and Jain in [12]. The sigma is fixed to $\sigma = 2\pi$. The main parameters are the orientation α and the scale λ. To obtain all filters of the bank these two parameters are changed successive in the following way:

- The orientations α of the filters are cycled through the interval $[0, \pi]$: $\alpha = \frac{a\pi}{K}$, where $a = \{0, \cdots, K-1\}$ and K is the number of orientations of the filter bank.
- The frequency band λ is shifted: $\lambda = \sqrt{2}^k \lambda_0$ where $\lambda_0 = \frac{2}{\pi}$ and $k = \{0, 1, \cdots\}$. k is limited to $\frac{2\pi}{\lambda_0}$.

Figures 4.3 and 4.3 show the spatial domain and the real part of the Fourier transform of the Gabor filter bank used in this work. The filters of one column pass frequencies of the same band, whereas the filters of one row pass waves with the same orientations. Note, that the filters passing high frequency contents show strong values in the centre of the Fourier spectrum, where the coefficients of the high frequencies are situated, whereas the filters passing low frequencies show strong values at the borders of the Fourier image. The orientations passed by the filters can be seen by the angle of the Gaussian distribution to the centre of spectrum (see figure 5.2).

The Gabor filter bank described above is used in this work to extract local texture features from images. The next chapters describe the application of the filters and how different image representations are built from the filter responses.

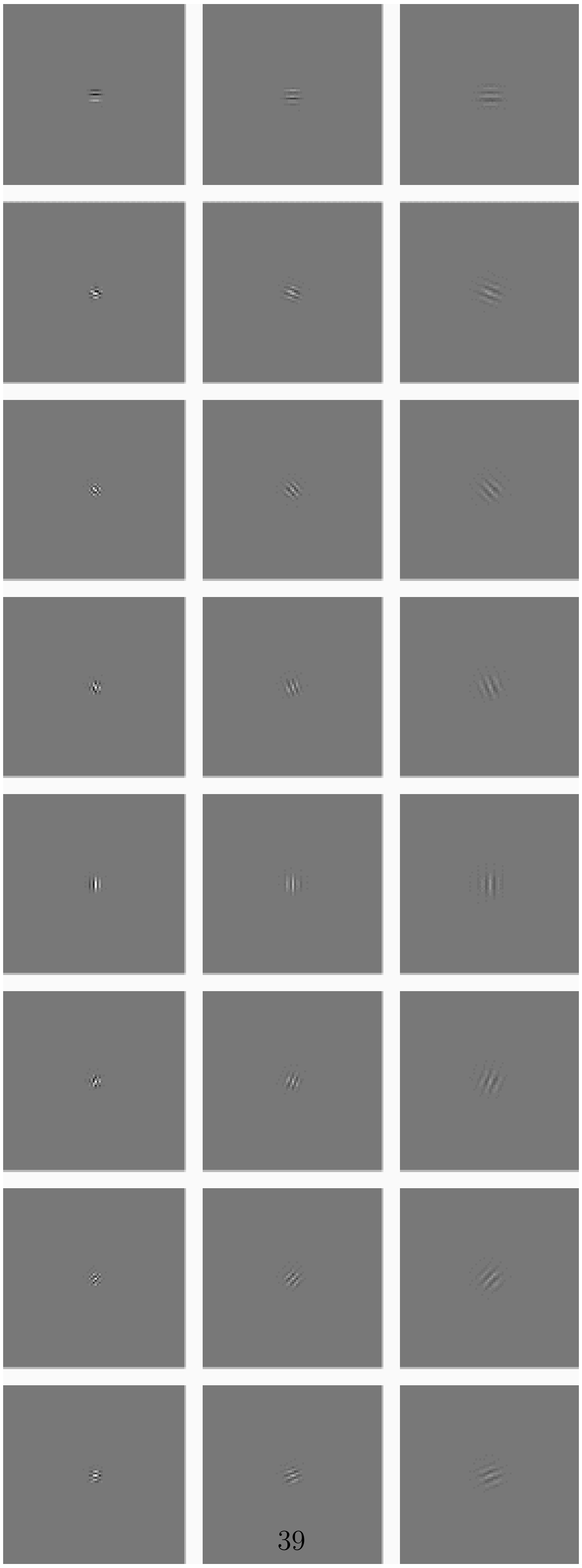


Figure 4.2: The filters of the Gabor filter bank represented in the spatial domain

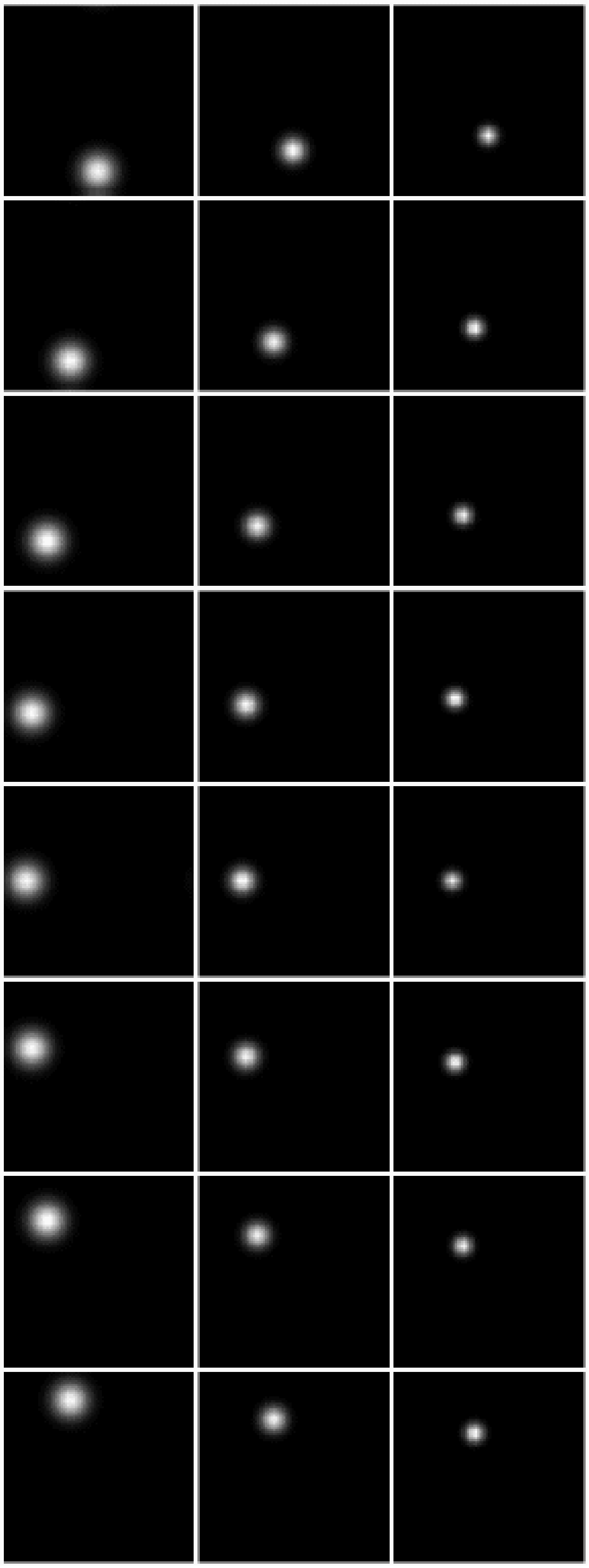


Figure 4.3: The real part of the Fourier transform of the Gabor filter bank

Chapter 5

Texture Features on Interest Points

The last two chapters described methods to collect texture features on images and to extract key points. This section explains methods of combining both texture features on interest points.

The basic idea in this work is to extract a fixed number of interest points N from the image and to select regions of fixed size R around each point, referred to as *interest regions*. To each interest region we apply a (Gabor) filter bank of $S \times K$ filters, K being the number of orientations and S the number of scales. Figure 5.1 shows the basic scheme.

From each of the resulting $S \times K$ filter responses per point the following characteristics can be extracted (see Figure 5.2):

Scale The scale is a property of the filter and determines the frequency interval that is passed by the filter.

Orientation Like the scale the orientation is a property of the filter. It denotes the orientation of the frequency that is passed by the filter. All other orientations are suppressed.

Amplitude The most important description of the filter response is the maximum amplitude of the response, from now on referred to as *amplitude* only. It tells how strong this interest region responds to the filter applied to it. Literally spoken, we could say that it specifies how much structure of the given orientation in the given scale can be found in the interest region.

Phase Although the phase information is important to localise the regular structure and to reconstruct an image from the responses of a filter bank, it is not used very often in indexing techniques, it will therefore not be considered further.

u0, v0 The coordinates of the maximum response in frequency space.

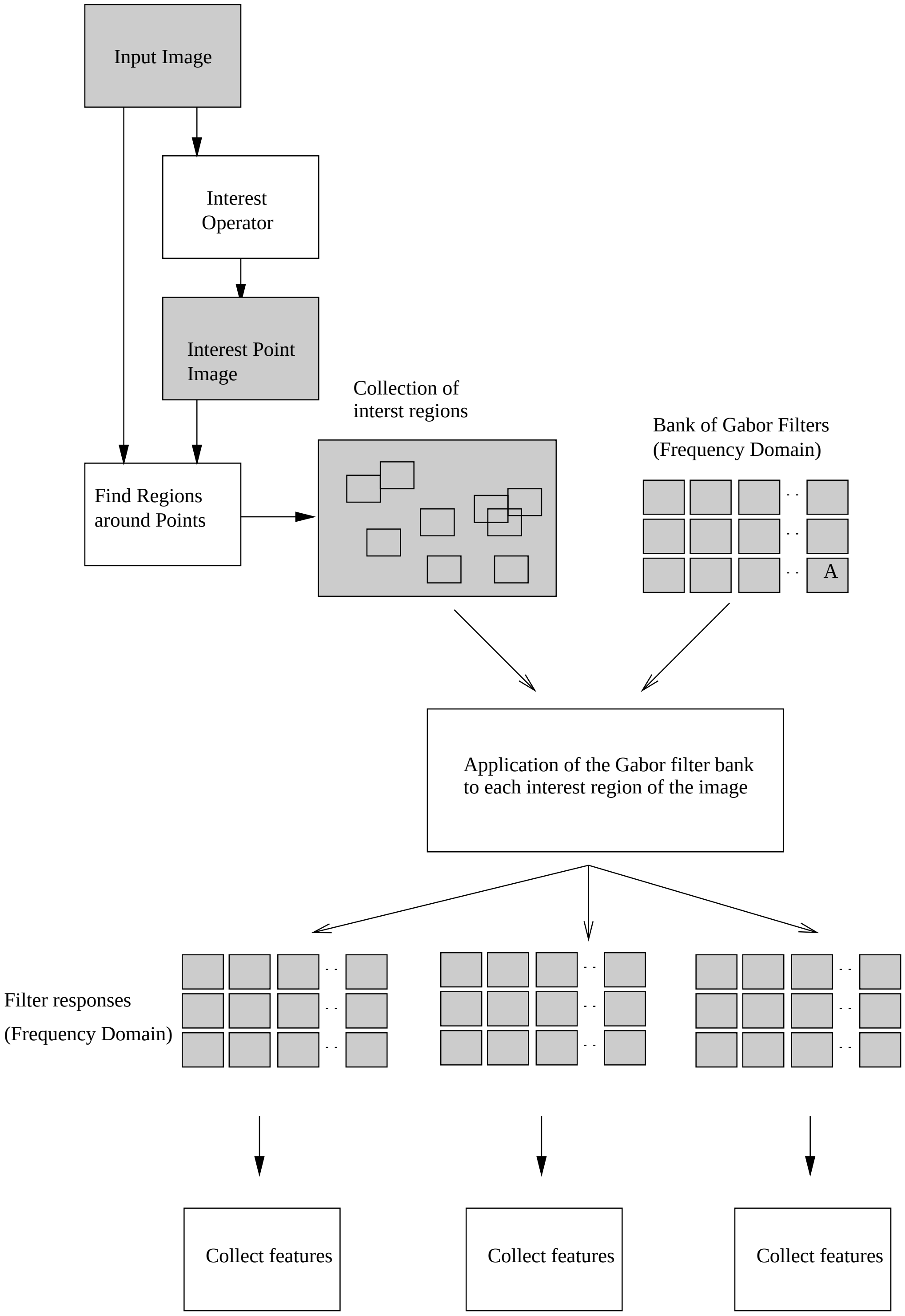


Figure 5.1: Process scheme of collecting texture features on interest points

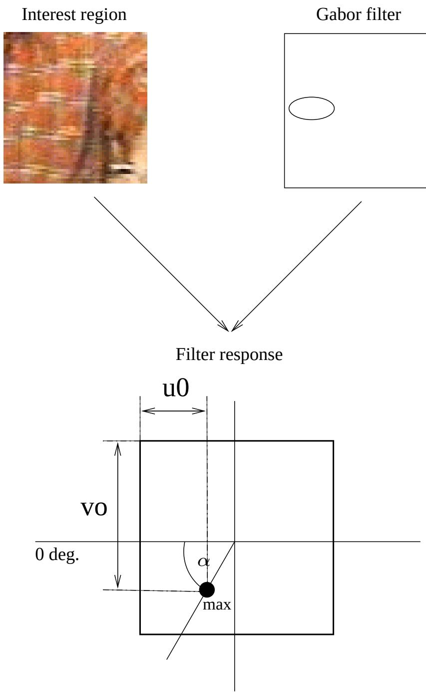


Figure 5.2: Feature information available from a filter response

Angle The angle of the maximum amplitude regarding the centre frequency of the filter. This value is correlated to the orientation of the filter. However, if the main orientations of a filter response are calculated, it can be useful to get a finer range of values.

What kind of information do we get after the extraction process? There are different possible interpretations of this data, which makes different models and comparison algorithms possible. Figure 5.3 shows the result of the feature extraction process. Let's assume, that for each filter response only one feature is extracted: The maximum amplitude. Then the output will consist of the outputs of the filter bank for each of the N interest regions. For each region $S \cdot K$ responses, thus $S \cdot K$ amplitudes $(F_1, F_2, \ldots F_{S\dot{K}})$ are extracted.

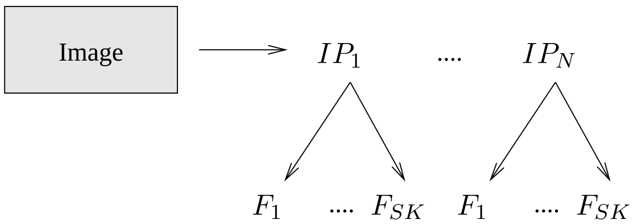


Figure 5.3: The output of the feature extraction process

Different combinations of these features are possible. One interpretation is a set of feature points, each associated with a vector of amplitudes (Figure 5.4.a). Each vector holds the amplitudes of the set of S scales and K orientations. The vectors describe the responses of the regions around the points to filters of different orientations and scales, giving a rough description of "what is happening" around a point. Hence, the description of an image is a set of these features extracted on interest regions. We describe a method of modelling and comparing these sets in chapter 6.

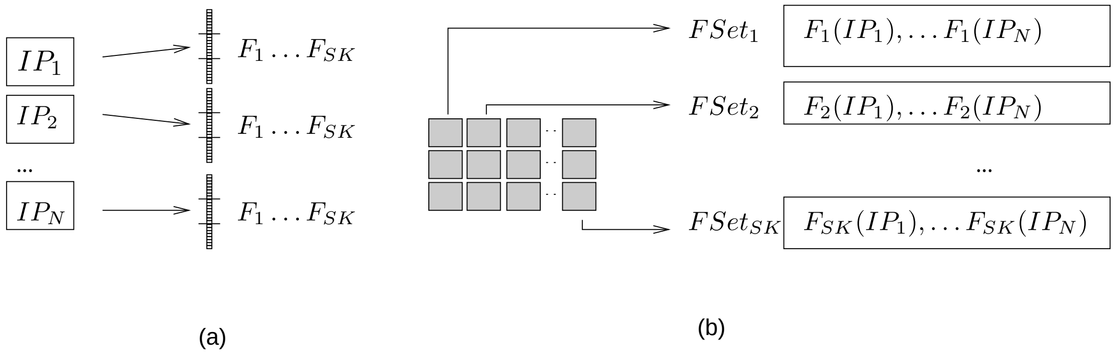


Figure 5.4: Interpretations of the available feature data ordered by interest point index (a) and by filter index (b)

A different and more statistical interpretation could be to order the data according to the scales and orientations of the filter bank (Figure 5.4.b). So for each of the $S \times K$ filters we get a set of N features — one for each interest point (e.g. amplitude and point location). What we get here would be distribution of the amplitude responses for each filter, literally speaking how much the image is responding to the filter passing this frequency and orientation.

Please note, that by simply re-ordering the data we can get a different view of the image, provided that it is supported by the calculation of the distance algorithm. The representation ordered by interest point index is centered on the interest regions of the image. Every region has an assigned feature vector describing it's texture contents, thus the distance algorithms compare regions. On the other hand the representation ordered by filter index provides a view centered on the responses for one filter. A way to describe them is a histogram for each filter describing the amplitude distribution for the whole image responding to this filter. Distance functions for this representation work different, they compare filter responses. Different approaches to the second representation based on histograms are treated in chapter 7.

Chapter 6

Feature vectors on interest points

The last chapters presented our method to extract texture features from digital images using interest point operators and a Gabor filter bank. We explained the different interpretations for the feature data gathered from the images, making it is possible to develop different feature representations. This chapter describes an indexation method based on an image representation, which orders the feature data by interest point index. This can be interpreted as a set of descriptions for interest points.

6.1 Motivation

Methods using feature vectors gathered on interest points have already been used in image indexation and especially object recognition. The goals of object recognition systems are similar to the ones of image databases: Pre-selected objects have to be found and located in one or more digital images. Some of the problems arising are caused by the occlusion of parts of the object by other objects or due to its position to the camera. One way to solve this are feature vectors on interest points (See the method of Lowe [17] in Section 2.2.2). The basics can be described as follows: Some points are detected in the object which needs to be located in the images. Together with the point locations features are gathered (based on colour, texture etc.). In the detection step these points are searched in the images. If a specific number of points can be found, then it can be assumed that the image contains the object. Even if parts of the object are occluded then still other points of the object can be found.

The requirements for image indexation systems are similar, so also in this domain algorithms based on feature vectors have been published. Schmidt and Mohr introduced an image retrieval method based on invariant features in [28] (See section 2.2.2). To compare two images, they detect interest points in both images using the Harris corner detector (See section 3.2) and collect invariant features from a local environment around these points. Then a feature vector is created from the features of each interest point. This way an image is represented by a set of feature vectors.

6.2 Representation

Our image representation presented in this chapter is based on a similar principle. Like already described in chapter 5, we use an interest point operator to extract a fixed number of regions of the image and a Gabor filter bank to extract features from these interest regions. The result of this filtering step are $N \times K \times S$ filter responses, where N is the number of interest points, K the number of orientations and S the number of scales of the Gabor filter bank. From each filter response we keep only one value: The maximum amplitude extracted from the frequency representation. Reordering this data according to interest point index we can assign to each point $K \times S$ amplitude values, which we put into a vector. Thus, an image is represented by a set of N feature vectors similar to the solution of Schmid and Mohr.

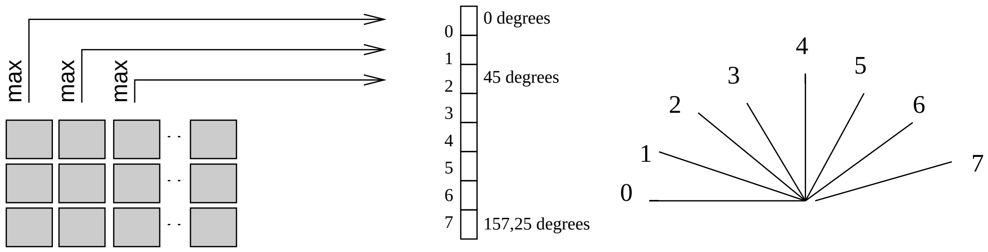


Figure 6.1: Feature vector storing amplitude for each orientation

We created two different types of features vectors. A simple version is shown in Figure 6.1. It consists of K entries, where each entry corresponds to one of K orientations of the Gabor filter bank. To calculate the value for one entry the maximum of all S amplitudes for this orientation is taken, i.e. the maximum for all scales of this orientation, and it's logarithm is stored in the entry. All possible vectors span a K-dimensional feature space, where each vector can be seen as a point.

The final, scale sensitive version of our feature vector keeps the scale information (Figure 6.2). The vector is split into 3 parts, one for each scale, where each part contains the amplitudes for each orientation like the first vector type. Our experiments have been conducted using the scale sensitive version of the feature vector. The simple first version does not represent the multiscale information, nevertheless it is a nice "vehicle" for the demonstration of distances.

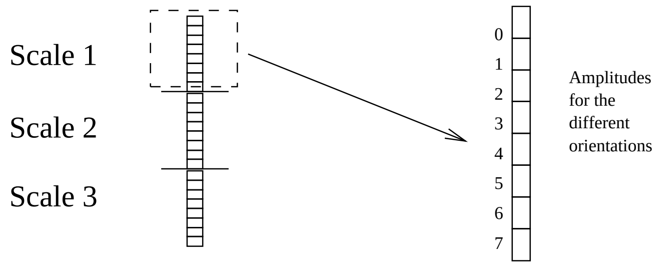


Figure 6.2: The scale sensitive version of the feature vector storing amplitude for each scale and orientation

6.3 Comparing feature vectors

Above we introduced a representation for images. To be able to retrieve images we need a distance function. Our methods to calculate the distance of two images, i.e. two sets of feature vectors, are based on the distances of feature vectors. So we need to define a distance between two single feature vectors first. This distance is different for the two types of vectors. Like we already stated, the simple version of the feature vectors can be interpreted as points in a K-dimensional vector space. We have to find a distance measure which reflects the distribution of the points in this vector space. Let's consider the cloud of points (feature vectors) gathered from the images of our image database. Each dimension represents one of the K orientations in the image. Provided that the orientations in the images of the database are distributed equally, we can assume that these points in the vector space are distributed equally across all components as well. Thus, given two feature vectors μ and ν, the Euclidean distance d_E can be used as distance measure between the two vectors:

$$d_E(\mu, \nu) = \sqrt{\sum_i (\mu_i - \nu_i)^2}$$

where μ_i is the amplitude for direction i of the vector μ. Figure 6.3 shows in 3 histogram plots the distribution of the elements of our feature vectors for one of the test image databases used in this work (See Section 8.2). One curve corresponds to a histogram plot for one element (i.e. one orientation) of the vector. As we can see, our assumption that the different orientations are distributed almost equally can be confirmed in this case.

The Euclidean distance d_E applied to two feature vectors provides in this case an efficient manner to compare the frequency characteristics of two regions. However, the feature vectors are not rotation invariant, so the comparison of two regions is not invariant to rotation as well. That means, that two regions containing the same textures have a higher distance if one of the regions is rotated towards the other one. A human on the other hand would perceive the two textures as similar

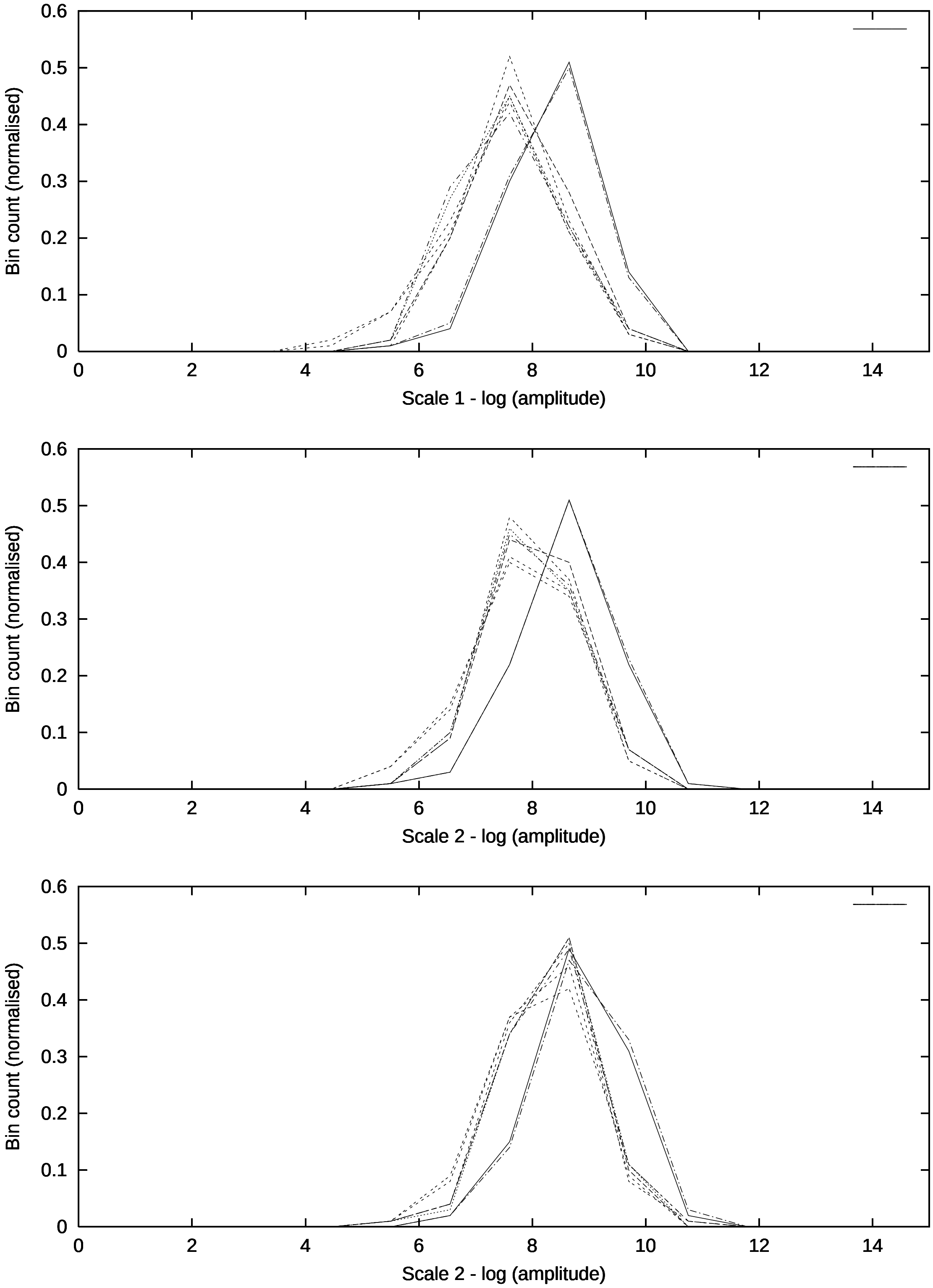


Figure 6.3: Histogram plots of the distribution of the different orientations for all features of a test database, separated by scales.

regardless if they are rotated or not. Classic image indexation applications like image or video indexing, are not interested in a complete rotational invariance but must take into account small variations of orientations. For this reason we introduced some shifting into the distance measure to compensate for rotation. We not only calculate the Euclidean distances between the two original vectors μ and ν but also the distances between μ and two different cyclic permutations of ν. These cyclic permutations of feature vectors are equivalent to rotations of the corresponding interest regions. Given a vector μ, the permutated vector $\rho = perm(\mu)$ is defined by

$$\rho = perm(\mu), \quad \rho_i = \begin{cases} \mu_{i-1} & \text{for } i = 2 \ldots K \\ \mu_K & \text{for } i = 1 \end{cases} \tag{6.1}$$

The K orientations of our Gabor filter bank are spread across an angle of π, therefore the cyclic permutation described above is equivalent to a geometric rotation of $\frac{\pi}{K}$. Given a Gabor filter bank of e.g. $K = 8$ orientations the features are rotated by $\frac{\pi}{8} = 22.5^o$. Hence, to make the distance measurement (We call it D_{F_1} - distance for the first version of our feature vectors) less "rotational sensitive" we add permutated vectors to the comparison process: The distance between two feature vectors μ and ν is actually the minimum of 3 distances: The Euclidean distance between μ and ν, the Euclidean distance between $perm(\mu)$ and ν, and the Euclidean distance between μ and $perm(\nu)$.

$$d_{F_1}(\mu,\nu) = \min \left\{ \begin{array}{l} d_E(\mu,\nu) \\ d_E(perm(\mu),\nu) \\ d_E(\mu, perm(\nu)) \end{array} \right\} \tag{6.2}$$

A similar distance is used for the scale sensitive version of the feature vector (Figure 6.2), which keeps $3K$ elements — K for each scale. But the permutation of the vector elements is done according to the structure of the vector, which resembles the structure of the filter Bank. Hence, the K elements for one scale are permutated separately, i.e. the distance d_{F_2} of two vectors μ and ν is calculated by computing the distances separately for the 3 sub vectors averaging these distances. Denoting the sub vector of μ, which contains the feature elements for scale x, as $\mu(x)$, we get the distance d_{F_2}:

$$d_{F_2}(\mu,\nu) = \frac{1}{S} \sum_{i=1 \ldots S} d_{F_1}(\mu(i), \nu(i)) \tag{6.3}$$

Please note, that during the distance calculation we handle the Gabor orientations different than the scales. We shift the orientations of the vector and calculate intermediate distances. These distances are combined using the *minium* function. It is not possible that a texture is present at more than one orientation, so only one of the distances is taken. On the other hand, we calculate these intermediate distances on each scale, and these distances are combined using the *average* function. The reason is the hierarchical nature of textures, which need to be examined at different scales. The distance measurement defined above computes a distance measure between two interest regions, where small changes in the orientation are tolerated.

6.4 Querying the database

To query the database a distance function for images is needed, which compares two sets of feature vectors. Our distance measurements are based on the distances of vectors. However, there are different ways to combine the distances of single vectors to a distance of sets.

6.4.1 Voting

Schmidt and Mohr proposed in [28] (See Section 2.2.2) a voting algorithm to query a database of images represented by sets of feature vectors. They pre-compute the vectors V_j of all database images M_k offline and store them in the database together with a link to their source images. Given the query image I, it's set of feature vectors V_i is computed, and each vector is compared to each vector V_j in the database. If the distance between V_i and V_j is below a threshold t, the respective image M_k gets a vote. The maximum votes are returned to the user. The cost of comparing two images is the cost of comparing all feature vectors of the query image with all feature vectors of the image to compare, thus $O(N^2)$, where N is the number of vectors per image, i.e. the number of interest points.

We adopted this algorithm to our image representation. This could be done easily, since the representations are very similar. We used our distance measure d_{F_2} described above for the distances between two feature vectors. The algorithm searches for images in the database, which have a high number of interest regions which are similar to the interest regions found in the query image. One region in the query image can be similar to one or more regions in one of the database images and vice-versa. I.e. it is irrelevant if ten regions correspond to ten equivalent regions, or if one region can be found ten times in a database image, the result will be same. Results of our experiments are shown in section 9.

6.4.2 Searching Corresponding Interest Points

We developed a different algorithm, which compares image by image and tries to search pairs of corresponding interest points in both images. The search is done in a greedy manner: The means to qualify two points as being a pair is the minimum distance in feature space. To do this we build a matrix which stores the distances of all possible feature pairs (figure 6.4), the lines i denoting the interest points of the query image, the columns j the interest points of the compared image, and the elements $E_{i,j}$ the distance between point i of the query image and point j of the compared image.

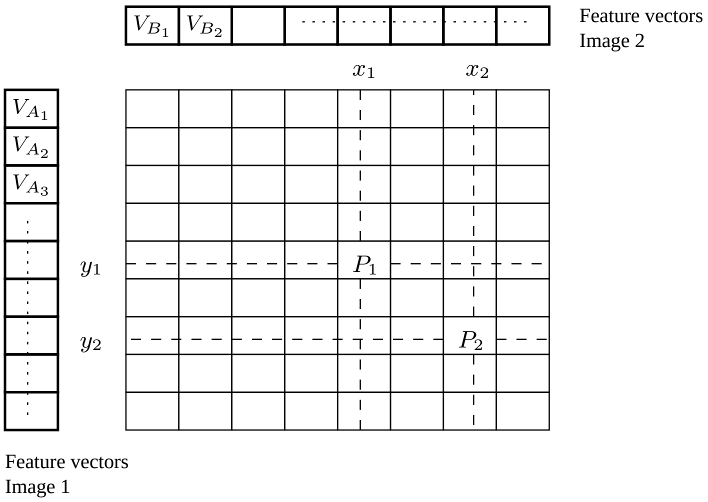


Figure 6.4: The algorithm for searching corresponding interest points

Then we search the minimum element of the matrix. The column and line number of this element denote the first pair of corresponding interest points. Both column and line are deleted from the matrix, since these two points are not available for other pairs anymore. Then we again search the minimum element of the remaining matrix. This algorithm is continued until the matrix vanishes (The points of at least one image are exhausted) or the minimum distance does not exceed a given threshold t (There are no more points having a corresponding partner). The distance $d(A, B)$ between the two images A and B is calculated using the number of corresponding points found:

$$d(A, B) = \frac{2 * \text{Number of corresponding points}}{N(A) + N(B)} \tag{6.4}$$

where $N(A)$ denotes the number of interest points of the image A.

The performance of the algorithm depends on the threshold t. If the threshold is too high, then too many couples of corresponding interest points are found and the system will retrieve false positives. If the threshold is too low, then no or too few couples will be found — false negatives or non deterministic ordering of the result set (in the case of $t = 0$) will be the consequence. We performed experiments with different threshold values for different image collections, whose results we present in chapter 9.

The distance measure presented above has one shortcoming, which will become visible for queries against large databases only. The reason is the granularity of the distance function. Looking at equation 6.4 we can see, that we can get at most $2N$ different distance values, where N is the number of collected interest points per image. Imagine that there are more than $2N$ images in a database, which are relevant for the query image, i.e. which have a high number of corresponding interest points. Although all these images have small distances and will be in the first positions of the result set, the ordering of these images in the result set among themselves will not be deterministic in some cases due to lack of sufficient different distance values.

Motivated by this drawback we developed another distance measurement, which uses not only the number of corresponding interest points for the distance calculation, but also the distances of the feature vectors involved. We give the new measure as

$$d(A,B) = \frac{2 * \sum\limits_{\xi \in C} s(\xi)}{N(A) + N(B)} \tag{6.5}$$

where C is the set of pairs of corresponding feature vectors and $s(\xi)$ is the *similarity* of the corresponding vectors $\xi = (\mu, \nu) \in C$. We calculate this similarity on the basis of the distance in feature space $d_{F_2}(\mu, \nu)$. However, we scale the distance, so that the similarity is equal to 1 if the distance in feature space is minimal, i.e. equal 0, and the similarity is 0 if the distance is maximal. Since the search for corresponding feature vectors is stopped at a threshold t, this threshold is equal to the maximum distance of two feature vectors. Hence we can write the similarity of two corresponding feature vectors as

$$s(\xi) = 1 - \frac{d_{F_2}(\mu, \nu)}{t}, \qquad \xi = (\mu, \nu) \in C$$

Please note, that this distance measure corresponds to the first measure defined in equation 6.4 if all vector similarities $s(\xi)$ are set constant to 1. Literally spoken, if we consider images represented as sets of sub images (regions), then our measure defined above allows a finer granularity of the distance values since not only the number of "similar" regions contributes to the distance, but also the similarity itself.

We performed experiments using both distance measures as well as the voting algorithm introduced by Schmid and Mohr. Results of these experiments can be found in chapter 9.

The cost for calculating the distance matrix is $O(N^2)$. The search of the corresponding pairs is dependent on the similarity of the two images. The higher the similarity the higher the cost, since the search for the minimum distance in the matrix has to be done more often. The first search takes $O(N^2)$, the next one $O((N-1)^2)$ etc. So, if for all interest points correspondences can be found, the overall cost of the algorithm is $O(N^3)$, where the cost of one step is small (comparing the value of the matrix with the current maximum). However, in

Figure 6.5: Two images and their map of corresponding interest points

reality the maximum number of point pairs is almost never found. The overall cost of the algorithm is

$$O(N^2) + O(N^3)$$

Figure 6.5 shows two test images and their map of interest points. Corresponding points are connected by a line or by a single big rectangle, if their spatial distance is smaller than 5 pixels.

Chapter 7

Histograms

In the last chapters we described our method to extract texture features from digital images using interest point operators and a Gabor filter bank. We also introduced one of the different possible interpretations for the feature data, which uses feature vectors sets to represent images. This chapter describes an indexation method based on an image representation, which orders the feature data by filter index and uses histograms for the representation. Histograms have already been used for image indexing extensively, especially colour based histograms [33] and structure based methods [11]. They provide an effective and efficient means to compare image contents. However, they rely on raw image data like colour or gray values, which are not available for textures without pre-processing. In this chapter we describe a method based on histograms filled with the data gathered from our feature extraction method described in chapter 4.

There are several reasons to use histograms to describe image contents:

Well known distance measures Unlike other image representations the distance measures for histograms are not as dependent on the type of features and the contents of the histogram, so the standard distance measures can be taken. Histograms are standard tools of statistics and also image processing, so powerful distance measures to compare them have already been developed (See section 7.3).

Speed One of the main strong points of histograms is the availability of fast comparison algorithms. Using interest points one advantage is the independence of the number of interest points, whereas our previously developed methods suffer from high computational complexity ($O(N^2)$).

View Histogram representations provide a different view on the data. The algorithms we presented in the last chapters described images as a set of feature vectors collected around points of interest. Comparison was done by searching corresponding interest points, the main "dimension" of our image key being the set of interest points. Histograms are more flexible. Their descriptive power depends on the way how they are *"filled"*, i.e. their power depends on the way how the feature data is translated into the raw data necessary to create them.

For the development of image retrieval systems the emphasis is not laid on the distance measure but on the design of the image representation, i.e. the contents of the histogram. It is necessary to design the contents in a way, that the histograms are representative. Different images should have different histograms, i.e. their distances should be large. We already mentioned this general requirement for image features in section 2.1. However, even if the image features, which need to be collected before building the histograms, fulfill this requirement of descriptiveness, it is still possible to generate badly scaled histograms by choosing wrong parameters.

The parameters need to be designed according to the feature data. Firstly, it is necessary to find a way to transform the feature data into a raw format suitable to "fill" it into the histograms. Moreover, size and format of the histograms must be well chosen. The main parameters, which need to be fixed for a histogram based representation, are the borders, and the bin count. The borders specify the interval which is represented by the histogram. Values, which do not fit into this interval are placed into the lowest or highest bin respectively. If this interval does not fit the distribution of the data, then the possibility of similar histograms for different features arises, because the differences of the feature data are hidden in single histogram bins.

In the following sections of this chapter we will present two different representations based on histograms. We close the chapter with a survey and description of histogram distances in section 7.3.

7.1 Amplitude representation using sets of Histograms

Considering the output of our Gabor filter bank the main information we gathered is the amplitude information for different orientations and scales of the Gabor filters per interest region. If we re-order the data we get a distribution of responses (i.e. amplitudes) per scale and orientation. The filter responses for one filter can be filled into a histogram, which therefore models the response of the image to this filter with the given orientation and the given scale. Images responding much to this filter have high bins with high bin indices and low bins with low bin indices and vice versa. Thus, the feature data of the whole image, i.e. of all filters of the filter bank, can be stored in an ordered set of $K \times S$ histograms, where S is the number of scales and K is the number of orientations.

Filling the histograms this way we ignore the spatial coherence of the different interest regions, i.e. the locations of the regions are not represented in the histogram. It does not matter where an interest region is found, since only the contents (translated into filter responses) is used to create the histograms. In order to add the spatial coherence to our representation we use two-dimensional histograms instead of one-dimensional ones. We need couples of data instead of single values to increase the histogram bins. These couples are created by performing a n-nearest neighbour search for every interest point.

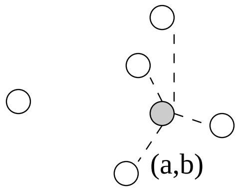


Figure 7.1: Nearest neighbour search

The entries, which increase the histograms, are pairs of interest points. All points of the image are traversed and taken as first points of these pairs. The second points are the neighbours found by a n-nearest neighbour search performed for each interest point found during the traversal. (Figure 7.1). Hence, for each interest point traversed we create n pairs of points. The amplitudes of the couples are entered into the histogram: The amplitude of the first point is used to calculate the x-index of the bin, the amplitude of the second point (the neighbour) to calculate the y-index.

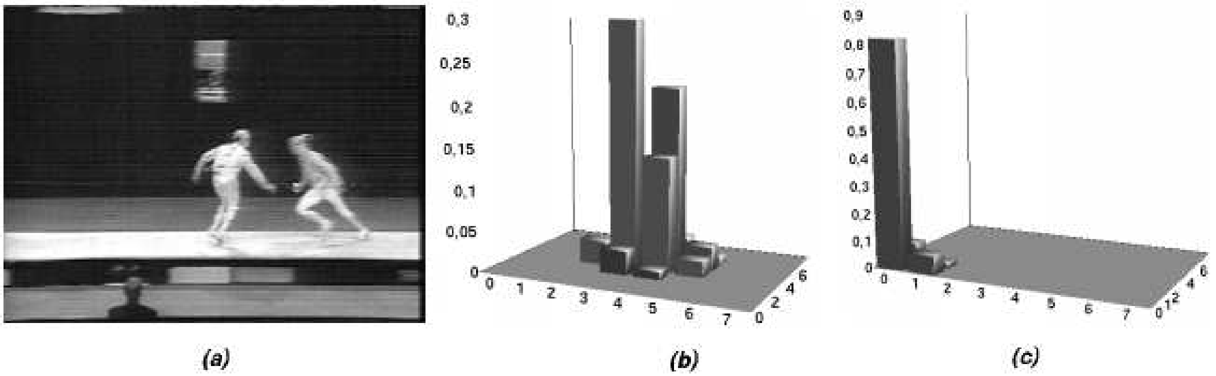


Figure 7.2: Image representation using ordered sets of histograms containing amplitudes: Example picture (a) and two histograms of the ordered set: orientation $0 = 0^o$ (b) and orientation $2 = 45^o$ (c) for scale index 0

Figure 7.2 shows an example image and two histograms of the set. We chose an image which contains frequencies mainly in the horizontal orientation (orientation

0). Therefore, the histogram for orientation 0 shows a distribution of strong responses, i.e. strong bins from indices 4 to 6. The histogram for orientation index 2, which corresponds to structures in orientations around 45 degrees, shows only one strong bin at index $(0, 0)$, i.e. almost no response.

The distance function necessary to compare this image representation needs to compare two sets of histograms. Our distance measure is based on the histogram distance measures discussed in section 7.3. A simple possible distance measure compares corresponding histograms of both sets and averages the distances:

$$D_{set}(A, B) = \frac{1}{S \cdot K} \sum_{u=1}^{S} \sum_{v=1}^{K} D(H_{Auv}, H_{Buv})$$

where $D_{set}(A, B)$ is the distance of the two images A and B, i.e. the distance of the two ordered sets, and H_{Auv} is the histogram for scale u and orientation v of image A. However, similar to the distance of feature vectors we included some compensation for rotation by comparing each histogram not only with its corresponding histogram but as well with the immediate neighbours of the same scale. In our feature vector approach (chapter 6) we used rotations of the feature vectors to compensate image rotations (equation 6.1). Similar we can partition our ordered set of $S \times K$ histograms into a set of S vectors, each containing K histograms. For each vector, which corresponds to one scale of the Gabor filter bank, we calculate intermediate distances. Furthermore we perform cyclic permutation to decrease the sensitiveness to rotations of the interest regions.

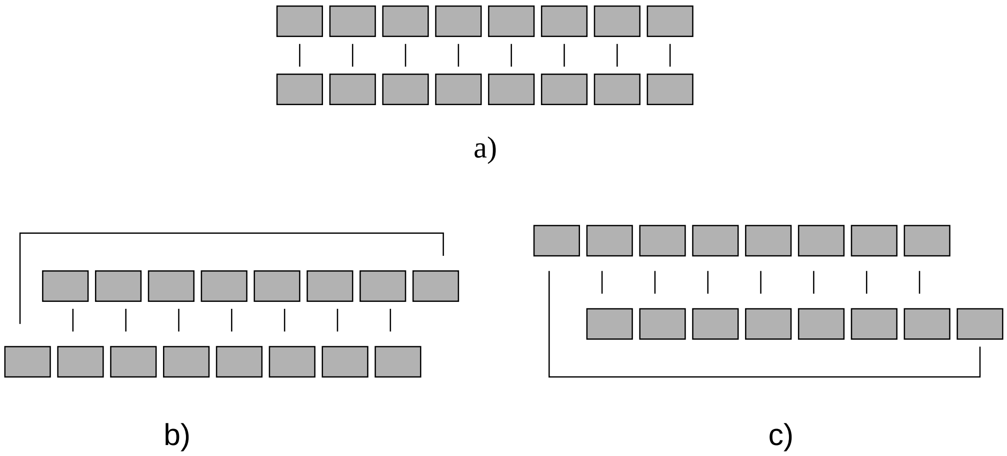


Figure 7.3: Comparison of ordered histogram sets: to compensate for rotation not only the corresponding histograms are compared (a), but also the immediate neighbours by rotating once in both directions (b), (c) and using the minimum.

Figure 7.3 depicts this rotation process. We not only calculate the distances between the two histogram vectors μ and ν but also the distances between μ and two different cyclic permutations of ν. These cyclic permutations of histogram vectors are equivalent to rotations of the corresponding interest regions. Given a vector μ, the permutated vector $\rho = perm(\mu)$ is defined by equation 6.1, similar to the rotation of vector vectors in chapter 6. The final distance between the two histogram vectors μ and ν (which do not represent a whole image but one scale only) is the minimum of the 3 distances:

$$d(\mu, \nu) = \min \left\{ \begin{array}{c} d_E(\mu, \nu) \\ d_E(perm(\mu), \nu) \\ d_E(\mu, perm(\nu)) \end{array} \right\} \tag{7.1}$$

The final distance $d(A, B)$ of two images A and B, i.e. of two histogram sets containing histogram vectors for each scale, is the average of the intermediate distances calculated for each scale.

$$d(A, B) = \frac{1}{S} \sum_{i=1 \ldots S} d(\mu(i), \nu(i))$$

where the $\mu(i)$ are the histogram vectors for scale i of image A and $\nu(i)$ are the histogram vectors for scale i of image B. Please note, that the distance calculation for the histogram sets corresponds to the distance calculation of vector vectors in chapter 6. The histogram vectors described above correspond to subvectors of a feature vector, and the complete histogram set corresponds to a complete feature vector for one interest point.

7.2 Differences of Amplitude and Neighbourhood Ranking

The histograms described in the last section were built on raw image data, like amplitudes. The data was fed pairwise, but still there was no relation whatsoever between the points. Huet and Hancock used pairwise geometric histograms to index large databases of line patterns [11]. They took pairwise geometric attributes like differences of angles and differences of length of pairs of line segments to build histograms. The idea was later adopted to structural image retrieval based on interest points by Popescu in [23]. The idea was to use the locations of the interest points to determine geometric attributes. Instead of pairs triples of interest points are taken, and the angle of the points is used to create the histograms (See section 2.2.4).

The main idea of these algorithms is the storage of relative feature data. Huet and Hancock fill histograms with differences of angles, i.e. a relative measure between two geometric elements. The images are described by the information how geometric attributes of line segments change in a local neighbourhood.

We use a similar approach based on the output of the Gabor filter bank. Like in the last section an image is characterised by an ordered set of $S \times K$ histograms. But instead of the absolute amplitude values we store differences of amplitudes for couples of interest points. For each filter response we traverse all points and search the n nearest neighbours. To calculate the bin index of the first dimension of the two dimensional histogram, we use the difference of amplitudes for each pair interest point and neighbour. The bin index of the second dimension is defined by the ranking of the neighbourhood.

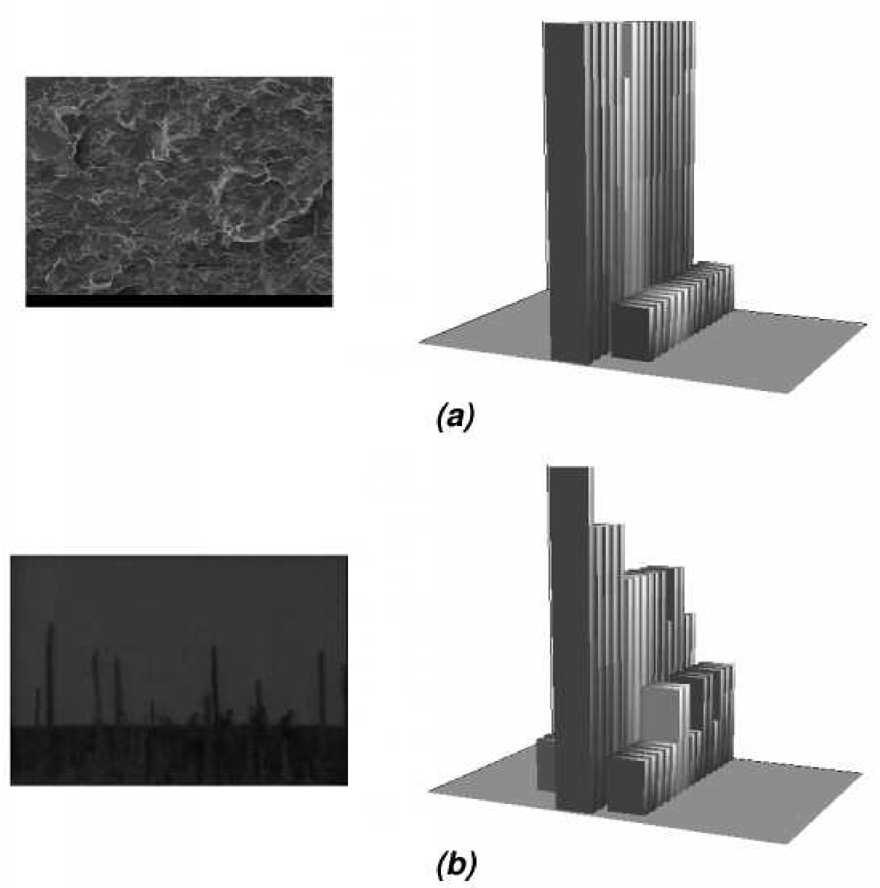


Figure 7.4: Image representation using ordered sets of histograms containing differences of amplitudes and neighbourhood ranking: Two example pictures and their histograms for scale index 0 and orientation $0 = 0^o$.

The information we store in this representation can be described in the following way: Like in our first histogram representation the data is separated by filter index, i.e. one histogram for each orientation and scale of the filter bank. Each histogram holds the information how the amplitudes for this filter change between the interest points to their neighbours. The y dimension of the histogram stores the neighbourhood ranking of the n-nearest neighbourhood search, thus the bins with higher y indices store the differences of amplitudes between interest points which are further away than couples of interest points stored in the bins with lower y indices.

Figure 7.4 shows two example images of one of our test databases and their histograms for scale index 0 and orientation index 0, which corresponds to an orientation of 0^o. The histogram of the texture example (Figure 7.4.a) shows an almost uniform distribution along the y coordinate of the histogram. This can be explained by the periodic nature of the texture example, where the differences in the amplitudes are almost constant across the different distances between two interest points. Considering the histogram of the natural image on the other hand (7.4.b), we remark the difference in the bin sizes along the y coordinate of the histogram. The reason is the different type of the image, which contains sharp changes in structure. In this image the distance between two interest points determines the difference in their responses to a specific filter, hence the difference in amplitudes.

The comparison of the histogram sets is done the same way as for the representation using the absolute amplitude values described in section 7.1. The distances between the corresponding histograms are combined in a way to tolerate small rotational changes in the images.

7.3 Histogram Distances

Our query methods based on histogram set representations use standard histogram distance measures which are already well known. Surveys of different measures can be found in [26]. Huet and Hancock also used different histogram distance measures to index cartographic material [11] (Section 2.2.4).

Histograms can be seen from different viewpoints. Depending on the point of view various distance measurements are possible:

The Minkowski form distances

One dimensional histograms with n bins can be seen as vectors in an n-dimensional vector space. Therefore, the distance measures known from vector spaces can be used. One example is the family of the Minkowski distances, whose general form is given by

$$D(H_A, H_B) = \sum_i |H_A(i) - H_B(i)|^r$$

Well known examples are the L_1 and the L_2 distances for $r = 1$ and $r = 2$ respectively. The L_2 distance of two histograms is equivalent to the Euclidean distance of the corresponding vectors.

$$D_{L_1}(H_A, H_B) = \sum_i |H_A(i) - H_B(i)|$$

$$D_{L_2}(H_A, H_B) = \sqrt{\sum_i (H_A(i) - H_B(i))^2}$$

The Bhattacharyya distance

The Bhattacharyya distance is a measure of correlation between to discrete distributions. Bins with zero contents do not contribute to the distance value, so the distance measure works best with histograms that are "well filled". The Bhattacharyya distance measure is given by

$$D_{Batt}(H_A, H_B) = -ln \sum_i \sqrt{H_A(i) \cdot H_B(i)}$$

The earth mover's distance

To understand the earth mover's distance, the two histograms which have to be compared must be understand as a set of piles of earth and a set of holes. The distance of two histograms is the minimal cost of transporting the filling the holes (the second histogram) with the earth from the piles (the first histogram). Thus, calculating the distance corresponds to solving a linear transportation problem.

A big advantage of the earth mover's distance is its characteristic as cross bin histogram distance measure. I.e., that not only the sizes of corresponding bins are compared but also neighbouring bins. This way the distance is less sensitive to shifts in the feature data and it allows for partial matches. According to Rubner et al. the distance better matches perceptual similarity than other distances [26]. A disadvantage of this distance measure is its high computational complexity compared to other measures.

We performed experiments using the measures L_1, L_2 and Bhattacharyya. In our query methods we store the feature data in two dimensional histograms. However, the distance measures are applied in a similar way. Results of these experiments are given in chapter 9.

Chapter 8

The Test Environment

This chapter explains the environment used to realise and test the proposed methods. The following sections describe the different test images databases we used in our experiments. We specify the size of the test databases by two values: The symbol B denotes the number of images in the database and F denotes the number of images used as query images in the experiments. This number depends on the contents of the database, since only images which have enough empirical similar images, can be used as query images.

8.1 The first test database

Our first test database contains $B = 609$ images grabbed from a single French television channel. The images are all of the same format (384x288 pixels) and coded in JPEG with 75% quality. The contents differs from outdoor activities (reports of sport activities) to talk shows, full scope shots of people, weather forecasts, logos and advertisement.

To be able to measure query performance (see chapter9.2) a clustering of the image set was necessary. The clustering was done manually using empiric criteria in a way that all images in a cluster are perceived as similar by us. In fact, the pictures of one group mostly are taken from the same telecast and sometimes even from the same scene. Figure 8.2 shows examples of these groups. However, although all images of the database are queried not all of them are grouped into clusters. The reason was to avoid too small groups, which would degrade the query performance curves without justification. Eliminating all clusters with less than 10 images, we get 568 (referred to as F in this document) images grouped into 11 clusters (Table 8.1). Figure 8.1 shows the representants of each cluster.

nr.	1	2	3	4	5	6	7	8	9	10	11
images	10	11	14	15	15	19	32	36	86	156	174

Table 8.1: The cluster sizes of test database 1

Figure 8.1: The representants of the 11 image clusters used in the first test database

8.2 The second test database

The second test database contains $B = 179$ images from various sources collected by Jean-Michel Jolion [1]. The contents differs from portraits, textures, landscape scenes to drawings etc. Figure 8.3 shows examples of this database. Again the images of one column belong to the same cluster of images perceived as similar. Table 8.2 shows the clustering of this database. Again we eliminated all clusters with less than 10 images, which left $F = 105$ images for this database to use as query images, grouped into 6 clusters with the following sizes.

nr.	1	2	3	4	5	6
images	10	12	14	15	26	28

Table 8.2: The cluster sizes of test database 2

8.3 The third test database

To evaluate the dependencies of our algorithms on some parameters we use an additional third image database. This commercial database contains various natural images (architecture, landscapes, animals etc.). Figure 8.4 shows examples of this database. Again the images of one column belong to the same cluster of images perceived as similar. Table 8.3 shows the clustering of this database. $F = 105$ im-

[1]Laboratoire Reconnaissance de Formes et Vision, INSA de Lyon

ages are used as query images which are grouped into 9 clusters with the following sizes:

nr.	1	2	3	4	5	6	7	8	9
images	12	17	18	19	35	83	99	131	163

Table 8.3: The cluster sizes of test database 3

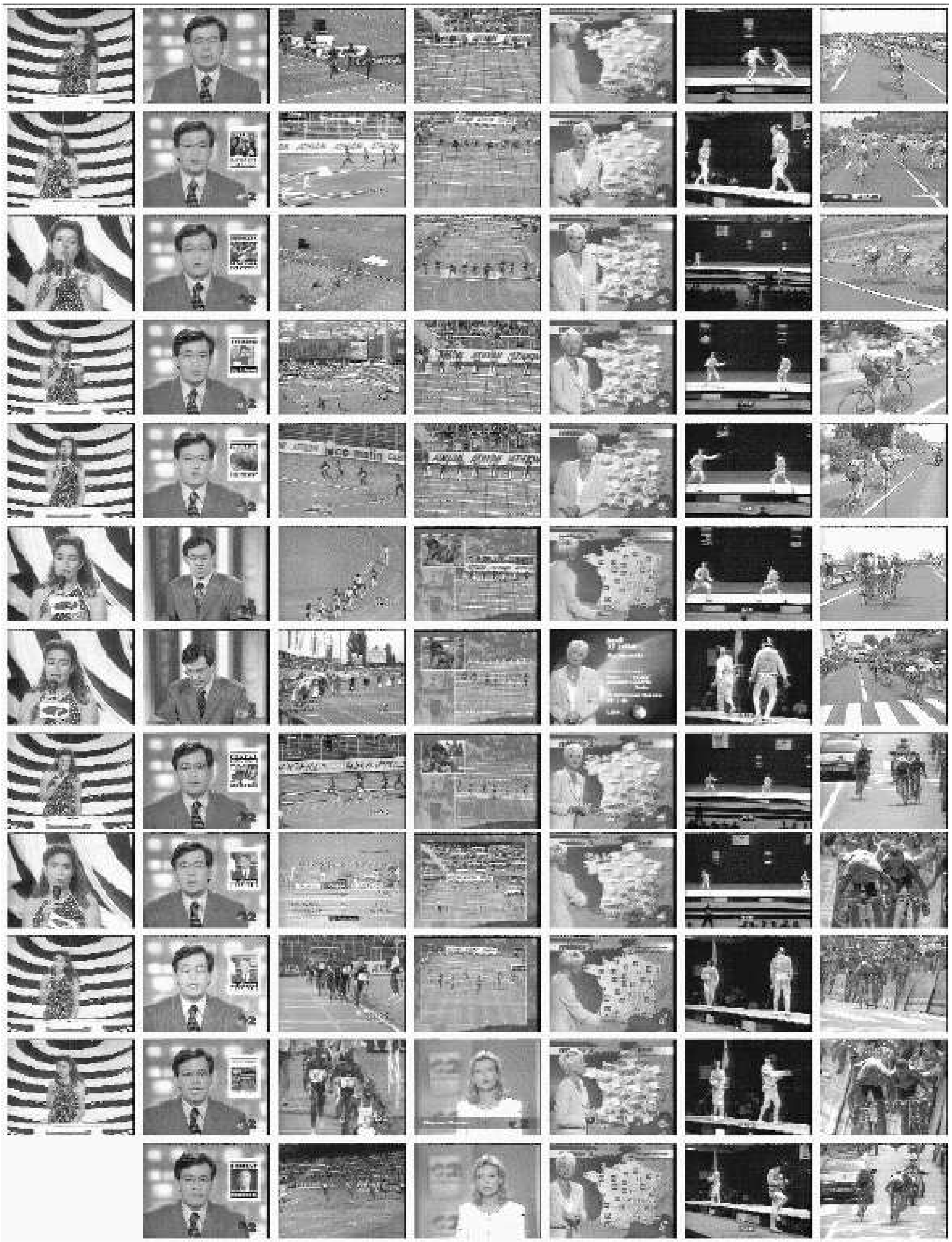


Figure 8.2: Examples of the first image database used (609 images total). All images in one column belong to the same cluster of images perceived as similar.

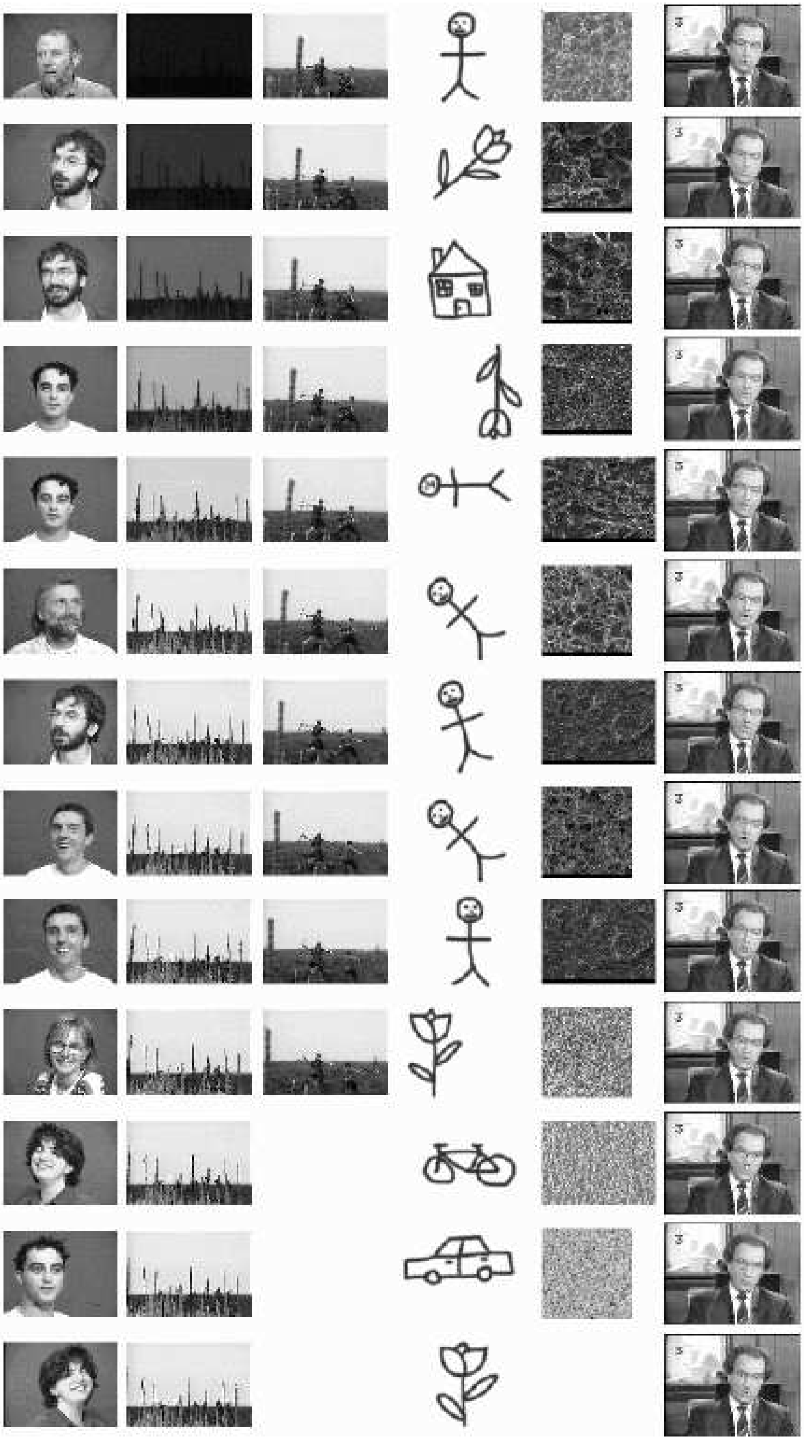


Figure 8.3: Examples of the second image database used (179 images total). All images in one column belong to the same cluster of images perceived as similar.

Figure 8.4: Examples of the third image database used (1505 images total). All images in one column belong to the same cluster of images perceived as similar.

Chapter 9

Experimental Results

This chapter presents a detailed experimental evaluation of the proposed methods. Before we can do that we have to discuss how to measure query performance in image retrieval. Considering a single query from a given image which returns a result set of d images, there are two relevant criteria to take care of: query speed and quality (the relevant-ness) of the result set. The former is easy to measure, we will discuss it in section 9.1. The latter is not as easy to compare, never the less there are well defined methods. We will discuss it in section 9.2

9.1 Speed

Speed is a performance criteria easy to measure. Two processes are relevant: The creation of the index of an image and the comparison of the query image with the database images. The time to index an image is independent of the number of the images in the database. It is needed when uploading a new image and when querying the database. It consist of several steps: Eventually converting the image to the desired format and to gray scale, applying the interest operator, creating the interest regions, applying the filter bank to each region, and creating and writing the key structure. Table 9.1 gives an overview of the different parts of the index creation algorithm and their speed/time issues.

Applying the filter bank is the part which takes most of the time and heavily depends on the number of interest points and the size of the interest regions used. If histograms are produced then the time to fill them effects indexing speed as well. Especially with growing numbers of interest points histogram creation becomes an issue as well (The algorithm is $O(N^2)$).

Table 9.2 gives an overview of indexing performance dependent on different values of region sizes and numbers of collected interest points. The values are measured indexing a single image *excluding* the time to collect the interest points, which is dependent of the detector. The hardware platform was a standard industrial PC equipped with a Pentium I processor at 300 MHz. The query system has

Task	Time depends of
Detection of interest points	Size of the image
Application of the filter bank	Number of interest points, size of the interest region, size of the filter bank
Creation of histograms	Number of interest points, number of neighbours searched for each point, size of the filter bank if histogram sets are used

Table 9.1: The different steps during the creation of an index

been implemented in C++.

points / region size	64	32	16
100	n.s.	5s	n.s.
200	22s	8s	3s
300	n.s.	12s	n.s.

Table 9.2: Query speed for different interest point counts

The time to query the database depends on the method and the number of images compared. Table 9.3 gives an overview of the speed of the different query algorithms applied to the $B = 609$ images of our test database 1. To obtain precise results for each method we used every image of the reference set of F images as query image and queried against the whole database of B images. The response time divided by F gives us the average time it takes to query one image against the database of B images. If not stated otherwise the methods used $N = 200$ interest points and a region size of 32×32 pixels. The time specified does *not* include the time to create the indices.

Algorithm	Time to query one image against F images
Searching corresponding interest points (100/70/50/30 points)	37s/9s/5s/2.9s
Histogram set: amplitude $\times$ amplitude (24 histograms of 8×8 bins)	2.6s
Histogram set: difference of amplitudes $\times$ ranking (24 histograms of 8×16 bins)	5.1s

Table 9.3: Query speed for different algorithms

The query time using the method of searching corresponding interest points depends on the number of interest points collected ($O(N^2)$). However, although the query performance is depending on the point count as well the differences between the performance curves are not that big (Figure 9.14).

9.2 How to measure Query Performance

The more important criteria of an indexation method is the quality of the result set, referred to as query performance from now on. Since there is no exact definition for similarity between images, measuring retrieval performance is a difficult task.

As an example see figure 9.1. These two images have been split up into 2 different clusters of the test database, i.e. from the users point of view they are not similar. Nevertheless some query methods return image (9.1.b) when querying for image (9.1.a). Can we blame any statistical process, which does not actually understand the contents of an image, to judge the images to be similar? From a texture point of view the images *are* similar. They both contain mainly structures in vertical directions in similar frequency bands. Thus to be able to compare two query methods we need to know the structure of the test database, the type of images used, the clustering into similar images etc.

(a) (b)

Figure 9.1: How do we measure query performance - are these images similar?

User annotation is necessary to observe the query process and check the quality of the query methods. We solved this problem by creating a set of reference images for each image used as query image. The reference images have been classified as "similar" to the image by us according to what we think is visual similarity. For our test databases (See chapter 8) we used the images of the same cluster as reference images of a given image. Thus, the query performance measurement depends on the user and it's judgement of similarity. All experimental results depend on the judgement of the user, who himself picks similar reference images for all query images.

In chapter 8 we explained the structure of our test image databases and their clustering into clusters of images regarded as visual similar. A single query searches visual similar images for a single query image taken from these databases. The query image is a member of a cluster C, which contains d images. These d images are called "relevant" images. The system answers with c images of which r are from the original cluster C. For a single query two measures are widely used for indexation systems: Precision and recall. Their definition is given by:

$$P = \frac{r}{c}, \quad R = \frac{r}{d}$$

where the following symbols where used:

P	Precision
R	Recall
r	Number of relevant images (i.e. from the same cluster) in the return set
c	Total number of images in the return set
d	Total number of relevant images (i.e. from the same cluster) in the database

9.2.1 Performance curves

As the name suggests the precision of the result of a single query denotes how precise the result set responds to the desires of the user. The higher the precision the higher the percentage of relevant images in the result set. By changing the number c of returned images by the system we get a curve of precision:

$$P(c) = \frac{r}{c}$$

For most querying methods the set of result images is ordered, returning the most similar images first. The images having higher indices in the result set are less likely to be relevant to the query image. Hence, the curve will generally show a decreasing value of precision for an increasing c.

Recall of a single query denotes how many of the relevant images in the database have been returned. The higher the recall the more the set of relevant images in the database has been "covered" by the query.

$$R(c) = \frac{r}{d}$$

Since the denominator d is independent of the size of the result set and r will grow with increasing c, the curve of recall will increase with increasing c.

Both of the measures are dependent of the size and the structure of the database, especially the number and size of the groups of similar images (see chapter 8). However, generally it is precision which is regarded as the more important measure to compare different methods. To create the performance curve for a method we use every image of the reference set of F images as query image and query against the whole database of B images. For each query we create a curve changing the size

c of the results set. The final curve is the average value of all curves for the single queries:

$$P(c) = \frac{1}{F}\sum_{i=1}^{F}\frac{r_i}{c}\ , \quad R(c) = \frac{1}{F}\sum_{i=1}^{F}\frac{r_i}{d_i} \tag{9.1}$$

where r_i is the number of relevant images in the result set of query image i and d_i is the total number of relevant images for query image i in the database.

The figures in this chapter display the query performance we achieve using our test databases and different query methods. The x-axis of all curves corresponds to the number of images in the return set c, with a range from 1 to 30. If not stated otherwise the y-axis displays the average precision for all query images (equation 9.1). Other possible measures are the variance of precision or the average recall.

For convenience the legends of the different query curves are sorted by performance in each figure. The legend of the best method is displayed at the top of the list. Exceptions are curves, which cross other curves. However, this case does not occur very often.

9.2.2 The limits of query performance

As we already noted the curves are dependent on the clustering of the database. A theoretical best curve of constant 100% precision can be reached only if the sizes of all clusters are equal or greater than the maximum number of returned images c. A query using a query image within a cluster having $n < c$ images cannot return more than n relevant images ($= r$). But that means that the precision $p = \frac{r}{c}$ is less than 100%. Our results display curves between 1 and 50 returned images. Since our test image databases contain groups with less than 50 images (see sections 8.1 and 8.2) we need to know the theoretical query limit, i.e. the optimal curve which can never be beaten by any method. A query producing the optimal result always returns the maximum number of relevant images, i.e. either the whole return set is relevant if the cluster is big enough, or at least the whole cluster:

$$r_{max} = \min(c, d)$$

Hence the curve of the optimal method is defined by

$$P_{max}(c) = \frac{1}{F}\sum_{i=1}^{F}\frac{\min(c, d)}{c}$$

In a similar way the lower limit of query performance can be calculated. The lower limit would be a query that returns as many false images as possible. Actually it is not this limit which is interesting but the curve of a random query. A random query performs better than the worst query possible. However, a good method should perform better than a random query. A single query choosing random images will return r_{random} images:

$$r_{random} = c\frac{d}{B}$$

Hence the curve of a method choosing random images can be written as

$$P_{random}(c) = \frac{1}{F}\sum_{i=1}^{F}\frac{c\frac{d_i}{B}}{c} = \frac{1}{F}\sum_{i=1}^{F}\frac{d_i}{B} \tag{9.2}$$

The upper and lower limits depend only on the clustering of the test database. The curves created for our test databases are displayed in figures 9.2, 9.3, 9.4 and 9.5. The curves for a random query are created with two different methods: Calculated using equation 9.2 and experimentally using a random number generator.

9.3 The Experiments

In this section we present our experiments and the results measured using the methods described above. We explain in detail the parameters used for the experiments and their influences on the query performance. We compare the results for the different query methods we implemented including the parameters we used and we show the differences of the performance of our system applied to our different test databases.

In the previous chapters we introduced query methods based on two types of image representations, which we want to compare in this section:

- The feature vector set representation using Schmid and Mohr's voting distance and the distance function searching corresponding interest points.
- The representations using sets of histograms which store amplitudes or differences of amplitudes and neighbourhood ranking respectively.

We conducted experiments to evaluate and to optimize the performance of our proposed methods. Section 9.3.1 gives an overview of the parameters of our system and their significance. Sections 9.3.2 and 9.3.3 explain the experiments for the two respective methods and their optimal parameters. Section 9.3.5 compares the two methods and gives conclusions on the experimental evaluation.

9.3.1 Parameters and Dependencies

Our query methods are dependent on a number of parameters, like the number of interest points collected, the sizes of the interest regions, thresholds etc. We performed experiments with different values for each parameter and compared the results. However, for reasons of computational complexity we did not perform experiments on all different combinations of different values for the parameters. Instead we only tested different values for one parameter at a time. I.e. we assigned

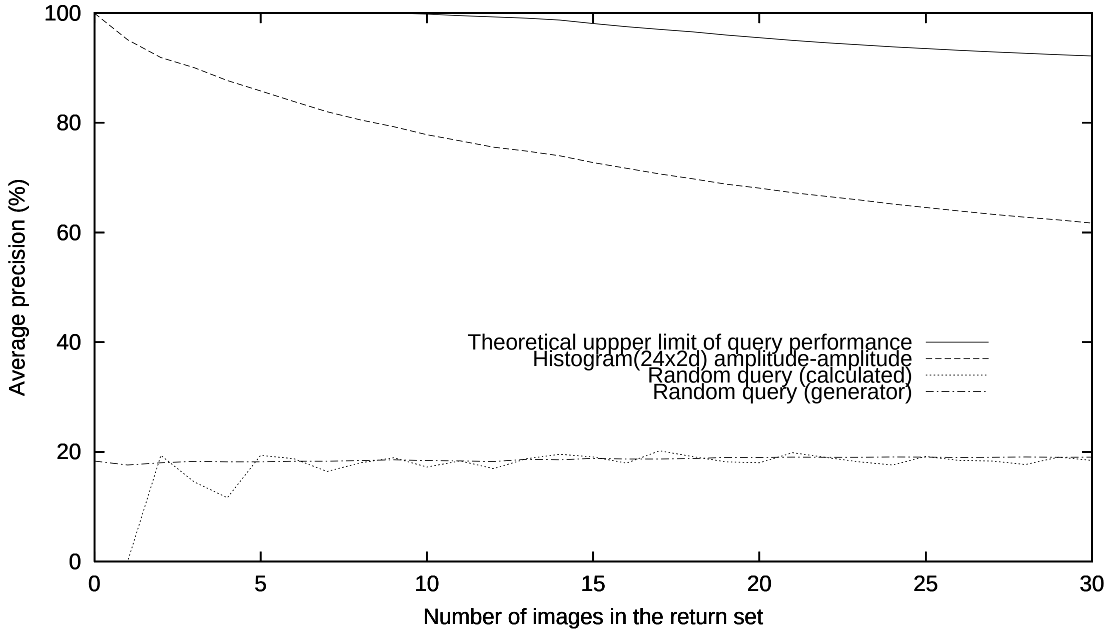


Figure 9.2: DB 1 - The best algorithm: The 2D histogram set method storing amplitudes in comparison with the *limits of query performance* (see chapter 9.2.2).

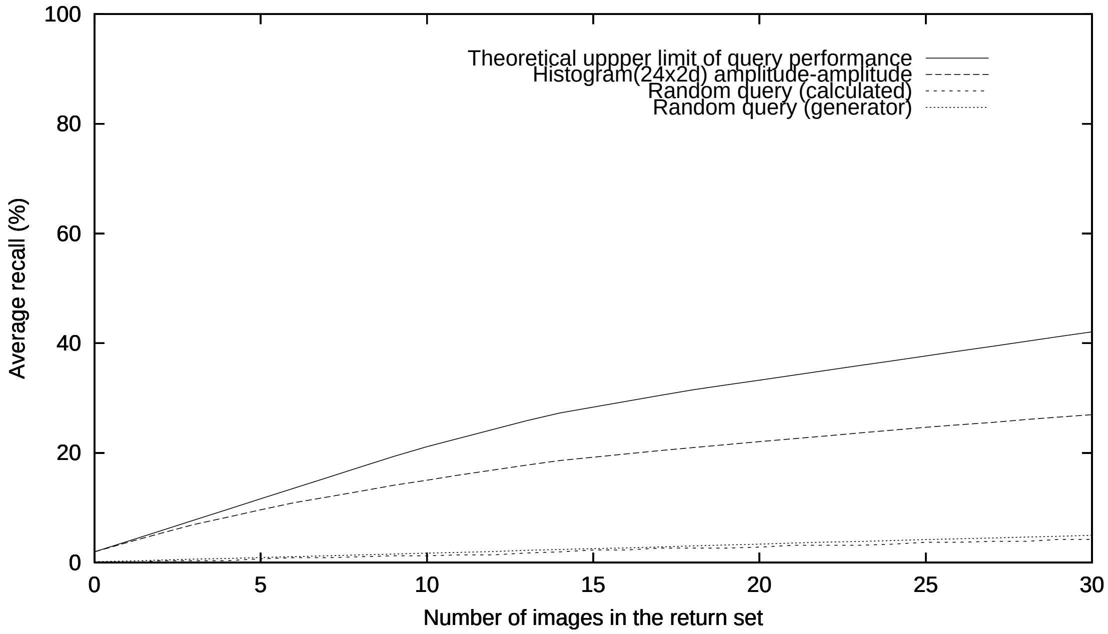


Figure 9.3: DB 1 - The 2D histogram set method in comparison with the limits of query performance (see chapter 9.2.2). *Recall* is displayed.

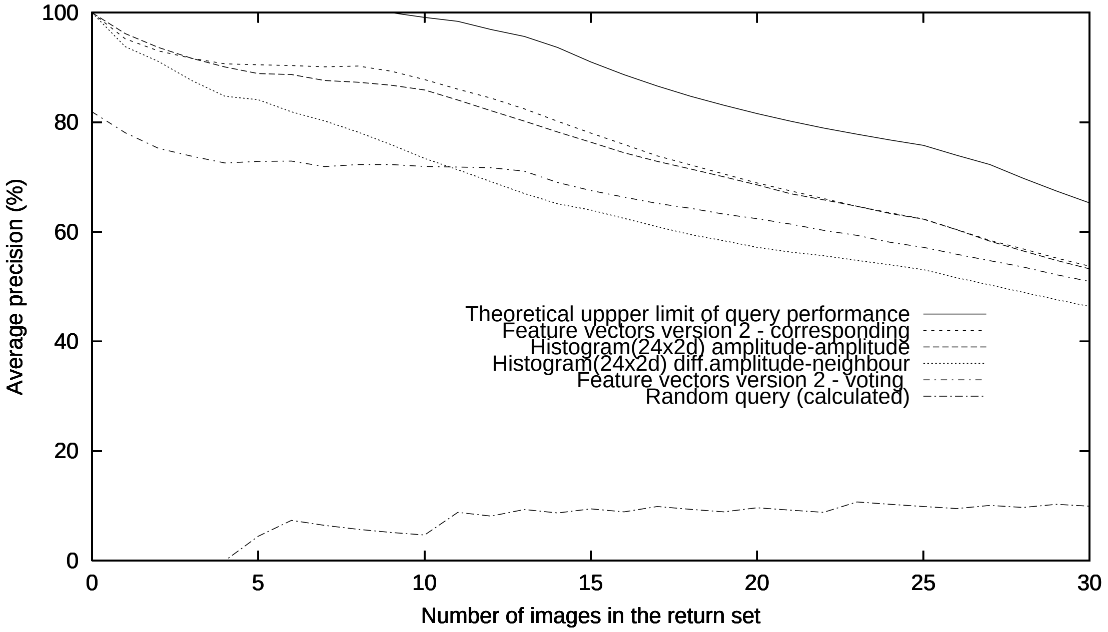


Figure 9.4: DB 2 - Different algorithms and the limits of performance - precision

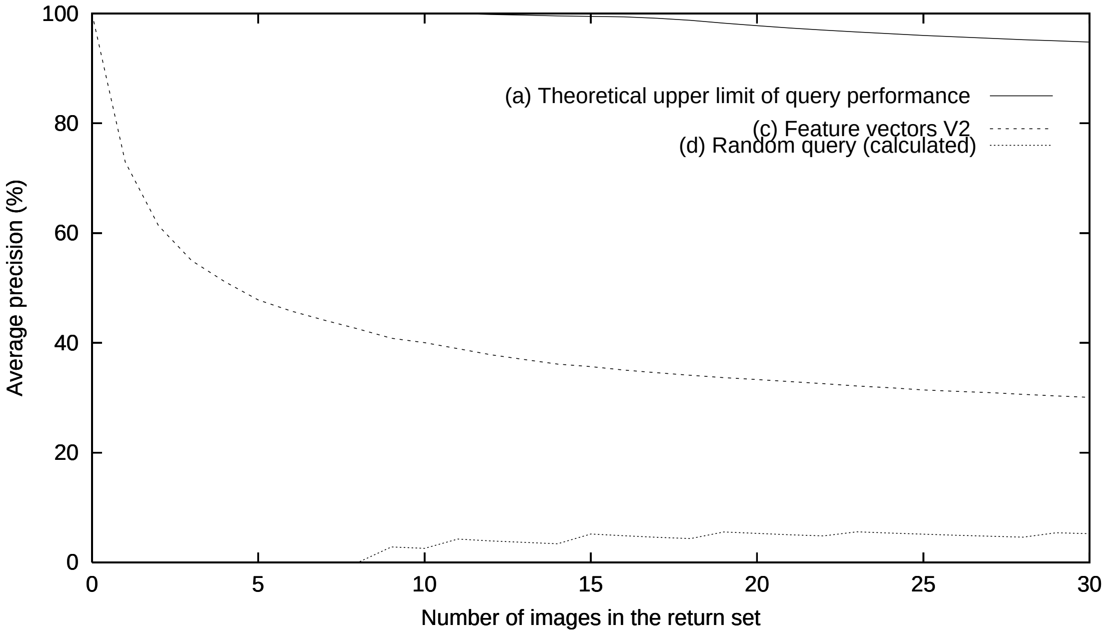


Figure 9.5: DB 3 - The feature vectors set based algorithm with the limits of performance - precision

fixed values to all other parameters, and performed experiments with a range of values for the tested parameter.

Table 9.4 shows the standard values we used for our parameters for all different methods if not specified otherwise. There are two different values for the number of interest points, because we used 100 interest points for experiments with feature vectors set representations and 200 interest points for experiments with histogram based representations. The two histogram dimensions correspond to the two different representations (Amplitudes $\times$ amplitudes and differences of amplitudes $\times$ neighbourhood ranking).

Parameter	Description	Standard value
N	Number of interest points	100/200
-	Interest region size	32×32 pixels
S	Number of scales of the Gabor filter bank	3
K	Number of orientations of the Gabor filter bank	8
n	Number of neighbours for the n-nearest neighbour search	48
t	Threshold for the voting algorithm	1.0
-	Histogram dimensions	$8 \times 8/8 \times 16$

Table 9.4: Standard parameter values used during the experiments

The following parameters are present in all of our image representations and query methods:

The number of interest points The number of interest points is an essential parameter of the indexation system. It determines how many interest regions are taken from the images and therefore also how much area of the image is covered by the description. Indexation speed and for some methods also query speed strongly depend on this parameter, so we are interested to keep it as low as possible (See section 9.1).

The region size The region size determines how much area is covered by the features collected on an interest point. The larger we set the area, the less sensitive the algorithm is to shifts of the interest point locations. On the other hand, the larger we set the size of the area, the less descriptive the features will be. The frequency spectrum of a whole image does not necessarily give lots of information about the texture contents of images, since most images contain more than one texture.

The number of scales of the Gabor filter bank This parameter determines how sensitive the Gabor filter bank is to changes of frequency.

The number of orientations of the Gabor filter bank This parameter determines how sensitive the Gabor filter bank is to changes of orientation.

9.3.2 Representation by Feature Vector Sets

In this section we explain the results of our experiments using the query methods based on the image representation by sets of feature vectors described in chapter 6. We describe the experiments we conducted to optimize the algorithms. The values of the following parameters have to be optimised:

- The distance formula
- The threshold value
- The number of interest points
- The interest region size

The distance formula

We performed experiments with three different distance formulas: The voting algorithm introduced by Schmid and Mohr (See Section 6.4.1) and the two distance measures for our method searching corresponding interest points using equations 6.4 and 6.5. The curves for the voting algorithm compared to the unweighted distance measure (Equations 6.4) are presented in Figures 9.6 and 9.4 for the image databases 1 and 2 respectively.

The figures show, that the performance of the algorithm searching corresponding interest points is superior to the voting algorithm. However, more striking is the fact, that for both databases the curve of the voting algorithm does not only show a different performance. It is also shaped differently than the other performance curves. If we compare the curves of the different algorithms in Figure 9.6, then we see that most of them are more or less parallel. Even the curves for the histogram based methods are almost parallel to the feature vectors curves using corresponding interest points. On the other hand the curves for the voting distance show a different progression. The performance drops quickly at the beginning, but with growing sizes of the result set the algorithm converges to the same performance as the algorithm searching corresponding points.

The explanation of the poorer performance of the voting distance can be explained with different facts. Firstly, in our test database for most similar image pairs one interest point of the query image corresponds to only one point in the database image. The second and probably more important explanation is the behaviour of feature vectors on flat or almost flat interest regions taken from the background of the images. A feature vector corresponding to one of these regions has a small distance compared with any other feature vector of this group of flat

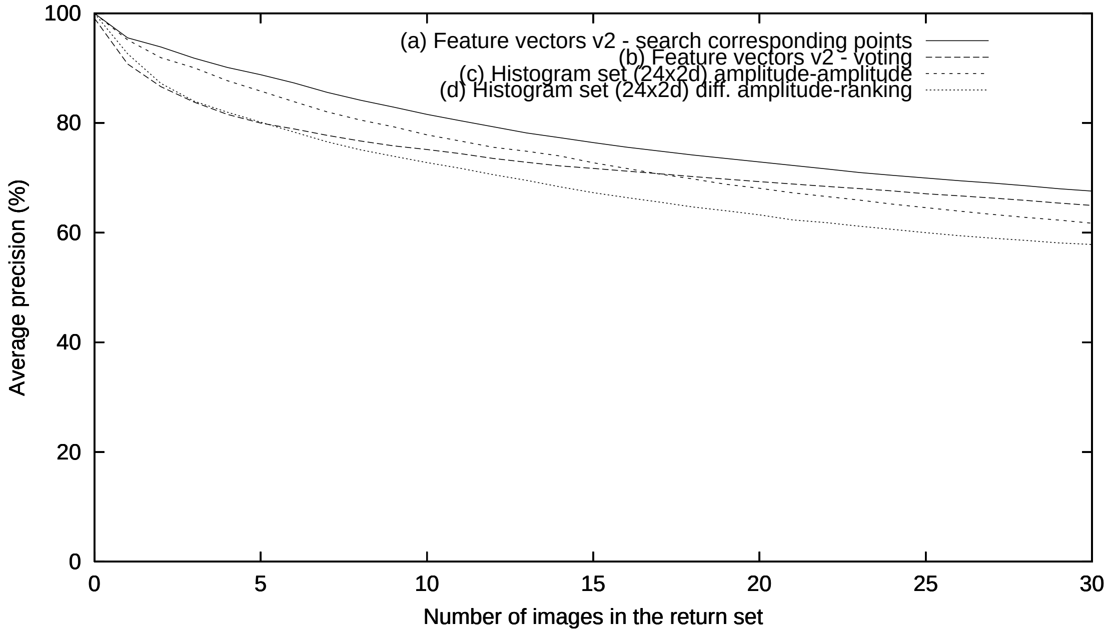


Figure 9.6: DB 1 - Comparison of the *different retrieval algorithms* for this work. For all methods the Haar transform interest point detector has been used. (a) Feature vectors on interest points. Comparison using the method of searching corresponding interest points (Equation 6.4). (b) Feature vectors on interest points. Comparison using voting. (c) The ordered set of two dimensional histograms storing the amplitude $\times$ amplitude distribution. (d) The ordered set of two dimensional histograms storing differences of amplitude $\times$ neighbourhood ranking.

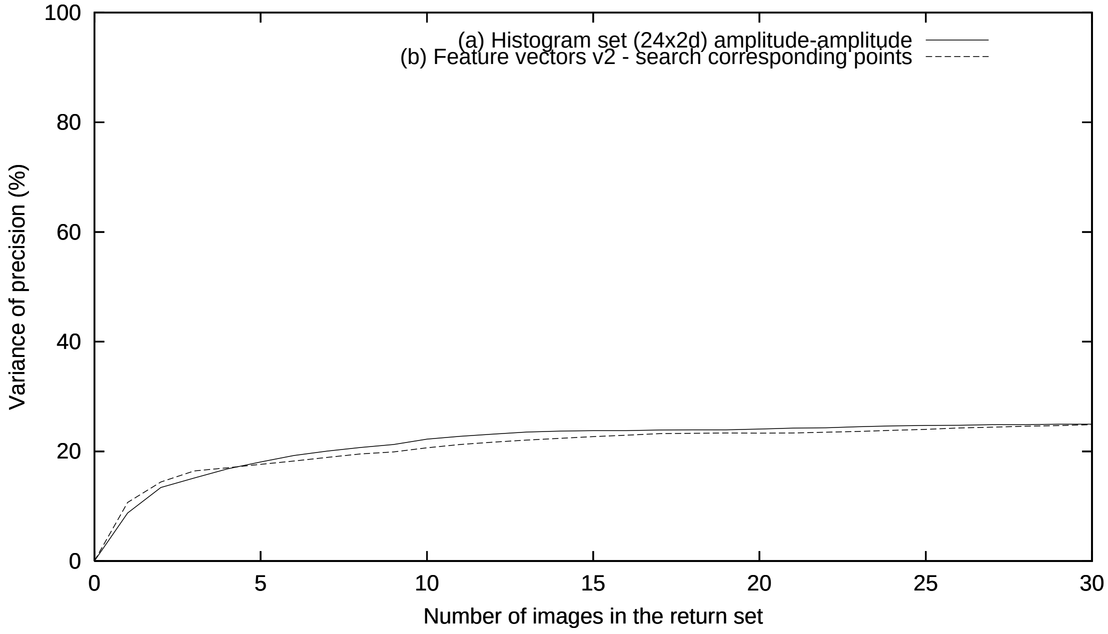


Figure 9.7: DB 1 - The results of the different methods - the variance of precision is plotted instead of the average precision.

regions. During the algorithm searching corresponding interest points these feature vectors are eliminated one by one and do not "harm" the distance function very much. Considering the voting algorithm on the other hand, one of these background feature vectors will match against possibly all other background feature vectors, which produces an enormous amount of votes. If we examine e.g. the comparison of two images, where each of them has 10 interest points taken from the background among all their 100 points each, then the comparison step produces approximately 100 votes, whereas the other votes will be in the range of 50 votes in the average case. I.e. that in this case $\frac{2}{3}$ of the votes correspond to the homogeneous background. This low relation between representative votes and unrepresentative background votes causes a worse query performance.

We performed experiments for the two different distance equations of the method searching corresponding interest points. From now on we will refer to the method defined by equation 6.4 as unweighted distance and to the method defined by equation 6.5 as weighted distance. Figures 9.8 and 9.9 show the performance curves for the two different distance measures for two of our test databases. The performance is almost equal for both distances within the order of statistical uncertainty. Nevertheless we believe, that the weighted distance measure works better than the unweighted measure on bigger test databases (See section 6.4.2). This fact has not yet been confirmed by experiments, since we did not perform any experiments with databases of sufficient sizes yet.

The threshold value

Unlike the histogram based algorithms the query methods for the feature vector set representations are equipped with a threshold value t which needs to be set. We conducted experiments with different values for this threshold: $t = 0.7, 1.0, 1.6$ and 2.0. The figures 9.11, 9.12 and 9.13 display result curves for the three test image databases using different threshold values applied to the method of searching corresponding interest points (Equation 6.4). As we can see, the query performance seems to be dependent on this value. One of the reason is the distance formula of the algorithm (equation 6.4). The discretisation of the distance values depends of the number of collected interest points, which was set to $N = 100$ in our experiments. Hence, there are at most $2N$ different values for the distance between two images. If the threshold values are too low or too high, then the distribution of distance values will shift to the respective end of the range of possible values, where the values cannot be distinguished.

A priori this fact is not very convenient. If a method depends on the correct settings of a parameter, then we need to find a solution how to find this parameter and under which circumstances it changes. However, a closer look to the results for all 3 image databases relativates this judgement. The optimal threshold value seems to be equal for all 3 databases, we set it to 1.0. We conclude, that the parameter is not strongly dependent on the input images, i.e. it is possible to

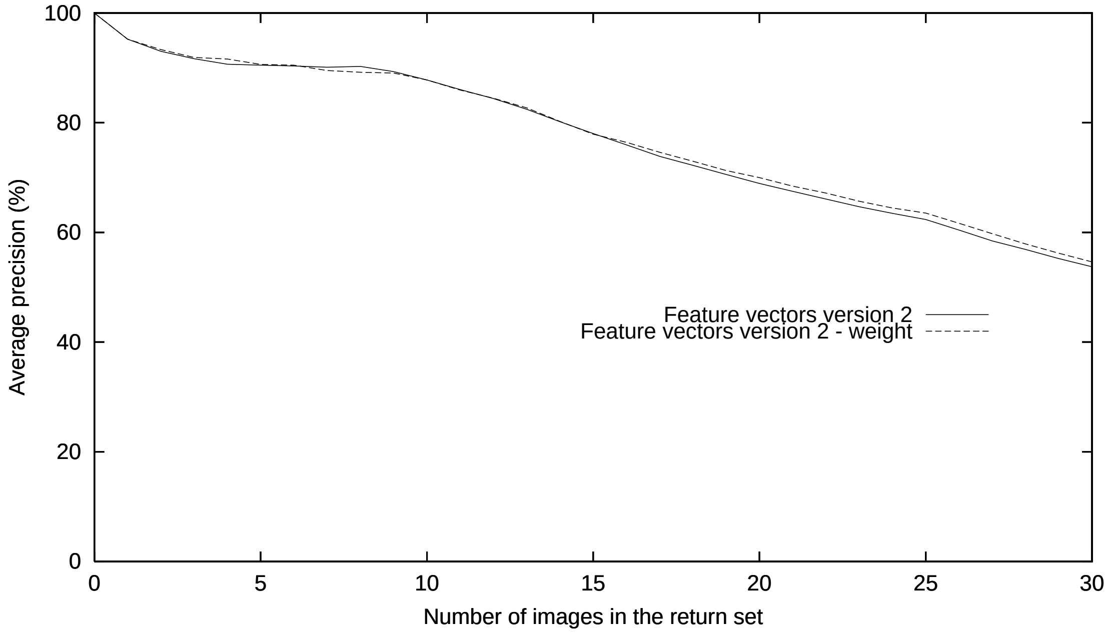


Figure 9.8: DB 2 - Feature vectors on interest points. Comparison using the method of searching corresponding interest points and *two different distance formulas*: (a) The distance calculation is based on the numbers of corresponding interest points found (Equation 6.4). (b) For the distance calculation not only the number of corresponding interest points is used, but also the distances between the corresponding feature vectors (Equation 6.5). Region size is 32×32, 100 interest points were collected.

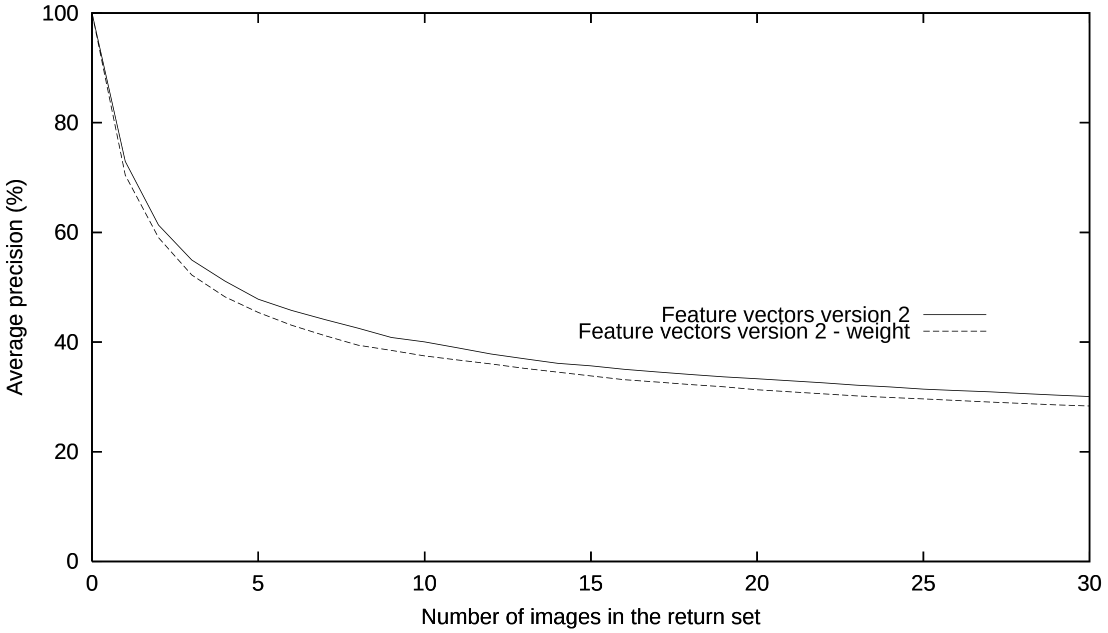


Figure 9.9: DB 3 - Feature vectors on interest points. Comparison using the method of searching corresponding interest points and *two different distance formulas*: (a) The distance calculation is based on the numbers of corresponding interest points found (Equation 6.4). (b) For the distance calculation not only the number of corresponding interest points is used, but also the distances between the corresponding feature vectors (Equation 6.5). Region size is 32×32, 100 interest points were collected.

assign it a fixed value before hand.

Another surprising result concerning the optimal threshold value is it's actual value, or the position of this value in the range of the possible distances between vectors in feature space. Figure 9.10 shows this distribution of the vector distances in a normalised histogram plot, which has been created by calculating and subsampling all possible distances of vectors in the first test image database and placing them into a one dimensional histogram. The x axis specifies the distance between two vectors (Equation 6.3), the y axis specifies the count of vector pairs having this distance. The histogram has been normalised, so that the sum of all bins is equal to 1.

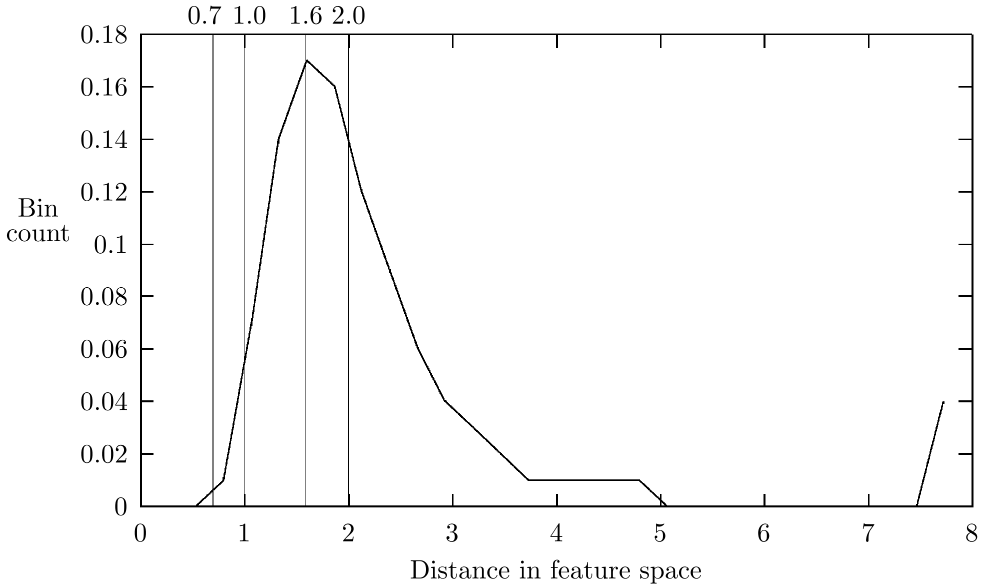


Figure 9.10: The distribution of the feature vector distances as normalised histogram plot

We used the threshold values $t = 0.7, 1.0, 1.6$ and 2.0 in our experiments, they are displayed in Figure 9.10 as thin vertical lines. We note, that the optimal threshold of $t = 1.0$ is situated near the lower end of the range of found distances. This low threshold has the effect, that the number of corresponding interest points is low. This fact can be seen e.g. in Figure 6.5, where in two rather similar images only few of the 100 select interest points are actually found as corresponding.

Number of interest points

Figure 9.14 depicts query performance curves for the feature vector set method with different counts of collected interest points. We used counts of $30, 50, 70$ and 100 points. The reason for the limitation of the number of points is the strong

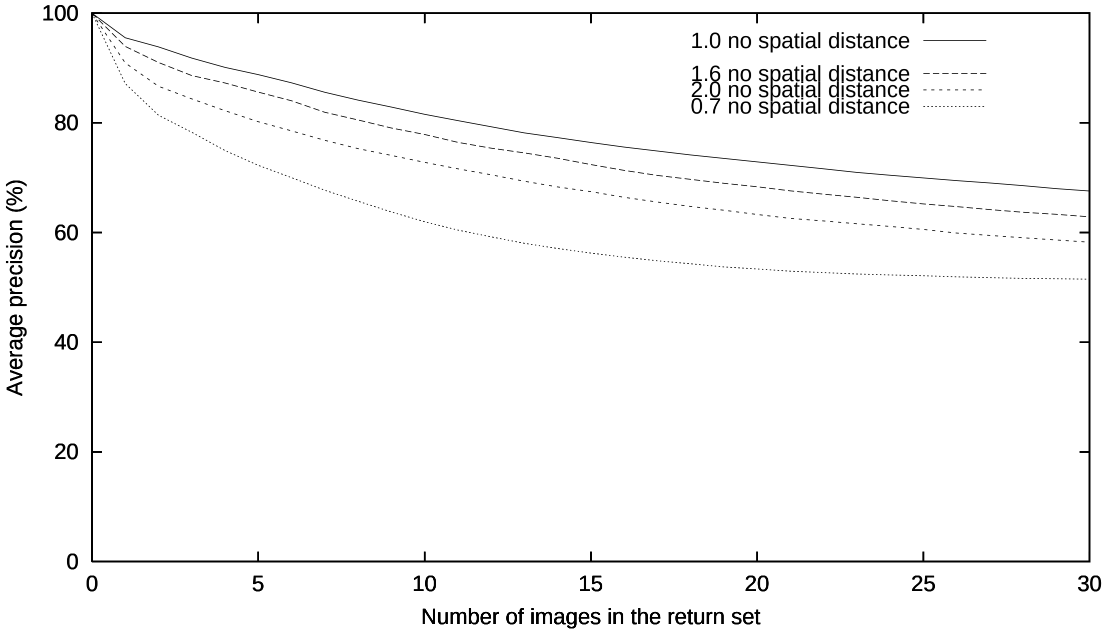


Figure 9.11: DB 1 - *Different thresholds* applied to the method using feature vectors on interest points, version 2. Comparison using the method of searching corresponding interest points (Equation 6.4). Region size is 32×32, 100 interest points collected.

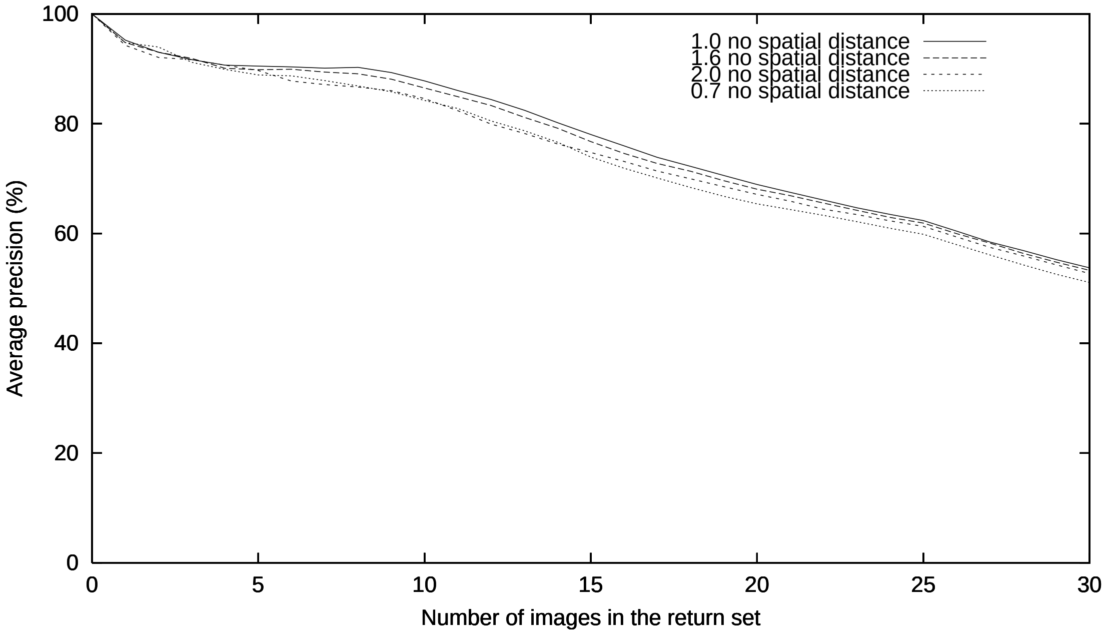


Figure 9.12: DB 2 - *Different thresholds* applied to the method using feature vectors on interest points, version 2. Comparison using the method of searching corresponding interest points (Equation 6.4). Region size is 32×32, 100 interest points collected.

dependency of the method on the number of points. It is computationally too expensive to perform the algorithm with 200 interest points collected on each image.

The query performance of the feature vector set based method does depend on the number of interest points. This can be explained by the nature of the distance algorithm. Since the distance of two images is based on the comparison of points, the importance of a single point is very high. Hence, increasing the query performance of the feature vector set based method by increasing the number of interest points N is possible but limited due to the computational complexity of the algorithm ($O(N^3)$).

Interest region size

We performed experiments using interest region sizes of 64×64 pixels, 32×32 pixels and 16×16 pixels to check if they influence query performance. From the computational point of view the size of the interest regions strongly determines indexation speed, but it has almost no effect on the query speed. Nevertheless we want to keep it as low as possible. Figure 9.15 shows query performance curves using the feature vector set based method and the three different region sizes. As we can see the query performance is not significantly affected by the change of the region sizes.

The interest point operator

For our experiments we used different interest operators, which we described in chapter 3: Two different implementations of the Harris corner detector, two different versions of the Loupias wavelet based interest point detector based on the Haar and the Daubechie wavelet respectively, and the Jolion multiresolution contrast based interest point detector. To evaluate the dependency of our algorithms on the choice of the interest point operator, we additionally implemented an "interest point operator" selecting a fixed number of random points in an image.

The results of our experiments are displayed in Figure 9.16 for our first test database. As we can see, the differences in performance between the various detectors are not big. The performance of the algorithms in experiments where the random interest point detector is used is not as good as the performance of the other interest operators, but still surprisingly good. The differences are in the order of the statistical uncertainty of the measurement of the algorithms. We conclude, that the performance of our algorithms only weakly depends on the locations of the interest points, i.e. on the choice of the interest point detector. This fact can be explained by the richness of our image features. The feature data collected on the interest points has enough descriptive power.

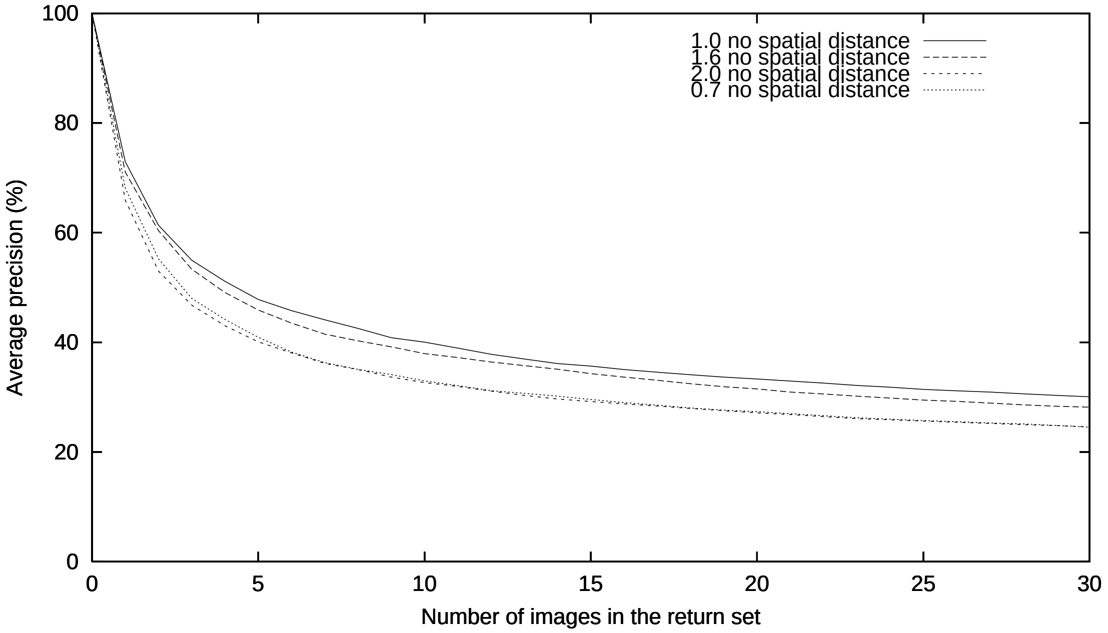


Figure 9.13: DB 3 - *Different thresholds* applied to the method using feature vectors on interest points, version 2. Comparison using the method of searching corresponding interest points (Equation 6.4). Region size is 32×32, 100 interest points collected.

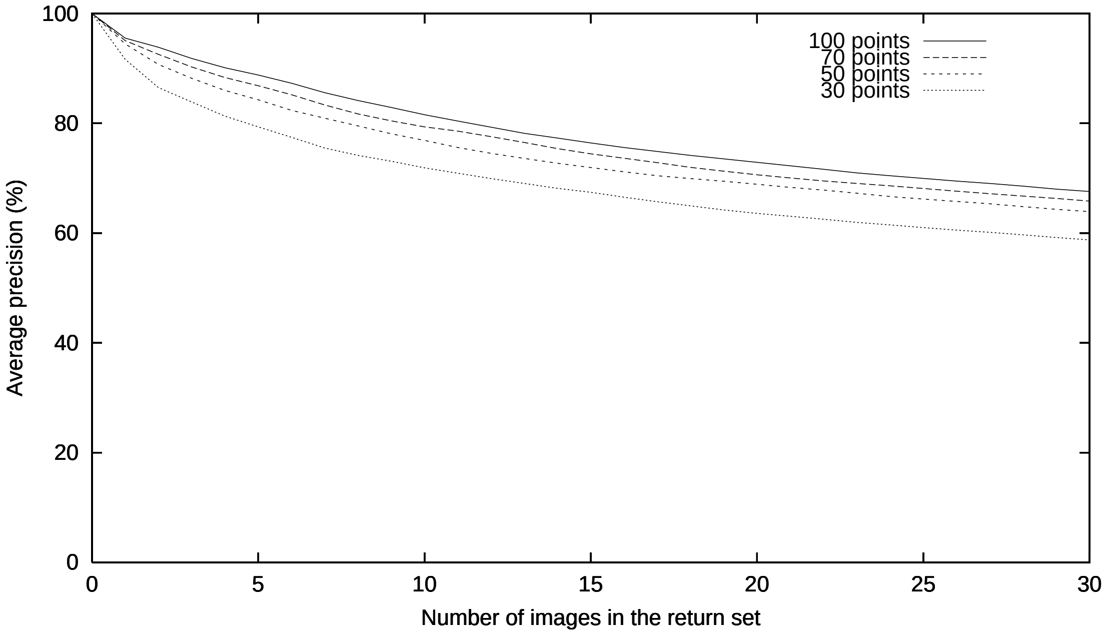


Figure 9.14: DB 1 - *Different counts of collected interest points* applied to the method searching corresponding interest points (Equation 6.4). Region size is 32×32, Haar transform detector.

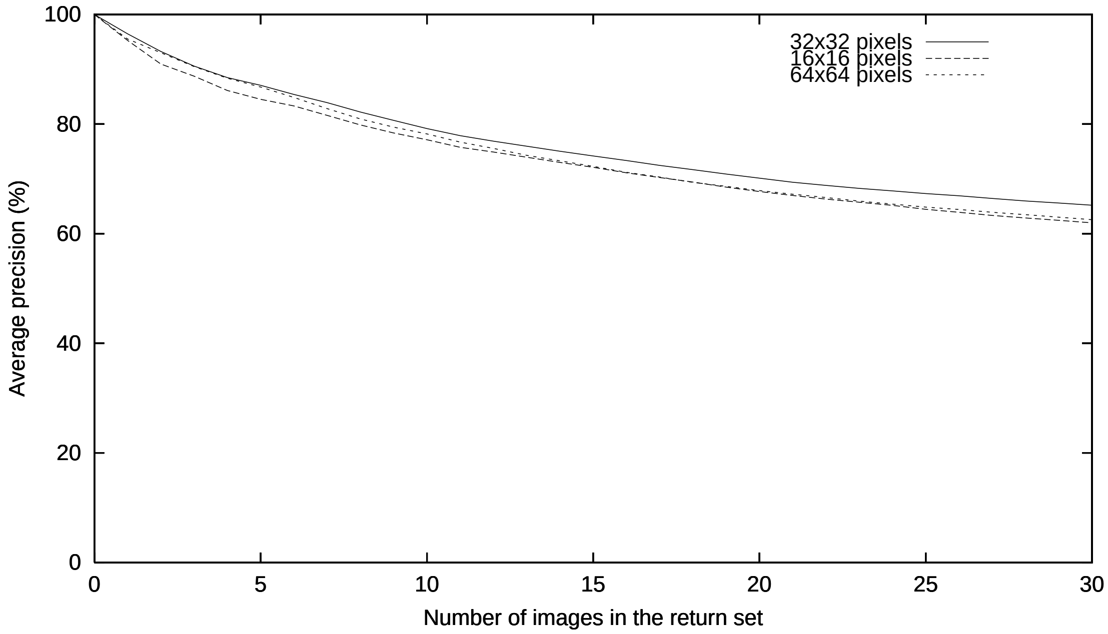


Figure 9.15: DB 1 - *Different interest region sizes* applied to the method searching corresponding interest points (Equation 6.4). Region size is 32×32, multiresolution contrast based IP detector.

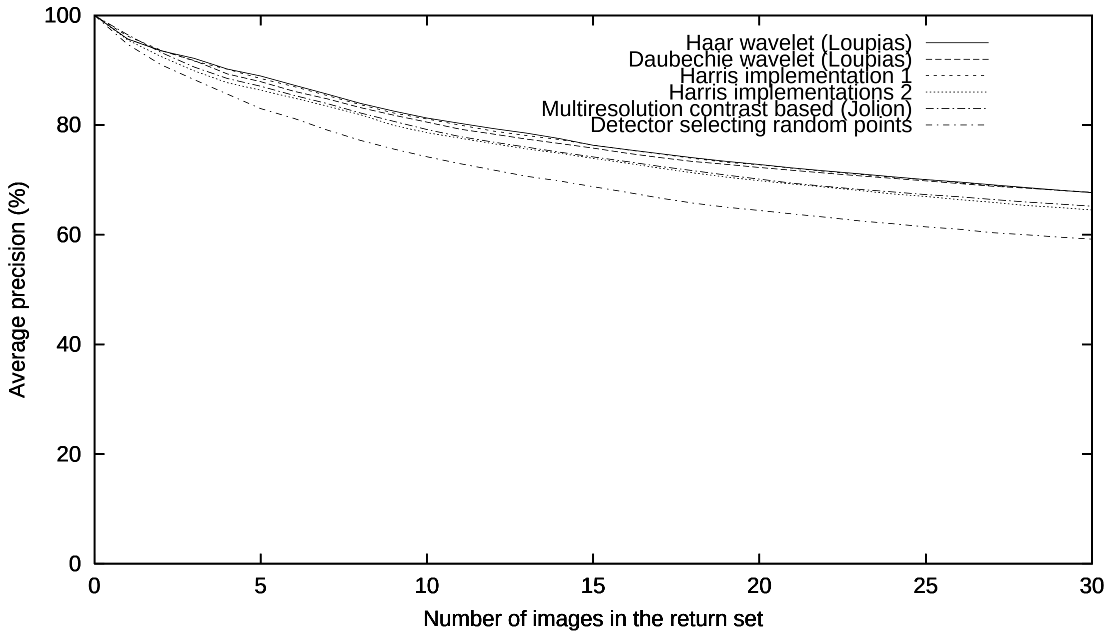


Figure 9.16: DB 1 - The results for the method searching corresponding feature vectors), using *different interest point detectors*. Region size is 32 × 32, 100 interest points were collected.

9.3.3 Representation by Histogram Sets

This section describes the experiments that have been conducted using the two histogram set representations described in chapter 7. As for the feature vector methods, we optimised the parameters of the algorithms to obtain the optimal query performance.

The histogram dimensions

An important parameter of the image representation is the dimensions of the histograms, i.e. the number of bins, and the borders, i.e. the interval of the raw data which is represented by the histograms. We designed the histogram borders according to the distribution of the feature data. Table 9.5 shows the histogram borders and dimensions for the two different representations used during our experiments.

Parameter	Amplitude $\times$ Amplitude	Amplitude $\times$ Ranking
x-bin count	8	8
y-bin count	8	16
x-min value	7	-6
y-max value	12	6
x-min value	7	1
y-max value	12	n = number of neighbours

Table 9.5: The histogram dimensions used during the experiments

In the case of the amplitude-amplitude histogram the values on the x and on the y axis are logarithmic amplitude values. The given borders have been calculated by decreasing step by step the borders of an accumulated histogram, which has been computed by summing up all histograms of our first test image database, until the amplitude data is best covered by the interval of the histogram. The same has been done for the x axis of the second histogram type, which stores differences of amplitudes in it's x axis. The y axis of the second histogram type, which stores the neighbourhood ranking of the n-nearest neighbour search, has trivial borders, which we did not need to calculate.

The histogram dimensions have been found by experiments. Like we already experienced with other parameters, the histogram dimensions influence the query speed, so we want to keep them as small as possible. However, too small histograms are not representative anymore, whereas too large histograms contain too many empty bins, which is not a good characteristic if we want to apply histogram distances.

The number of neighbours

The number of neighbours n for the n-nearest neighbour search determines how much influence of the spatial coherence of the feature data is represented in the histogram. However, the number has to be adjusted to the data. If the algorithm does not search enough neighbours, then the spatial coherence is under represented in the histogram. If the algorithm searches too many neighbours on the other hand, then the information about the spatial coherence is lost, since the found neighbours tend to be the same for points near to each other. We conducted experiments using 3, 6, 12, 24 and 48 neighbours. Surprisingly there is almost no difference in the query performance (Figure 9.17), although counts of $n = 12$ or even $n = 48$ are far too high to store any spatial information in the histograms, considering that $N = 200$ interest points are collected on each image. We conclude, that the additional information about the spatial coherence does not change the descriptiveness of the histograms.

The histogram distance measure

We also conducted experiments with the histogram distance measurements described in section 7.3. As can be seen in Figure 9.18 the best results are obtained with the Battacharyya distance. This has been confirmed by Huet and Hancock in their work as well [11].

The histogram representation

We introduced two different histogram based image representations. Representation 1 consists of a set of histograms storing amplitudes and representation 2 consists of a set of histograms storing differences of amplitudes and the ranking of the neighbours of the n-nearest neighbour search. We performed experiments using both of the two representations. Comparing the query performance of the methods applied to test image database 1 (Figure 9.6) and test image database 2 (Figure 9.4). We remark that the first method based on the amplitude distribution only performs better than the second one. We conclude, that the absolute amplitude information is more descriptive than the relative information. I.e. the information, which orientations and scales are present in the image is more descriptive than the information how much the amplitudes change in the spatial neighbourhood of the interest points.

The number of interest points

As for the feature vector approach we conducted experiments with different counts of collected interest points to optimise the algorithm for this parameter. Figure 9.19 depicts the experiments with numbers of $50, 100, 200$ and 300 points. For this algorithm it was possible use higher numbers of interest points, because unlike the

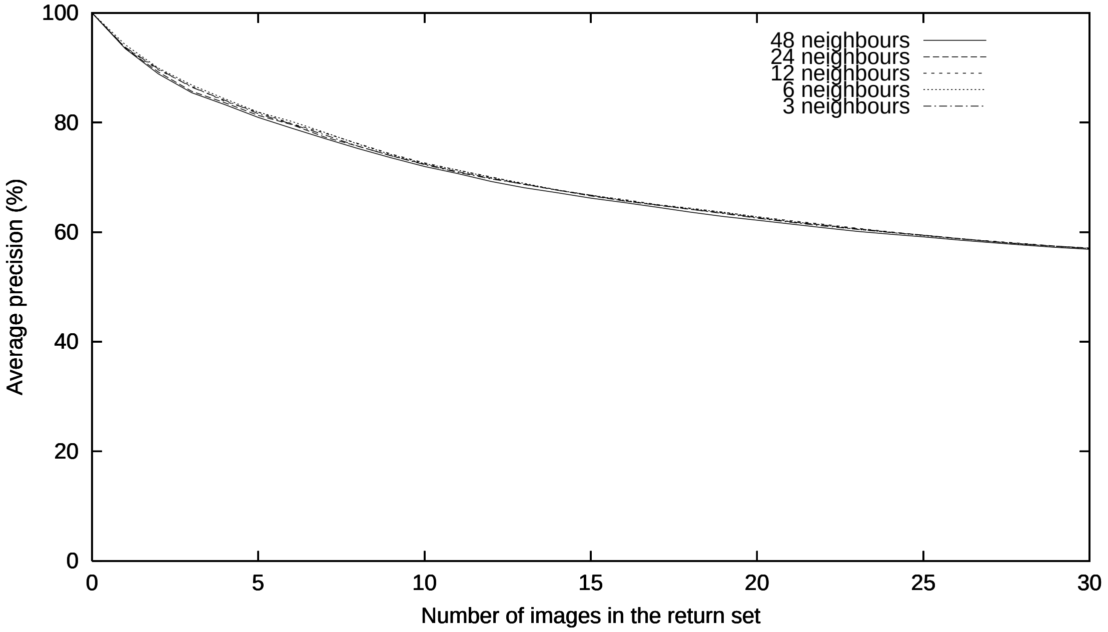


Figure 9.17: DB 1 - The method using ordered set of histograms (storing amplitude $\times$ amplitude), *different counts of neighbours* for the n-nearest neighbour search. Region size is 32×32, Haar transform detector.

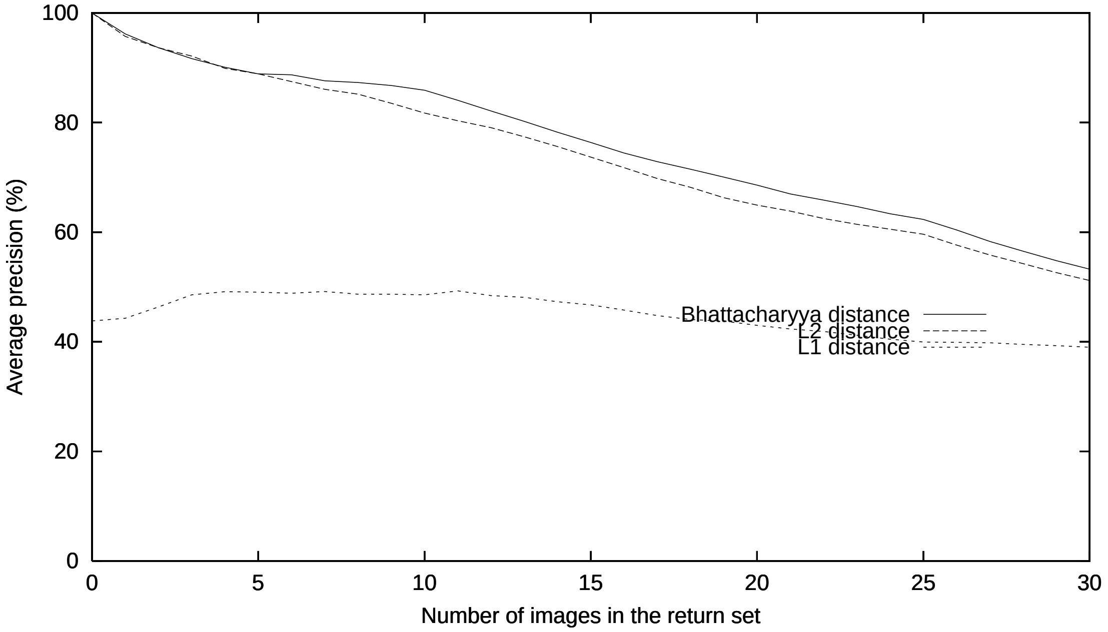


Figure 9.18: DB 2 - The results for the ordered set of histograms (storing amplitude $\times$ amplitude), *using different histogram distances*. Region size is 32×32, 200 interest points were collected.

feature vector set method the query speed does not depend on this parameter. As we can see the performance of the histogram based algorithm is weakly dependent on the number of interest points. The differences in the query performance are within the statistical uncertainty of the measurement algorithms.

For the histogram based methods the count of interest points is not as important. If enough area of the image is covered by interest regions, and if the histograms are filled enough — i.e. the number of non-zero bins of the normalised histogram is sufficiently high — then increasing the number of interest points does not increase query performance.

The interest region size

Similar to the feature vector set method we performed experiments with different interest region sizes. We used sizes of 64×64 pixels, 32×32 pixels and 16×16 pixels. Figure 9.20 shows query performance curves using the histogram set based query method (histograms sets storing the amplitude distribution) and the three different region sizes. As for the feature vector set method, the query performance is not significantly affected by the change of the region sizes.

The interest point operator

In our experiments we used the same interest point operators we already applied in the feature vector set methods. The results are displayed in Figure 9.21 for test database 1 and in Figure 9.22 for test database 2. The results are similar to the results of the experiments for the feature vector set methods: The differences in performance between the various detectors are not big. However, surprising is the good performance of the algorithms in experiments where the random interest point detector is used. The performance of the random points operator is equal to the performance of the other operators.

We already explained the weak dependency of our algorithms to the choice of the interest operator with the richness of our image features. However, the histogram based approach is even less sensitive to the choice of the interest operator than the feature vector set approach. This can be explained by the fact, that we do not compare single feature vectors, i.e. single interest points, but distributions of interest points. The feature data collected on the interest points has enough descriptive power, which is not improved by a stable interest point detector choosing points "appropriate" for our type of features.

9.3.4 Comparison of the Methods per Query Image

The query curves show the performance of the different query methods as the precision averaged for all query images. We don't get any information which query method works well with which type of image. To produce this kind of information

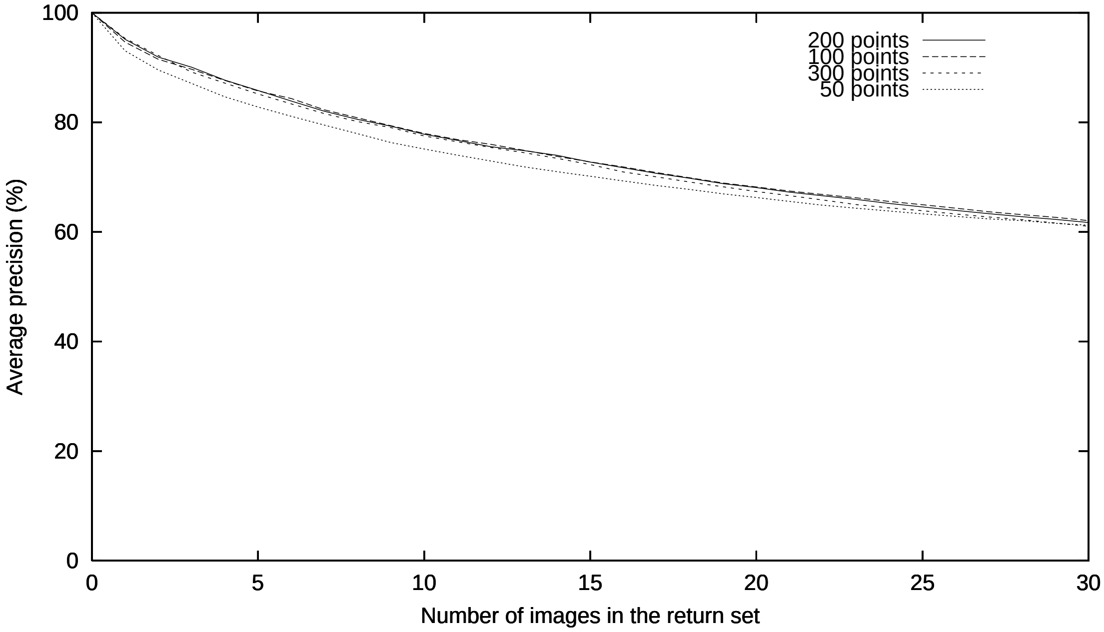


Figure 9.19: DB 1 - *Different counts of collected interest points* applied to the method using ordered set of histograms (storing amplitude $\times$ amplitude). Region size is 32×32, Haar transform detector.

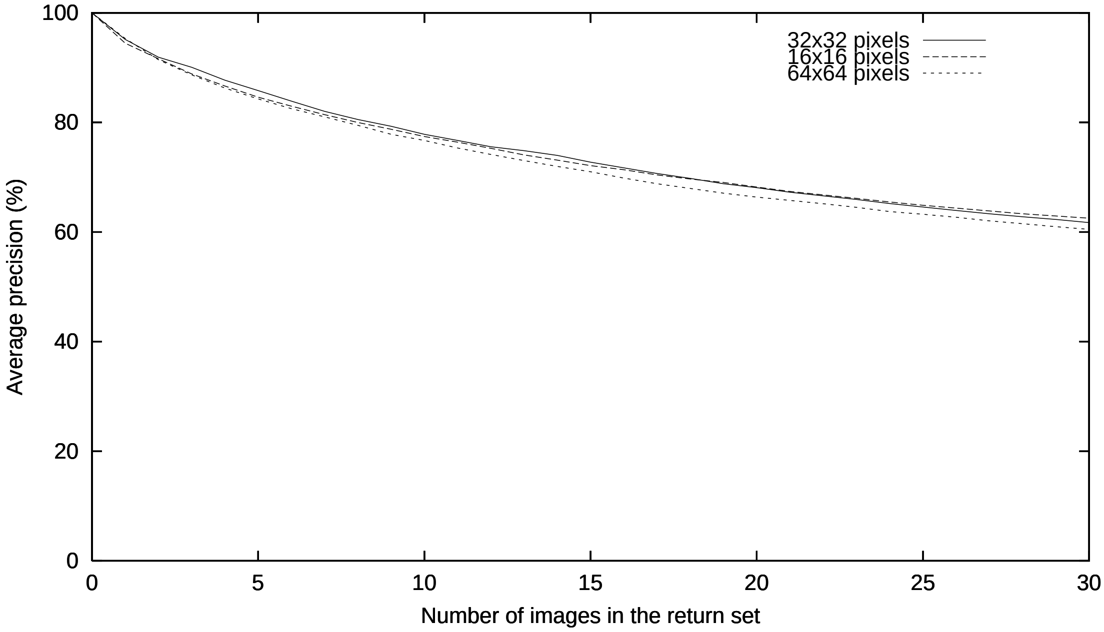


Figure 9.20: DB 1 - The results for the ordered set of histograms (storing amplitude $\times$ amplitude), using *different interest region sizes*. Haar transform detector, 200 interest points collected.

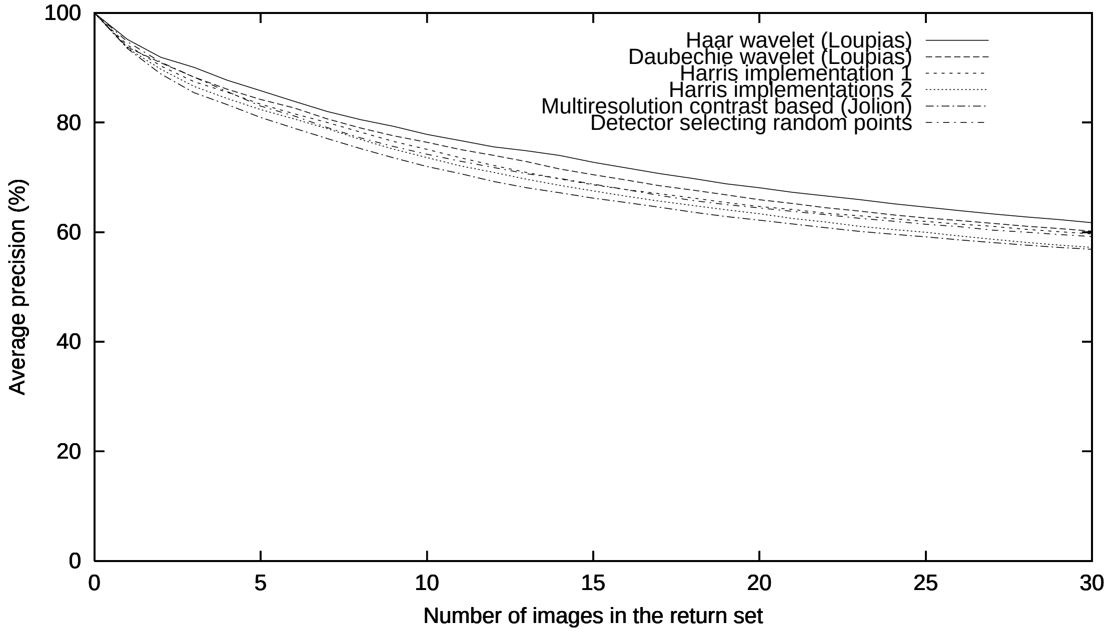


Figure 9.21: DB 1 - The results for the ordered set of histograms (storing amplitude $\times$ amplitude), using *different interest point detectors*. Region size is 32×32, 200 interest points were collected.

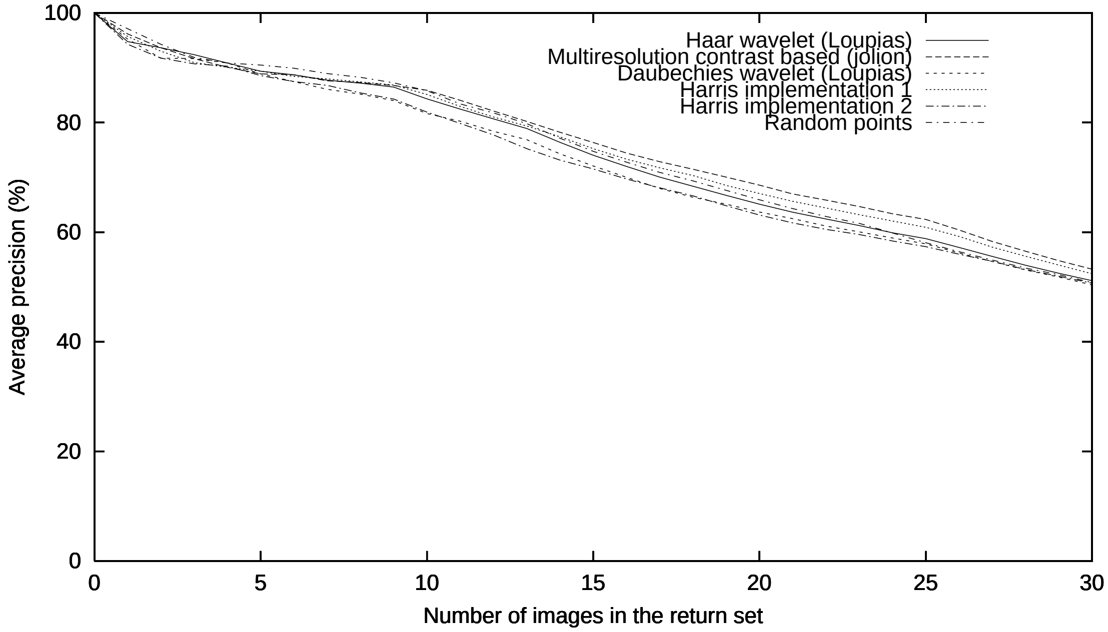


Figure 9.22: DB 2 - The results for the ordered set of histograms (storing amplitude $\times$ amplitude), using *different interest point detectors*. Region size is 32×32, 200 interest points were collected.

we created statistics at a lower level of detail, i.e. the level of single query images. For each query image and each query method we calculated the query precision for a return set of 10, 20, 30 and 40 images, and the average precision for all return sets with sizes between 1 and 50 images. Then we compared the precision values for all methods for each query image. The query method which is best for this query image gets a vote. Continuing this process for all query images we get several statistics for each size of the return set how many queries work best for each method.

Algorithm / Images in the return set	10	20	30	40	avg. 1-50
Histogram set amplitude $\times$amplitude	347	248	235	220	201
Histogram set diff. amplitudes $\times$ ranking	93	122	126	142	126
Feature vectors version 2	129	199	208	207	242

Table 9.6: Results for different sizes of the result set

It can be seen, that with growing size of the return set the histogram set method storing amplitudes looses performance against the growing performance of the feature vector method. However, more important is the information, which query method is preferable for which type of image. So we classified the same information according to the image clusters in the following two tables. The rows represent the different query methods, the columns the different image clusters. An entry in the table specifies how many images of this cluster had the best results for this query method. The following tables were produced for 10 images in the return set and the average value of 1 to 50 images in the return set respectively.

Algorithm / cluster	1	2	3	4	5	6	7	8	9	10	11	total
Histogram set amplitude $\times$amplitude	3	11	1	12	14	3	26	25	79	81	92	347
Histogram set diff. amplitudes $\times$ ranking	5	0	0	3	0	5	3	8	0	49	20	93
Feature vectors version 2	2	0	13	0	1	11	3	3	7	26	63	129

Table 9.7: Results: 10 images in the return set

The difference of the two tables is not small, which means that the ordering of the "good" results in the return set is different for the different methods.

9.3.5 Comparison and Conclusion

In our experiments we evaluated two query methods applied to the three different test databases described in chapter 8. As we can see, the algorithmic distance

Algorithm / cluster	1	2	3	4	5	6	7	8	9	10	11	total
Histogram set amplitude $\times$amplitude	0	10	1	7	14	1	0	25	43	49	51	201
Histogram set diff. amplitudes $\times$ ranking	5	0	0	6	0	6	0	6	14	67	22	126
Feature vectors version 2	5	1	13	2	1	12	32	5	29	40	102	242

Table 9.8: Results: Average value of 1 - 50 images in the return set

methods using the feature vector set representation perform slightly better then the statistical distance methods based on the histogram set representations. Considering, that the difference in query performance between the feature vector methods and the histograms is almost insignificant, then the relation between cost (query speed) and benefit (query performance) is significantly better for the histogram based methods.

The performance of our algorithms is only weakly dependent on parameters. Both algorithms are invariant to changes of the interest region size. The threshold t of the feature vector set algorithm can be fixed according to our experiments. The performance of the histogram based method does not depend on the number of the interest points collected.

As we already noted, the query performance of the feature vector set based method can be improved by increasing the number of interest points, but the amelioration is limited due to the computational complexity of the algorithm. Future work could be done to combine the feature vector set representation with hierarchical representations and methods [14] in order to reduce the complexity of the distance algorithm.

Chapter 10

Conclusion and Outlook

The main contribution of this thesis has been the description of several possibilities to realise a content based image retrieval system which uses texture similarity to calculate distances between images. We showed how to combine interest point detectors and the application of a Gabor filter bank to create descriptive image representations. We introduced two different query methods which used different image representations. We described an algorithmic method based on frequency descriptions of image regions and a statistical method based on histogram representations of the images. Both query algorithms give good results according to our test image databases. For indexation purposes we recommend the histogram based statistical method, because it needs much less computational efforts, whereas the performance decrease is statistically not significant.

Figure 10.1: Example Query

Our algorithms do not use any a priori model of the image contents. They have been evaluated and tested in various experiments using different databases of test images of various sources. Because we did not restrict the set of images supported to a specific group, the question arises, which type of queries can be handled by our methods.

The image representation introduced in this thesis holds a rough texture description of images. The similarity measure is able to distinguish groups of images of the same type, i.e. images having similar content without considering many details. Typical applications could be e.g. databases of television broadcast stations, which need to find screenshots of similar scenes or shots of the same telecast in a large set of television screenshots. Experiments with one of our test image databases which contains television screenshots, prove the good performance for this task.

Figure 10.1 shows a typical result set of a query against one of our test databases containing images of various types (portraits, drawings, screenshots of video sequences, textures etc.). The query image — a portrait — is displayed in the left upper position, the result image are ordered from left to right and from top to bottom. Our query algorithms are capable to retrieve the other portraits of the database. However, they are unable to recognise details of the images, i.e. in this case they do not retrieve the images showing the person on the query image on the first positions of the result set. Tasks like this need to be done by specialised recognition systems, which use specific image data to create image similarities designed for the application domain.

Other possible applications of our methods could be to use them as preprocessing steps to specialised image databases. Because our query methods are capable of retrieving images of the same type from a database containing images of various types they can be used to narrow the set of images to search for other methods which perform search algorithms specialised for a specific task. Considering the example described above, one possibility would be to search all portraits of a general database using our texture based method, and to apply a face recognition algorithm afterwards on the result set of our method.

The following future tasks are planned to deepen the experiences we made during this work:

- The integration of a structural component by combining our feature vector set based query method with attributed graph pyramids [13] [14]. The aim is reduce the complexity of the feature comparison step and to add a hierarchical component to the image similarity.

- Another task currently pursued is to join this texture based approach with methods based on colour, structure and shape into one weighted indexation system, which uses feedback of the user to determine the preferences and to recalculate the weights of the system (See Section 2.3).

Appendix A

Table of symbols

The following symbols have been used through out this work:

Symbol	Description
Database	
B	Number of images in the database
F	Number of images used as query images
N	Number of interest points collected on an image
Filters	
S	Number of scales of the Gabor filter bank
K	Number of orientations of the Gabor filter bank
$u0$	The x-coordinate in frequency space of the max. amplitude of a filter response
$v0$	The y-coordinate in frequency space of the max. amplitude of a filter response
Algorithms	
t	The threshold for the distance of feature vector sets.
n	The number of neighbours for the n-nearest neighbour. search
μ, ν	Feature vectors or histogram vectors
Measures	
$P(c)$	Precision of a query returning a result set of c images
$R(c)$	Recall of a query returning a result set of c images

Table A.1: Symbols used in this document

Appendix B

Screenshots

For demonstration purposes and to make tests of our algorithms easier we developed a query system with graphical interfaces. Screenshots of these tools for the X-window system and the web are displayed in Figures B.1 and B.2. The system is accessible via the world wide web at the following address:

`http://www.prip.tuwien.ac.at/Research/ImageDatabases/Query`

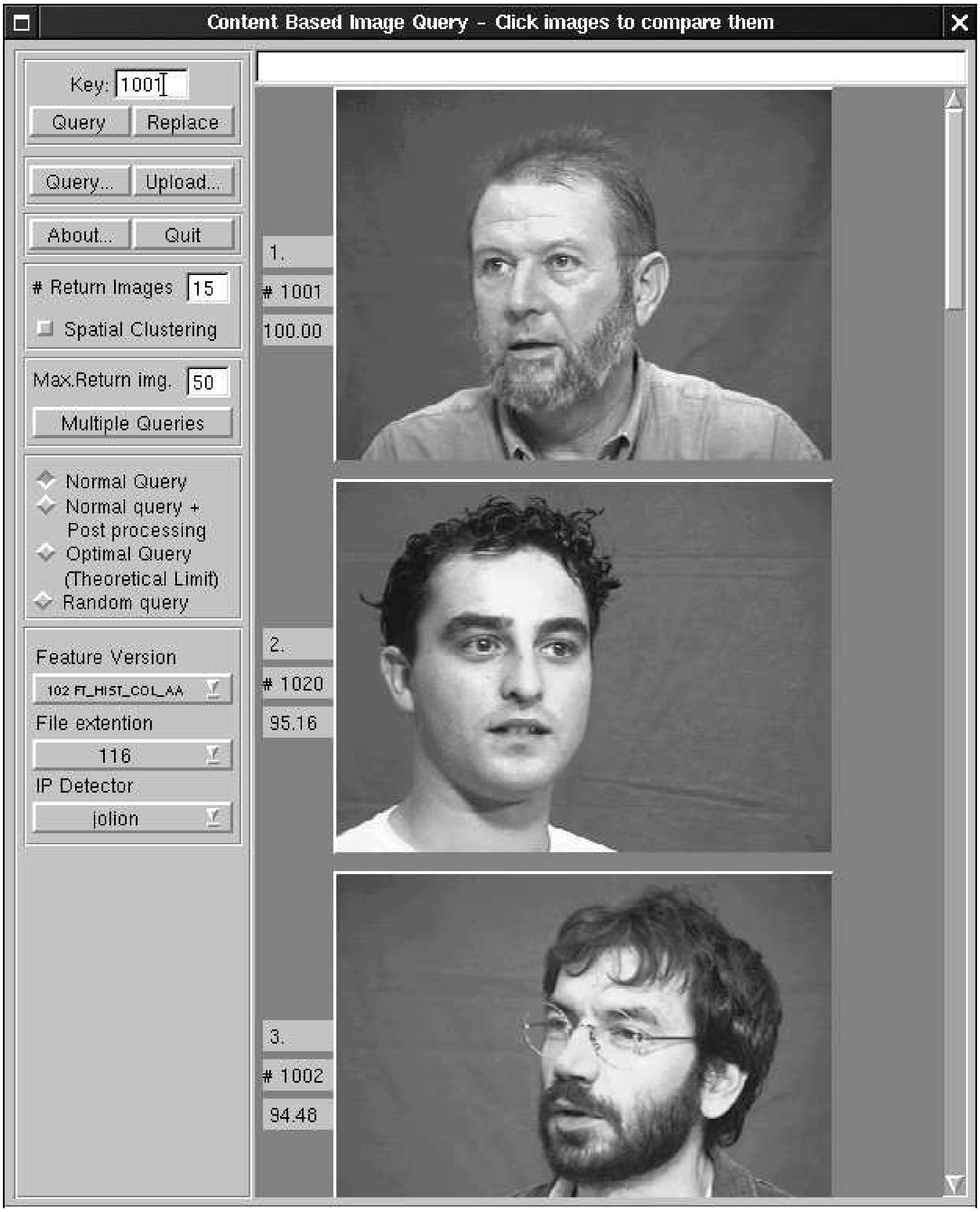


Figure B.1: The graphical Frontend for the X-window System

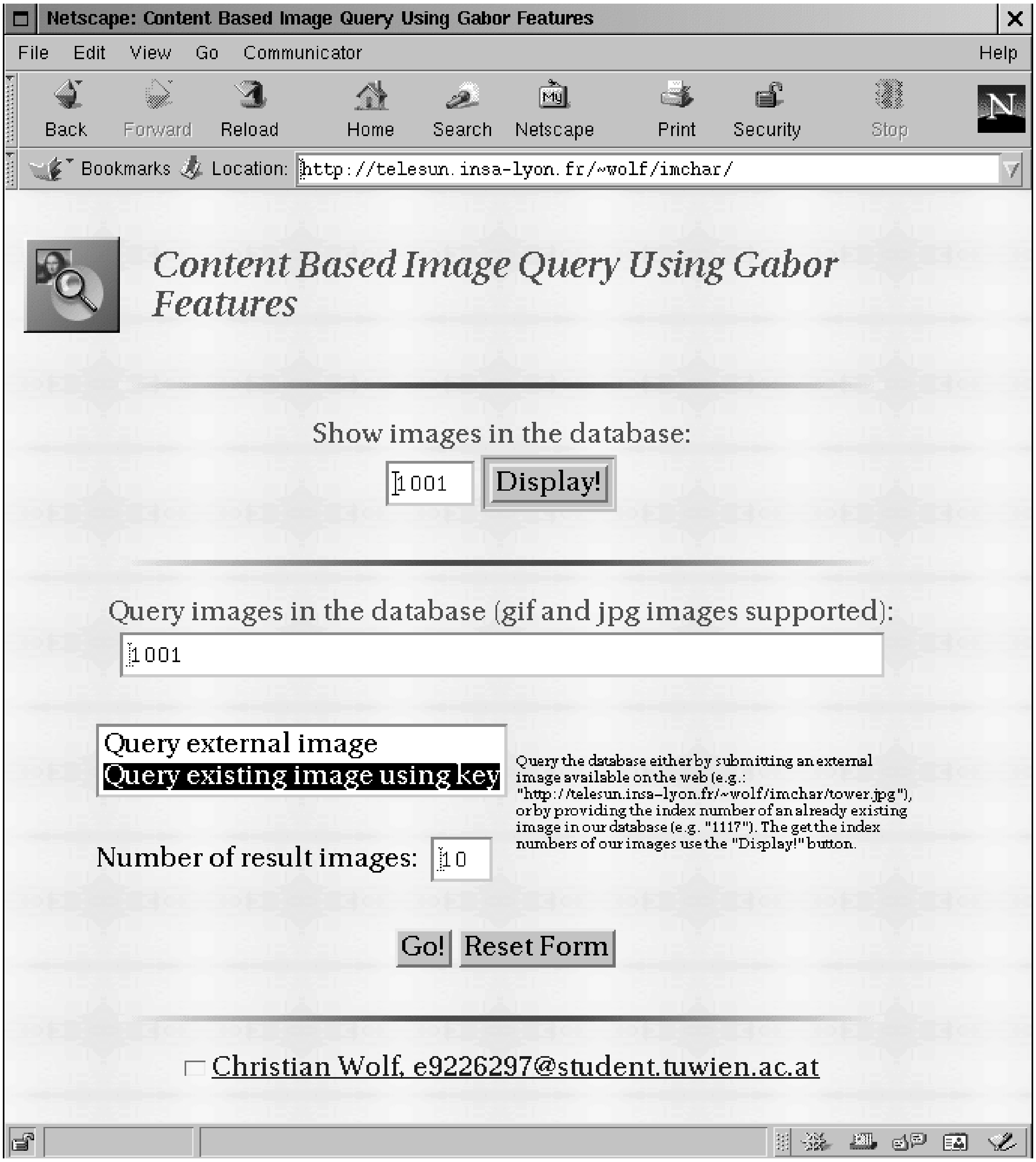


Figure B.2: The Main Page of the Web Frontend

Bibliography

[1] S. Bhattacharjee. Image retrieval based on structural content. Technical Report SSC/1999/004, Dept. of Electrical Eng., E.P.F.L, CH-1015, Lausanne, 1999.

[2] S. Bres and J.M. Jolion. Detection of interest points for image indexing. In *3rd Int. Conf. on Visual Inf. Systems, Visual 99*, pages 427–434. Springer, Lecture Notes in Computer Science, 1614, June 1999.

[3] J.M. Coggins and A.K. Jain. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, 3(3):195–203, 1985.

[4] E. Etievent, F. LeBourgeois, and J.M. Jolion. Assisted video sequences indexing, shot detection and motion analysis based on interest points. Technical Report TR-9903, Laboratoire de Reconnaissance de Formes et Vision, March 99.

[5] H.G. Feichtinger and T. Strohmer. *Gabor Analysis and Algorithms*. Birkhäuser, 1998.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ahsley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steel, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–31, Sept. 19995. http://www.ibm.com.

[7] D. Gabor. Theory of communication. *J. IEE (London)*, 93(III):429–457, 1946.

[8] A. Graps. An introduction to wavelets. Technical report, Institute of Electrical and Electronics Engineers, 1995.

[9] A. Gupta. Visual information retrieval: A virage perspective. Technical report, Virage, Inc., 1995-97. http://ei.cs.vt.edu/ mm/cache/VIRpaper.htm.

[10] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings 4th Alvey Visual Conference*. Plessey Research Roke Manor, UK, 1988.

[11] B. Huet and E.R. Hancock. Cartographic indexing into a database of remotely sensed images. In *Third IEEE Workshop on Applications of Computer Vision (WACV96)*, pages 8–14, Sarasota, Dec 1996.

[12] A.K. Jain and F. Farrokhina. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[13] J.M. Jolion and A. Montanvert. The adaptive pyramid, a framework for 2d image analysis. In *Computer Vision, Graphics and Image Processing: Image Understanding*, pages 55(3):339–348, May 1992.

[14] W.G. Kropatsch. Building irregular pyramids by dual-graph contraction. *IEE Proc.-Vis Image Signal Process.*, 142(6):366–374, December 1995.

[15] K.I. Laws. Rapid texture identification. In *Proc. SPIE Conf. Image Processing for Missile Guidance*, pages 376–380, 1980.

[16] E. Loupias, N. Sebe S. Bres, and J.M. Jolion. Wavelet-based salient points for image retrieval. In *International Conference on Image Processing, Vancouver, Canada*, 2000.

[17] D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999.

[18] J. Matas, D. Koubaroulis, and J. Kittler. Performance evaluation of the multi-modal neighbourhood signature method for colour object recognition. In *Czech Pattern Recognition Workshop 2000, Czech Technical University, Center for Machine Perception*, Feb. 2000.

[19] J. Matas, R. Marik, and J. Kittler. The color adjacency graph representation of multicolored objects. Technical Report VSSP-TR-1/95, Department of Electronic & Electrical Engineering, University of Surrey, Guildford, 1995.

[20] R. Megret. Mesure de l'erreur introduite par jpeg sur le filtrage de gabor. Technical report, Laboratoire Reconnaissance de Formes et Vision, INSA de Lyon, July 1998.

[21] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. Technical report, Vision Speech and Signal Processing Group, Departement of Electronic and Electrical Engineering, University of Surrey, Guildford, Surrey GU25XH, England, 1999.

[22] H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. of 5th International Joint Conference on Artifical Intelligence, 584*, page p. 587, 1977.

[23] O. Popescu. Utilisation des points d'intérêt pour l'indexation d'images. Technical Report RR 99.07, Laboratoire Reconnaissance de Formes et Vision., 1999.

[24] T. Randen and J. Hakon. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4), April 1999.

[25] Y. Rubner and C. Tomasi. Texture-based image retrieval without segmentation. In *International Conference on Computer Vision*, 1999.

[26] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. Technical report, Computer Science Departement, Stanford University.

[27] S. Santini and R. Jain. The graphical specification of similarity queries. *Journal of Visual Languages and Computing*, 7(4), 1997.

[28] C. Schmidt and R. Mohr. Local gray value invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5), May 1997.

[29] S. Siggelkow and H. Burkhard. Local invariant feature histograms. Technical Report IIF-LMB 01/98, Albert-Ludwigs-Univ. Freiburg, Institut f. Informatik, Jan 1998.

[30] S. Smith and J. Brady. Susan - a new approach to low level image processing. *Int. Journal of Computer Vision*, 23(1):45–78, May 1997.

[31] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: A primer, part i. *IEEE Computer Graphics and Applications*, 15(3):76–84, May 1995.

[32] M. Stricker and A. Dimai. Color indexing with weak spatial constraints. *SPIE*, 2670/29(0-8194-2044-1), 1996.

[33] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[34] M. Tuceryan and A.K. Jain. *Handbook Pattern Recognition and Computer Vision*, chapter Texture Analysis, pages 235–276. World Scientific, 1993.

[35] B. Zitova, J. Flusser, J. Kautsky, and G. Peters. Feature point detection in multiframe images. Technical report, Czech Pattern Recognition Workshop, Feb. 2000.