A Probabilistic Similarity Framework for Content-Based Image Retrieval

Selim Aksoy

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2001

Program Authorized to Offer Degree: Department of Electrical Engineering

University of Washington

Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Selim Aksoy

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Chair of Supervisory Committee:

Robert M. Haralick

Reading Committee:

Mari Ostendorf

Linda G. Shapiro

Date:

University of Washington

Abstract

A Probabilistic Similarity Framework for Content-Based Image Retrieval

by Selim Aksoy

Chair of Supervisory Committee:

Professor Robert M. Haralick
Department of Electrical Engineering

Content-based retrieval from image databases has become a popular research area where conventional database retrieval methods are not sufficient because they depend on exact matches of keywords and require an enormous amount of human involvement during manual annotation. Initial work on content-based retrieval focused on using low-level features like color and texture for image representation, and a geometric framework of distances in the feature space for similarity. A challenging problem in image retrieval is the fusion of information from multiple features and similarity measures. In this dissertation, we pose the retrieval problem in a probabilistic framework where the goal is to minimize the classification error in a setting of two classes; the relevance and irrelevance classes of the query. We propose effective solutions to different levels of the retrieval process within this framework. Feature extraction and normalization is done by maximizing class separability, similarity is measured using likelihood and posterior ratios, and post-processing is done using graph-theoretic image grouping and a Bayesian relevance feedback architecture. A key aspect of our framework is a two-level modeling of probability. The first level uses parametric density models to compute class-conditional probabilities from feature vectors and can be interpreted as a mapping from the high-dimensional feature space to the two-dimensional probability space. The second level includes training simple linear classifiers in multiple probability spaces for multiple feature vectors and corresponds to a modeling of "probability of probability" to compensate for errors due to imperfect density modeling in the feature space. Furthermore, classifier combination rules and a naive Bayesian network effectively fuse information from multiple features and similarity models.

Performance evaluation was done using extensive experiments on three groundtruth databases including aerial, satellite, texture and stock photo images. The proposed probabilistic framework performed more robustly and significantly better than the commonly used geometric framework and two competing algorithms from the literature. We obtained 8-20% relative improvement in precision over the cases where the best feature vectors were used individually. Moreover, a few feedback iterations resulted in an average precision of more than 94% for all three databases.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Professor Robert M. Haralick for his guidance throughout my graduate study. I feel lucky to experience with him the process of searching for correct, complete and consistent solutions to challenging research problems. He has constantly led me in approaching the problems from different states of understanding. I am also thankful for his financial support which has let me work on interesting projects.

I am very thankful to Professor Linda G. Shapiro for her generous help and advice at every stage of my work. I am also grateful for her financial support during the last year of my study. I would also like to thank Professor Mari Ostendorf for her detailed and stimulating comments on my dissertation. I thank Professor Jenq-Neng Hwang for serving on both my master's and doctoral committees. I also thank Professor Eve Riskin for her comments on an early draft of this dissertation.

The Intelligent Systems Laboratory has been a very pleasant place to work. I would like to thank all of my colleagues, in particular Dr. Gang Liu, Ming Ye, Yalin Wang, Mingzhou Song, Desikachari Nadadur, Dr. Jisheng Liang, Dr. Lei Sui and Dr. Qiang Ji for their help and valuable discussions. I thank Dr. Giovanni Marchisio and Dr. Kris Koperski for their stimulating discussions during my internship at MathSoft Inc., Professor Michael D. Perlman from the Statistics Department for answering my questions about parameter estimation and Bayesian networks, and Professor Howard J. Chizeck, Professor John D. Sahr, Frankye Jones and other members of the graduate advising staff for their help in getting this dissertation turned in on time.

I am extremely grateful to Shihomi Ara for her constant encouragement, special support and friendship. I am also very thankful to Ibrahim Karliga, Gokhan Sahin and other friends for their fun and supportive friendship.

Last, but never the least, I thank my parents and relatives for their endless love, support, understanding and patience.

DEDICATION

To my mother and father,

for their love, support and patience.

Chapter 1

INTRODUCTION

1.1 Content-Based Image Retrieval

Image database retrieval has become a very popular research area in recent years [85, 171, 187] due to the large amount of images that are generated by various applications and the advances in computation power, storage devices, scanning, networking, image compression, desktop publishing and the World Wide Web. New tools are required to help users create, manage, and retrieve images from on-line databases. The value of these systems can greatly increase if they can provide the ability of directly searching non-textual data, the "content" of the image, instead of searching only on the associated textual information. The main purpose of a content-based image database retrieval system is to effectively and efficiently use the information stored in the image database.

In a typical content-based image retrieval application, the user has an image and/or just a subject he or she is interested in and wants to find images from the database that are similar or related to the example image. Possible application areas include fashion design, museum catalogs, movie production, architectural design, remote sensing, geographic information systems, scientific database management, logo and trademark database management, law enforcement and criminal investigation, commercial or personal picture archiving, and military and medical databases. There are already some commercial systems as well as museums and other institutions allowing content-based search in their large image collections [183, Chapter 8].

Conventional database retrieval methods will not be sufficient to retrieve this kind of data because they depend on file IDs, keywords, or text associated with the images. They require an enormous amount of human involvement during manual annotation. They do not allow queries based directly on the visual properties of the images. They depend on the particular vocabulary used, and they do not provide queries for images "similar" to a given image. In conventional databases, retrieval is based on an exact match of the attribute values which do not have the ability to rank-order results by the degree of similarity with the query image. Unfortunately, it is impossible to represent the content of an image in a few words and also, different people may perceive the same image differently.

Content-based retrieval emerged as an alternative research area to address this problem in the early 90's [171, 187]. To overcome the difficulties caused by the manual annotations in text-based retrieval methods, images are indexed by their own visual content, hence the name "content-based" retrieval. Queries can be based on the use of measures that evaluate the similarity of two images based on pre-defined criteria. In the general framework, first a feature extraction and/or object recognition algorithm is used to extract information from an image during its insertion into the database. After images are added to the database and features are extracted, queries can be formed to allow users to retrieve images.

Queries for a content-based image database retrieval system can be based on different features and can be from different classes like color, texture, shape, sketch, spatial constraints, browsing (interactive) and attributes [85, 183]. Color queries let users retrieve images containing specific colors as input by the user. The user can specify percentages and locations of colors in the image. Grids can be used to specify the color layout. Texture queries allow retrieving images containing a specific texture. Similar to color, texture queries also include histograms or gridded texture layouts selected from texture libraries. Shape queries may be automatic or semi-automatic where the user helps the computer outline the shapes of interest in the image. Retrieval by sketch lets users draw some shapes and colors in an image and then retrieves a similar image from the database. The spatial constraints category deals with a class of queries based on spatial and topological relationships among the objects in an image. These relationships may range from directional relationships to adjacency, overlap, and containment involving a pair of objects or multiple objects. Retrieval by browsing (interactive retrieval) is performed when users are not exactly clear about their retrieval needs or are unfamiliar with the structure and types of information available in the image database. All of these queries can be formed either using some predefined options or using another image, which is also called query-by-example.

1.2 Overview of Previous Work

Previous work in image retrieval can be roughly divided into the following categories according to the order of processing:

- feature extraction,
- region finding,
- matching,
- feature combination,
- relevance feedback.

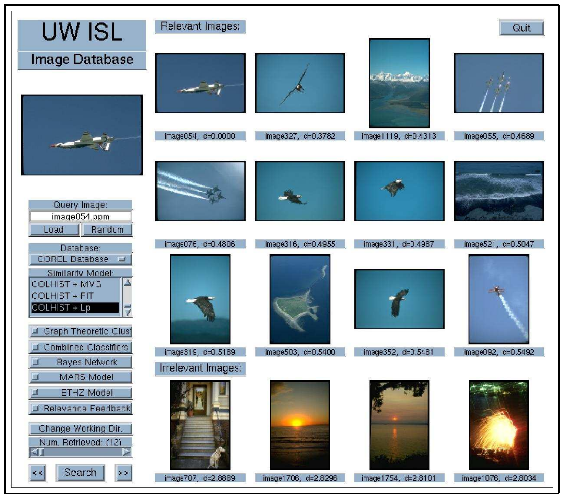
(Practical database retrieval systems also involve other issues like graphical user interface design and indexing problems in very large databases [18, 23] but these are out of the scope of this dissertation.)

Feature extraction has received the highest attention among all the other categories. Initial work on content-based retrieval focused on using low-level features like color and texture for image representation. Color features include histograms [192] of RGB values or values of other color models, color moments [191], angles between average color vectors [12], co-occurrences of colors [189], and invariant color models [83]. Even though color is very important for visual perception of many images, it cannot always distinguish objects as shown in Figure 1.1. (The first three rows in the user interface show the best 12 matches and the last row shows the worst 4 matches to the query. The user interface will be described in more detail in Section 1.5.4.)

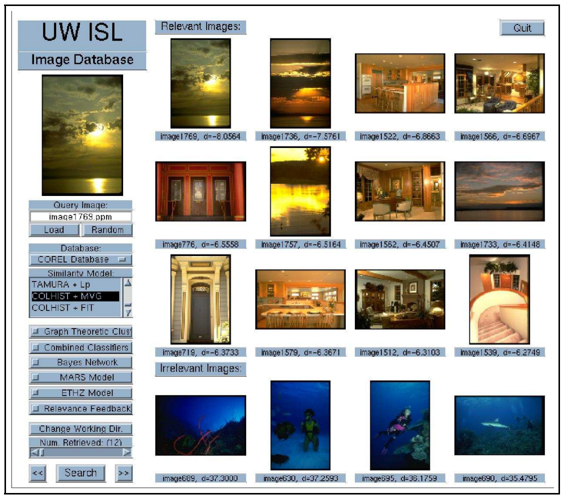
Texture proved useful in many problems in computer vision like remote sensing and inspection so it became an obvious choice as a feature for many image retrieval systems. Examples include the Wold features [133], Laws' texture energy maps [116], wavelet coefficients [102], Gabor filters [136], spatial features [131], and edge maps [213]. Similar to color, texture can also have problems in finding similarities between images and can give many false alarms which are often harder to understand than the ones for color. An illustration is given in Figure 1.1.

A relatively less popular approach is to use shape information. Since image segmentation is a very hard problem, especially in complex images, it is very hard to automatically compute accurate shape features. Some example uses of shape in image retrieval include stiffness matrices [160], Fourier descriptors [169], elastic matching [28], and deformable shape models [134]. Features that are developed for specific domains are also used, e.g. facial features [16], fingerprint minutiae patterns [103], and skin color features [73].

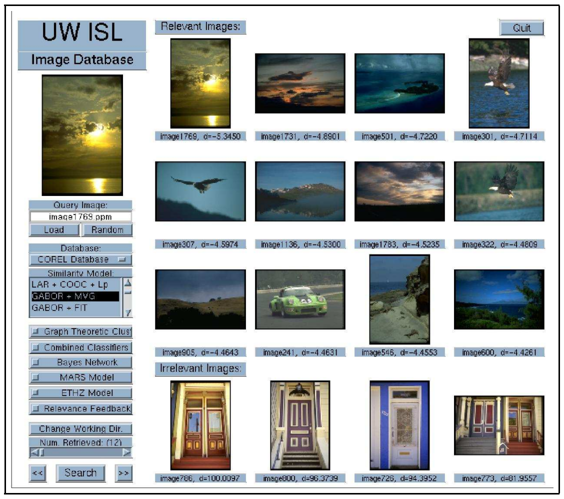
More recent approaches developed region-based query systems which involve image segmentation based on color and texture but the region segmentation algorithms are still too slow to be used in an image retrieval application. Some promising approaches for region finding include clustering and grouping in spatial color-texture space [169], segmentation in the color and texture space using expectation-maximization [38], identifying the direction of changes in feature values [135], graph-based approaches [184, 185, 71, 72], and non-parametric clustering [156].



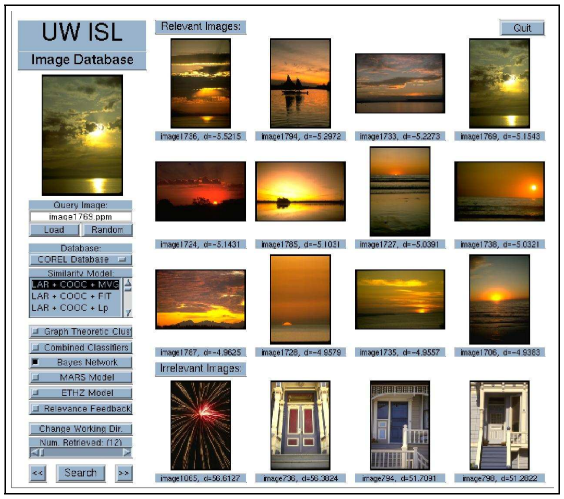
(a) An example query for airplanes using color histograms (4/12)



(b) An example query for sunsets using color histograms (4/12)



(c) An example query for sunsets using Gabor texture features (3/12)



(d) An example query for sunsets using Bayesian network (12/12)

Figure 1.1: Example queries using color and texture feature vectors. Using only color cannot distinguish airplanes from eagles in (a) and sunsets from rooms and doors with similar colors in (b). Using only texture gives worse results for this particular sunset image in (c). Effective combination of color and texture (using a model that will be proposed in this dissertation) can give significant improvements in (d). The numbers in parentheses in sub-captions show the number of correct matches.

After each image or region is associated with a feature vector, the next major step is to use measures to find similarities between images. In systems where images or regions are represented by feature vectors, the nearest neighbor rule [64] has been the most popular choice. A distance measure is used to rank the database images in ascending order of their distances to the query image, which is assumed to correspond to a descending order of similarity. Popular distance measures have been the Euclidean (L_2) distance [74, 160, 131, 188], the city-block (L_1) distance [136, 188], the general Minkowsky L_p distance [178], the weighted Euclidean distance [19, 172], the Cauchy distance [180], and the Mahalanobis distance [160, 188]. Some systems which include domain-specific features use similarity measures that are specific to those features. These include histogram intersection [188], Chamfer measure for shape matching [31], Hausdorff distance for object matching [168], tree search, self organizing maps [212], and graph matching [98].

The nearest neighbor rule suffers from the fact that low-level features cannot always map visually similar images into nearby locations in the feature space and images that are quite irrelevant to the query image can be easily retrieved simply because they are close to it in the feature space. It was shown that probabilistic measures can be much more effective than the geometric distance-based approaches [146, 145].

To overcome the limitations of low-level features, some systems allowed the user to query the database using multiple feature vectors. Possible approaches include appending different feature vectors and treating the result as a big global feature vector [131], letting the user weight different features [74, 15, 107], hierarchical classification using different feature vectors at each level [201, 200, 68], taking linear or Boolean combinations of distances computed using different feature vectors [24], neural networks [87, 86, 152, 153], Bayesian networks [177, 124, 204], and boosting multiple classifiers that are trained on individual features [196]. Combining information from multiple feature vectors can be quite useful as shown in Figure 1.1.

Recently, relevance feedback is motivated from the success in document retrieval [175] and has been a popular technique for improving retrieval performance. An initial search in the database is done using the original query input by the user. Then, the user labels some of the results as relevant and irrelevant and this information is incorporated into the database search in terms of iterative retrievals.

Commonly used approaches for relevance feedback include the vector space model [173, 47, 82], switching between multiple feature vectors and/or similarity measures according to the comparisons between the rankings of the system and the rankings of the user [174, 20], giving weights to each feature and/or similarity measure and updating the weights according to the positive and negative feedback from the user [172, 159, 170], self organizing feature maps [143], creating a probabilistic user model [51, 50, 155], using a rule-based model [32], modifying the distance measure [43], reorganizing the retrieval results [195, 44, 42], and feature density estimation [150, 141].

The following sections give the notation, motivation and problem definition for the framework proposed in this dissertation. A more detailed literature review will be given in Chapter 2.

1.3 Notation

The following notation is used throughout the dissertation.

ξ_i **:** i'th image in the database.

$\mathbb{R}$ **:** set of real numbers.

$\mathbf{M} \in \mathbb{R}^{(n \times m)}$ **:** matrix with n rows and m columns.

$\mathbf{x} \in \mathbb{R}^{(q \times 1)}$ **:** vector with length q.

$\mathbf{x_i} \in \mathbb{R}^{(q \times 1)}$ **:** i'th vector in the sample for $\mathbf{x}$.

$\mathbf{x}_i \in \mathbb{R}$ **:** i'th component of the vector $\mathbf{x}$.

$x \in \mathbb{R}$ **:** scalar.

$x_i \in \mathbb{R}$ **:** i'th value in the sample for x.

$z^{(i)}$ **:** i'th element of z (can be image, vector or scalar) in time or in an ordered set.

$\mathcal{C}$ **:** set or class.

$\#\mathcal{C}$ **:** number of elements in set $\mathcal{C}$.

1.4 Motivation and Problem Definition

The image retrieval scenario addressed here begins with a query expressed by an image. The user inputs one or more images to retrieve images from the database that are similar to those input. The goal of this work is to develop similarity models to improve retrieval performance. Since the problem of assigning similarity between images is still not well-defined, there are many unjustified heuristics in the literature. Most of the algorithms and decision criteria are developed by trial and error thresholds with insufficient performance evaluation that use only a few examples. However, a well-defined formulation is required to tune the algorithms, perform parameter estimation, choose thresholds, etc.

Images that we have to deal with usually have a lot of structure. In the level where the users usually form their queries a high-level alphabet and a complex grammar is required. This alphabet will not likely be formed from keywords because manually extracting all possible keywords from all images is not feasible. However, simple low-level visual features are not effective enough alone to do retrieval with acceptable results either. On the other hand, semantic-level object recognition is still an unsolved problem in computer vision. Thus, the alphabet we use should be simple enough so that it can be easily computed and be generally applicable to a wide range of images. Even though the individual elements of this alphabet may not be powerful enough to represent complex images, a strong grammar can make use of their advantages and achieve a better representation.

In this dissertation, we develop an effective solution to this problem using low-level visual feature vectors as the elements of the alphabet and the pattern recognition and Bayesian theory as the grammar for their combination. We approach the problem by defining sub-problems that can be optimized or tuned in a theoretical framework, and then combine these approaches to obtain the advantages of different algorithms. Although there has been an enormous amount of work on developing features for usually restricted domains of images, similarity measures have not received significant attention. Besides, there is no generally applicable and effective framework to combine multiple feature vectors and similarity measures.

We pose the retrieval problem in a classification framework. The goal is to minimize the classification error in a two-class setting, where the classes are the relevance class and the irrelevance class. Given a pair of images, one being the query image and the other one being an image in the database, the pair should be assigned to the relevance class if two images are similar and to the irrelevance class if they are not.

Unlike other approaches where an ambiguity exists about the images that do not belong to any of the defined classes (city, forest, etc.), or where there are as many classes as the number of images in the database, the binary setting of the relevance and irrelevance classes completely and unambiguously partitions any database given the query image. In our setting, the classes are defined relative to the query image, not to the images in the database. Here we assume that similar images have similar feature values and dissimilar images have relatively different feature values, and base similarity between two images in terms of feature differences. For example, two images of trees can be modeled by the relevance class if they have similar feature values (i.e. small feature differences) as well as two images of sunsets which also belong to the relevance class because of their similar features. On the other hand, one image with a tree and another image that includes a sunset belong to the irrelevance class because of the differences in their feature values as well as an image of a horse and a building image which are expected to have different features. Hence, the two-class modeling can intuitively classify pairwise similarities between different images under the assumption mentioned above.

Given the relevance class $\mathcal{A}$, the irrelevance class $\mathcal{B}$, the query image ξ_i and an image ξ_j from the database, the classification error for the image pair (ξ_i, ξ_j) is computed as

$$\begin{aligned} P(\text{error}) =& 0.5\, P((\xi_i,\xi_j) \text{ assigned to } \mathcal{B}, (\xi_i,\xi_j) \text{ belongs to } \mathcal{A}) + \\ & 0.5\, P((\xi_i,\xi_j) \text{ assigned to } \mathcal{A}, (\xi_i,\xi_j) \text{ belongs to } \mathcal{B}). \end{aligned} \tag{1.1}$$

Pattern recognition literature provides many choices for a classifier. Since the Bayes classifier gives the theoretical minimum classification error [78, 64], it is the ideal choice for the classifier. Since it uses the posterior probabilities to make the decision, the posterior probabilities are the ideal features for classification. This setting can be interpreted as a mapping from the high-dimensional feature space to the two-dimensional probability space. Similarity can then be computed as likelihood in the probabilistic setting instead of computing distances in the geometric setting.

The Bayes classifier minimizes the classification error in Equation (1.1) using the decision rule [64]

$$\text{assign } (\xi_i,\xi_j) \text{ to } \begin{cases} \text{class } \mathcal{A} & \text{if } P(\mathcal{A}|(\xi_i,\xi_j)) > P(\mathcal{B}|(\xi_i,\xi_j)) \\ \text{class } \mathcal{B} & \text{otherwise,} \end{cases} \tag{1.2}$$

i.e. it compares the *a posteriori* probabilities and assigns the input to the class whose *a posteriori* probability is the largest. Therefore, these probabilities carry sufficient information to set up the Bayesian classifier[1]. The Bayes error in this space is identical

[1]Since the sum of these probabilities is 1, they are linearly dependent. Thus, $P(\mathcal{A}|(\xi_i,\xi_j))$ is actually the smallest set for classification.

to the Bayes error in the original feature space and no classification information is lost.

In the two-class problem, the discriminant function is commonly represented in the posterior ratio form

$$\Delta(\xi_i, \xi_j) = \frac{P(\mathcal{A}|(\xi_i, \xi_j))}{P(\mathcal{B}|(\xi_i, \xi_j))} \tag{1.3}$$

which gives the decision rule

$$\text{assign } (\xi_i, \xi_j) \text{ to } \quad \begin{cases} \text{class } \mathcal{A} & \text{if } \Delta(\xi_i, \xi_j) > 1 \\ \text{class } \mathcal{B} & \text{if } \Delta(\xi_i, \xi_j) \leq 1. \end{cases} \tag{1.4}$$

If we assume that both of the classes are equally likely, i.e. $P(\mathcal{A}) = P(\mathcal{B})$, the discriminant function becomes the likelihood ratio

$$\begin{aligned} \Delta(\xi_i, \xi_j) &= \frac{P((\xi_i, \xi_j)|\mathcal{A})P(\mathcal{A})}{P((\xi_i, \xi_j)|\mathcal{B})P(\mathcal{B})} \\ &= \frac{P((\xi_i, \xi_j)|\mathcal{A})}{P((\xi_i, \xi_j)|\mathcal{B})}. \end{aligned} \tag{1.5}$$

Then, the database images that are assigned to the relevance class $\mathcal{A}$ can be ordered according to the likelihood ratio $\Delta(\xi_i, \xi_j)$ and the best matches to the query can be found as the ones with the largest likelihood ratio (which gives exactly the same ordering as the posterior probability for the relevance class under the equal priors assumption, or when the priors are unknown but constant).

After defining the decision rule, the problem becomes finding effective models for the class-conditional distributions. We divide the retrieval process into three levels:

- pre-processing level (feature extraction and normalization),
- similarity level (similarity computation between the query image and the images in the database),
- post-processing level (iterative retrievals to improve the performance).

This dissertation describes the solutions that we propose for these levels. Each level includes several probabilistic models to estimate the class-conditional probabilities $P((\xi_i, \xi_j)|\mathcal{A})$ and $P((\xi_i, \xi_j)|\mathcal{B})$. We also have criteria for optimization or selection of each model in each level. We compare each model to other approaches in the same level. Then, we describe a framework where these solutions can be combined to make a final decision about the similarity between images. The overall system is also compared to other content-based retrieval systems. The experiments show that classification performance is a good indicator of the retrieval performance. Designing the retrieval system in our two-class classification framework achieves better performance than the commonly used geometric framework which uses distances in the feature space as the similarity measure.

An object/process diagram of the proposed system is given in Figure 1.2, where rectangles represent objects and ellipses represent processes. Different sections of the system will be presented in the following chapters.

1.5 Experimental Setup

1.5.1 Databases for Experiments

We use three different manually groundtruthed databases for performance evaluation:

1. ISL Database:
   It includes images from the Fort Hood Data [76] that were supplied for the RADIUS Project by the Digital Mapping Laboratory at the Carnegie Mellon University. These aerial images consist of visible light images of the Fort Hood area in Texas. We used the images `fhn711`, `fhn713`, `fhn715`, `fhn717` and `fhn719`, and obtained 9,000 256×256 images with some of them overlapping by at most half of their area. The second source for this database is the Remote Sensing image collection from the LANDSAT and Defense Meteorological Satellite Program (DMSP) Satellites. These include images of USA (800×720), Chernobyl

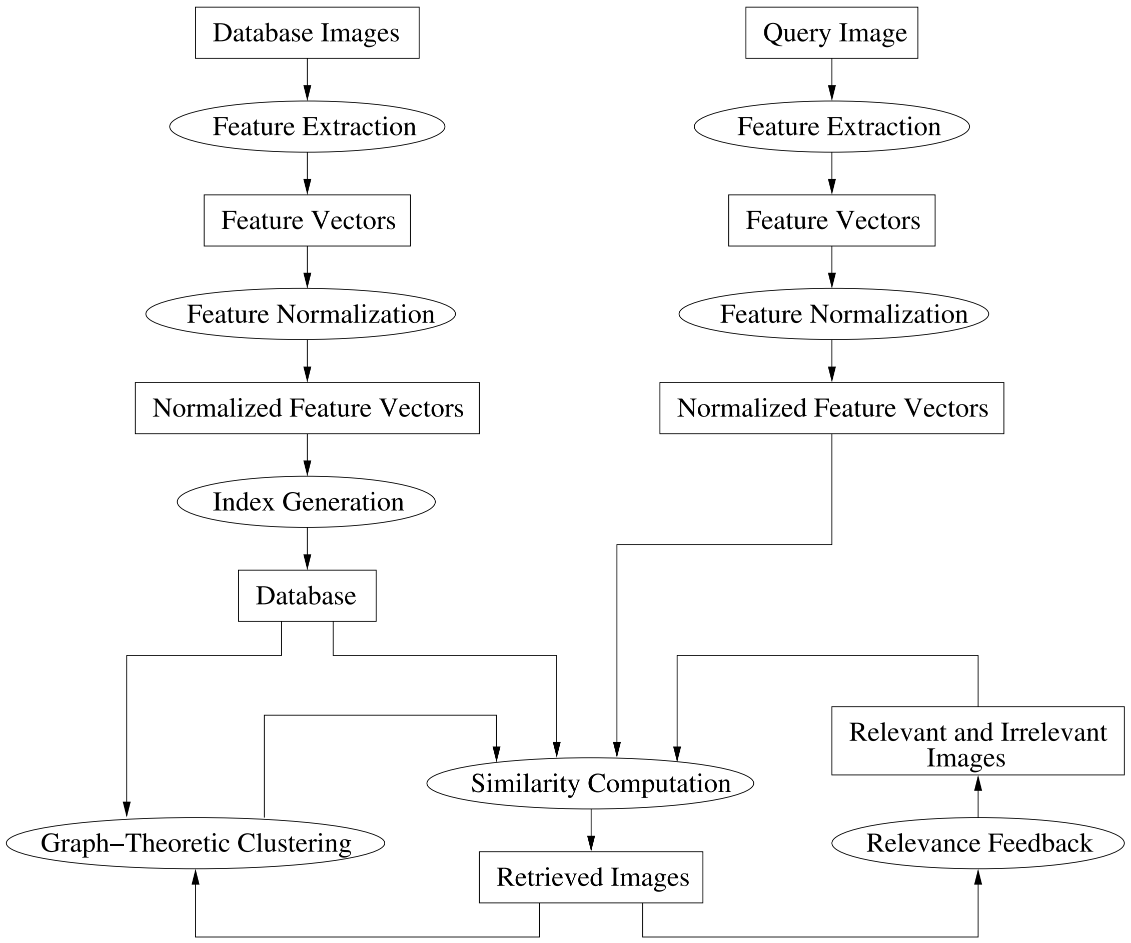


Figure 1.2: Object/process diagram for the proposed content-based image retrieval system. Rectangles represent objects and ellipses represent processes.

(512×512), and the North Pole (608×896), making a total of 1,410 256×256 images.

We randomly selected 2000 images among the non-overlapping ones from the total of 10,410 and grouped them into 7 categories: parking lots (85 images), roads (85 images), residential areas, (86 images), landscapes (85 images), LANDSAT USA (85 images), DMSP North Pole (87 images) and LANDSAT Chernobyl (87 images). The final groundtruth database contains 600 images. Since these images are grayscale, we use only texture feature vectors for image representation. All images in the groundtruth groups are given in Appendix A.

2. VisTeX Database:

   MIT Media Laboratory's Vision Texture Database [207] has been recently used as a common dataset because it provides an easy-to-obtain groundtruth. We use 46 512×512 images: Bark.0000, Bark.0001, Bark.0006, Bark.0008, Bark.0012, Brick.0000, Brick.0002, Brick.0005, Fabric.0002, Fabric.0005, Fabric.0007, Fabric.0009, Fabric.0011, Fabric.0013, Fabric.0015, Fabric.0017, Fabric.0019, Flowers.0000, Flowers.0002, Flowers.0007, Food.0000, Food.0001, Food.0004, Food. 0005, Food.0006, Grass.0002, Leaves.0003, Leaves.0008, Leaves.0010, Leaves.0011, Leaves.0014, Leaves.0016, Metal.0000, Metal.0002, Metal.0004, Misc.0000, Misc. 0002, Sand.0000, Sand.0005, Stone.0002, Stone.0005, Tile.0007, Water.0002, Water.0003, Water.0004 and Wood.0002. Since each image has a relatively homogeneous texture, they are divided into 16 128×128 non-overlapping images that make a groundtruth of 736 images divided into 46 categories with 16 images in each category. Smaller parts of this dataset were used in many experiments in the literature (for example in the University of Illinois' MARS project [172]). We use both texture and color feature vectors for image representation for this database. All images in the groundtruth groups are given in Appendix B.

3. COREL Database:

   COREL Photo Stock Library [48] includes CDs on many topics with 100 images in each CD. It has recently become popular as a test dataset for content-based retrieval experiments. However, due to the high level categorization in the CDs, not many groundtruth-based quantitative evaluations were reported on this dataset. We use 18 categories for a total of 1,575 images: air shows (82 images), Arabian horses (90 images), auto racing (88 images), bald eagles (80 images), cheetahs (91 images), coasts (92 images), divers and diving (92 images), doors of San Francisco (89 images), English country gardens (85 images), fields (85 images), fireworks (91 images), glaciers and mountains (85 images), land of the pyramids (91 images), owls (85 images), polar bears (85 images), residential interiors (91 images), roses (89 images) and sunsets/sunrises (84 images). Most of these categories were used in the experiments in the MIT Media Lab (e.g. [206]) and University of California at Berkeley's Blobworld project (e.g. [37]). Some of the images in each group were discarded because they are visually too inconsistent with the rest of the category. A partial list of the images that were discarded was obtained from Dr. Chad Carson from the University of California at Berkeley. We use both texture and color feature vectors for image representation for this database. All images in the groundtruth groups are given in Appendix C.

1.5.2 Experimental Protocol

We use independent training and testing image sets for the experiments. For a particular database, we first randomly choose N images from each groundtruth group. This set of images form the training dataset. The training image pairs for the relevance class consist of all possible within-group pairings (N^2K pairings where K is the number of groundtruth groups) and the training image pairs for the irrelevance class consist of the same number of randomly selected between-group pairings. Randomly selected pairings of the images that are not in the training dataset are used for testing the classification algorithms. These images are also used as queries in retrieval experiments.

We use approximately one-third of all data for training and the remaining two-thirds for testing. N is chosen to be 30 for the ISL Database, 6 for the VisTex Database and 30 for the COREL Database. The following list summarizes the number of image pairs used in the experiments:

- ISL Database
  - 600 images in 7 categories (180,300 total possible image pairs),
  - 6,300 training image pairs,
  - 21,175 testing image pairs formed by images that are different from the ones in training.
- VisTex Database
  - 736 images in 46 categories (271,216 total possible image pairs),
  - 1,656 training image pairs,
  - 4,600 testing image pairs formed by images that are different from the ones in training.
- COREL Database
  - 1,575 images in 18 categories (1,241,100 total possible image pairs),
  - 16,200 training image pairs,
  - 27,378 testing image pairs formed by images that are different from the ones in training.

All of these training and testing sets are symmetric, i.e. if the image pair (ξ_i, ξ_j) is in a training (testing) set, the image pair (ξ_j, ξ_i) is also in that training (testing) set.

1.5.3 Performance Evaluation

Testing content-based retrieval systems and comparing their performances is an open question. In most of the content-based retrieval literature, researchers presented example queries to visually evaluate the performance of their systems. To compare two content-based retrieval systems, we need experiments based on groundtruth data. There are measures like precision, recall, misdetection rate and false alarm rate that were already proposed in information retrieval and pattern recognition literature to evaluate the results of such experiments. After computing these measures for each content-based retrieval system, we will be able to compare them based on how well they perform on the groundtruth data.

Two traditional measures for retrieval performance in the information retrieval literature are precision and recall. Given a particular number of images retrieved, *precision* is defined as the percentage of retrieved images that are actually relevant $\left(\frac{\textit{retrieved and relevant}}{\textit{total number of retrieved}}\right)$ and *recall* is defined as the percentage of relevant images that are retrieved $\left(\frac{\textit{retrieved and relevant}}{\textit{total number of relevant}}\right)$ [175]. Given a query, high precision implies that very few irrelevant images have been retrieved, and high recall implies that much of what is relevant in the database has been retrieved. Lack of precision can be compared to a type 2 error (false alarm) and deficiency in recall for a given search is comparable to type 1 error (misdetection). For performance evaluation, one can plot precision and recall as a function of the number of images retrieved as well as the precision vs. recall curves for different numbers of images retrieved.

Much of the older work in content-based retrieval only gave illustrations of system performance without any real evaluation. Some of the newer work include performance measures like precision and recall but usually for small databases. Other measures that were used to evaluate the performance are the number of retrievals that have a specific target image among the set of retrieved images [107] and the average number of images required to converge to the desired specific target [49]. Some researchers first added noise to the database images and then used these noisy images to search for the original version in the database. Then the accuracy is defined as the average number of cases a noisy image can retrieve the original image [179]. These measures evaluate the effectiveness of the system in finding a specific target image instead of looking for visual similarities between the query image and the images in the database. Besides, they are also application specific. However, obtaining a general groundtruth dataset for content-based retrieval [198] is a very difficult task due to the difficulties in assigning complex images into specific categories.

We use the similarity-based approach of measuring precision and recall on our groundtruth databases for retrieval performance. We also use other well-known measures (like misdetection and false alarm) and define new measures (like consistency and progress) to evaluate specific components of our system.

1.5.4 Graphical User Interface

The Graphical User Interface (GUI), which is shown in Figure 1.3, is developed in MATLAB and can run with version 5.0 and higher. Most of the algorithms are implemented in C and are integrated to the GUI using UNIX system calls. In addition, we have a Java-based interface that supports a small subset of the algorithms. All the quantitative performance evaluation experiments that will be presented in this dissertation are done using UNIX shell scripts and the visual examples are created using the MATLAB-based GUI.

The upper left image in the GUI window shows the query image. The GUI supports the input of a query image either by specifying its path and filename, or by selection during random browsing of the database. The menus on the left allow the user choose different databases, feature representations, similarity models and their combinations. After a database search, the first three rows show 12 images that are the most relevant to the query in descending order of similarity and the last row shows 4 images that are the most irrelevant to the query in descending order of dissimilarity. User can also click on the retrieved images to give feedback to the system. A left mouse click represents positive feedback (and the image is marked green) and a right mouse click represents negative feedback (and the image is marked red). Subsequent clicks toggle the selections. The images selected as relevant by the user are also collected in a separate window for later use. The details of the algorithms are given in the rest of the dissertation.

1.6 Summary of Major Contributions

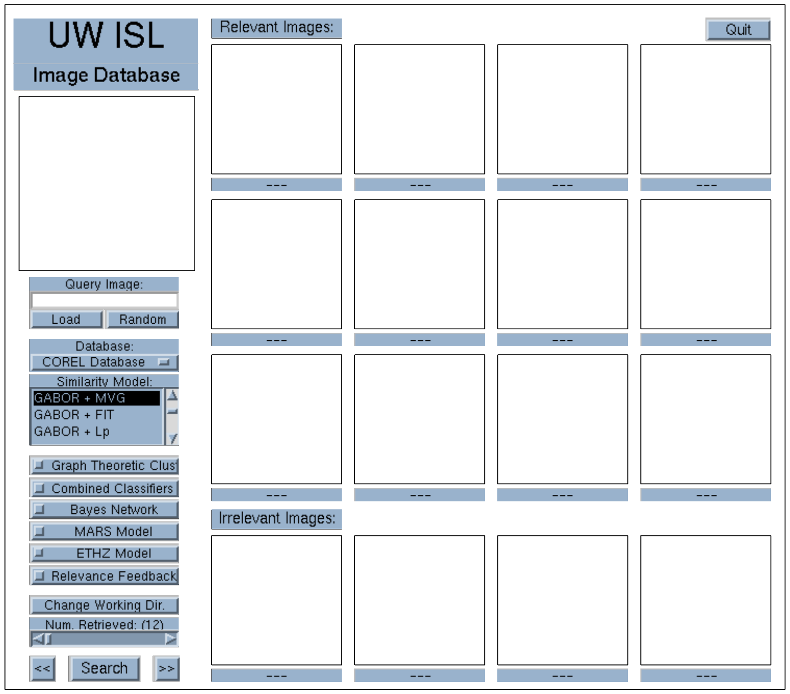
This dissertation contains original work in developing a classification framework for image retrieval, feature normalization, probabilistic similarity models, clustering for image grouping, combining multiple feature vectors and similarity measures, and relevance feedback.

In particular, this dissertation contains

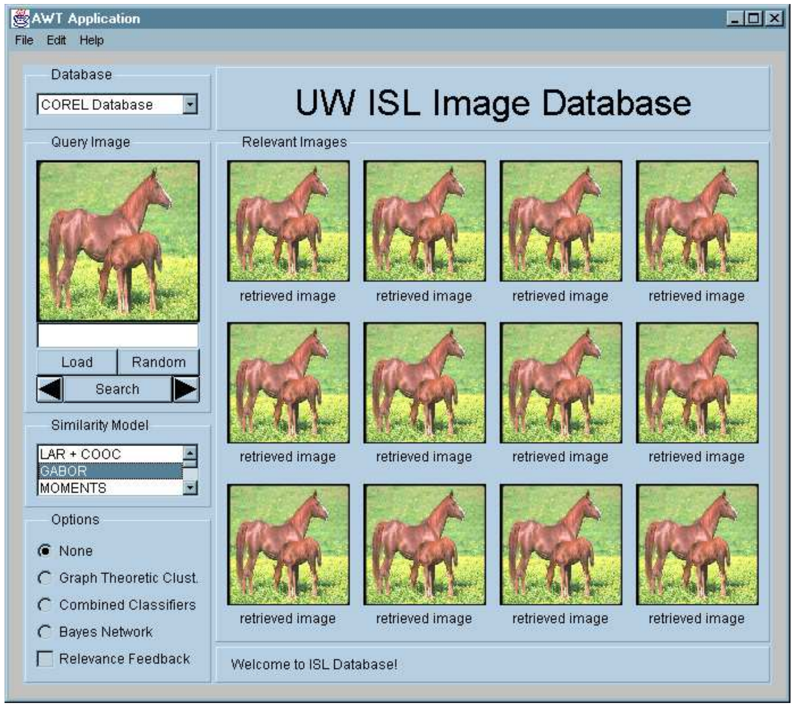
- a two-class classification framework and shows the strong relationship between classification and retrieval. Therefore, one can do the design (parameter estimation, model selection, choosing thresholds, etc.) in the classification framework and expect better results in retrieval;

- a study of feature normalization methods and their effects on retrieval performance. A class separability-based criterion is used to decide which normalization method should be used for a particular database. It is shown that studying the distribution of the features and using the results of this study significantly improves the results compared to making only general or arbitrary assumptions;

- the development of probabilistic similarity models and a likelihood ratio-based criterion to compute similarity between images. These models, which can be considered as a mapping from the high-dimensional feature space to the two-dimensional probability space allow us to do effective classification and retrieval



(a) MATLAB-based interface



(b) Java-based interface

Figure 1.3: Graphical user interfaces for the algorithms described in this dissertation.

by training simple linear classifiers. We perform a two-level modeling of probability because density models for high-dimensional feature vectors are not perfect and do not give the true posterior probabilities (therefore, we cannot compute the Bayes error). In two-level modeling, the first level includes models to compute probabilities of feature vectors and the second level includes models to compute probabilities of these probabilities to compensate for errors in modeling in the first level. The probabilistic similarity measures performed significantly better then the commonly used geometric framework where distances between feature vectors are used for image similarity. We also describe a classification-based criterion to choose the value of p in the commonly used Minkowsky L_p metric;

- a graph-theoretic approach for image grouping and retrieval which formulates the database search as a graph clustering problem. Furthermore, a model is developed to estimate the probability of each image being relevant to the query image given these clusters. It is shown that improvements in performance can be obtained using the constraint that retrieved images should be consistent with each other as well as being individually similar to the query image;

- a weighted distance approach to relevance feedback by weighting each feature component separately where the weight updating problem is formulated in an estimation and regression framework instead of the commonly used approach where the weights are updated using heuristic methods;

- a unified framework to combine different feature representations and similarity models. First, simple linear classifiers are trained to model probability in the two-dimensional probability space and then classifier combination rules from the pattern recognition literature are used to combine the decisions made by individual classifiers to obtain a final measure of similarity. Furthermore, a naive Bayesian network is used to incorporate the uncertainty in the estimation of probabilities and likelihood ratio values into the decision process. Extensive experiments for both quantitative and qualitative performance evaluation show that the proposed framework performs significantly better than two competing algorithms from the content-based image retrieval literature;

- a Bayesian relevance feedback model which intuitively and effectively uses the probabilities computed using different models to incorporate user's feedback to improve the performance. This feedback algorithm was more robust than two competing algorithms and achieved 4-14% relative improvement in precision compared to the best performing competitor in the experiments on three databases.

1.7 Dissertation Outline

The rest of the dissertation is organized as follows. A review of previous work in content-based image retrieval is given in Chapter 2. Chapter 3 discusses the effects of feature normalization and presents five normalization methods. Feature normalization is required to approximately equalize ranges of the features and make them have approximately the same effect in the computation of similarity. A class separability-based criterion is described to decide which normalization method should be used for a given dataset. Chapter 4 proposes a probabilistic approach to image retrieval. We describe probabilistic similarity models that compute the likelihood of two images being similar or dissimilar, one being the query image and the other one being an image in the database. We use three different methods to estimate the class-conditional probabilities used in the classifier. Effects of operating in the feature space versus operating in the probability space are discussed. The performances of probabilistic similarity methods are compared to the performances of the commonly used geometric approaches like the nearest neighbor rule with the Minkowsky L_p metric in ranking the images in the database. Chapter 5 introduces a post-processing method, a graph-theoretic clustering approach to image grouping and retrieval. We address a common observation in retrieval results that sometimes images that are quite irrelevant to the query image are also retrieved simply because they are close to the query image in the feature space. We propose a graph-theoretic approach for image retrieval by formulating the database search as a graph clustering problem by adding a constraint that the retrieved images should be close to each other as well as being close to the query image in the feature space. Another post-processing method, a weighted distance approach to relevance feedback is described in Chapter 6. Relevance feedback tries to capture the high-level concept of similarity and subjectivity in human perception by including the user in the retrieval loop. After formulating the weight updating problem in an estimation and regression framework, we compute the optimum weights that will be used to iteratively refine the effects of different feature components in the database search. Given probabilistic models of similarity described in previous chapters, Chapter 7 proposes a unified framework to combine multiple feature vectors and similarity models. Different classifier combination methods and a naive Bayesian network classifier are described and are used to compute similarity and incorporate relevance feedback in a Bayesian framework. Extensive classification and retrieval experiments with both quantitative and qualitative performance evaluation on three groundtruthed databases are presented in Chapter 8. The performances of the proposed models are compared to the performances of two competing algorithms from the content-based retrieval literature. The dissertation concludes with Chapter 9 where the proposed algorithms are summarized and future research directions are discussed.

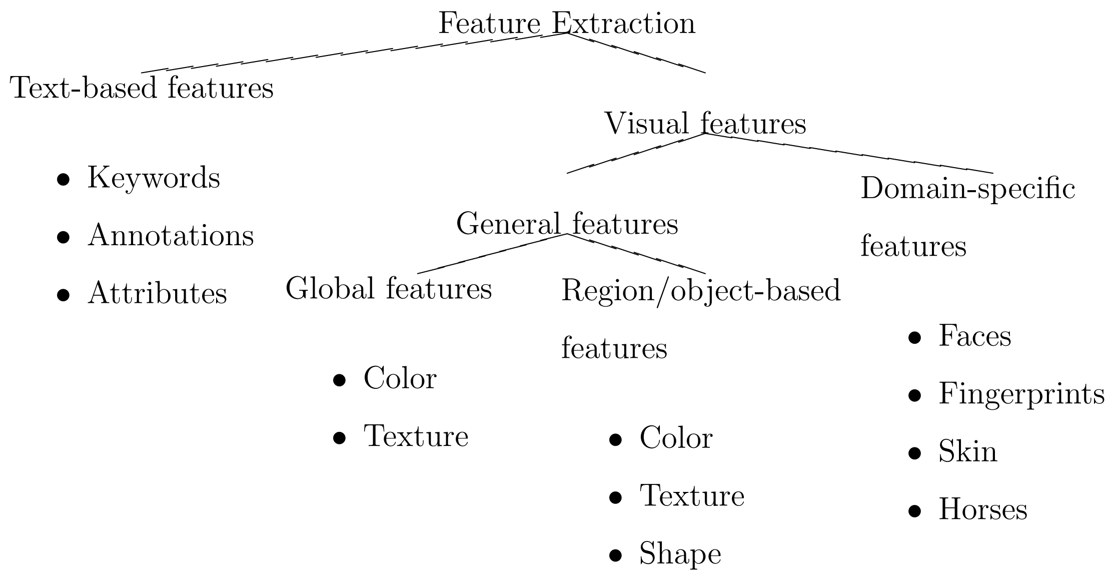
Two of the texture feature extraction methods in Chapter 3 were published in [2, 3, 4, 7, 9]. Parts of Chapters 3 and 4 were published in [8, 10]. Part of Chapter 5 was published in [5, 6]. Part of Chapter 6 was published in [11]. Part of Chapter 4 and materials in Chapters 7 and 8 are in preparation for publication.

Chapter 2

LITERATURE REVIEW

2.1 Feature Extraction

Initial work on content-based retrieval focused on using low-level features like color and texture for image representation. Feature extraction methods in the literature can be divided into the following categories:



As being one of the first approaches that use color for image retrieval, Swain and Ballard [192] proposed to use color histograms. Stricker and Orengo [191] used color moments with the motivation that most of the information in a histogram can be summarized using low-order moments. Androutsos *et al.* [12] used the angles between the average color vectors, and Smith and Li [189] used a color-based co-occurrence approach for color-based image retrieval. Gevers and Smeulders [83] developed color models that were invariant to the viewpoint, geometry of the object and illumination conditions.

Texture has been one of the most important characteristics which have been used to classify and recognize objects and scenes [88, 90, 197]. In the MIT Photobook Project, Pentland *et al.* [160] used 2-D Wold-based decompositions that were described in [133] to measure periodicity, directionality and randomness as texture descriptions in the Texture Photobook. In the Los Alamos National Lab.'s CANDID Project, Kelly *et al.* [116] used Laws' texture energy maps to extract textural features from pulmonary CT images and introduced a global signature based on a sum of weighted Gaussians to model the texture. They also used these Gaussian distributions to visualize which pixels contribute more to the similarity score. In [117] they applied these methods to LANDSAT TM data. Barros *et al.* [17] tried to retrieve multi-spectral satellite images by first clustering image pixels according to their spectral values using a modified k-means clustering procedure, then using the spectral distribution information as features for each connected region. Jacobs *et al.* [102] used Haar wavelet decompositions and a distance measure that compared how many wavelet coefficients that two images had in common for image retrieval. They used only a few significant wavelet coefficients and also quantized them to improve the speed of the system. Manjunath and Ma [136] used Gabor filter-based multiresolution representations to extract texture information. They used means and standard deviations of Gabor transform coefficients, computed at different scales and orientations, as features. Gabor filters performed better than the pyramid-structured wavelet transform, tree-structured wavelet transform and the multiresolution simultaneous autoregressive model (MR-SAR) in the tests performed on the Brodatz database. In [133], Liu and Picard treated images as 2-D homogeneous random fields and used the Wold theory to decompose them into three mutually orthogonal components. These components corresponded to the perceptually important "periodicity", "directionality" and "randomness" properties. They compared the features that they compute from the 2-D Wold model to other models, namely the shift-invariant principal component analysis, the multiresolution simultaneous autoregressive model, the tree-structured wavelet transform and Tamura *et al.*'s [194] features that were used in [74]. The Wold-based features performed better than others in terms of average recall for a Brodatz texture dataset. Li *et al.* [131] used 21 different spatial features like gray level differences (mean, contrast, angular second moments, directional derivatives, etc.), co-occurrence matrices, moments, autocorrelation functions, fractals and Robert's gradient on the Brodatz image set and on remote sensing images. The spatial features they extracted outperformed some transform-based features like the discrete cosine transform, Gabor filters, quadrature mirror filters and uniform subband coding. Zhou *et al.* [213] used a water-filling algorithm to extract structural information from edge maps.

Systems that use shape information include MIT's Photobook Project [160] which used the Karhunen-Loeve transform to select eigenvectors to represent variations from the prototypical appearance as appearance-specific descriptions in the Appearance Photobook and modeled the connections in a shape using stiffness matrices produced by the finite element method as shape descriptions in the Shape Photobook. Rui *et al.* [169] proposed a modified Fourier descriptor which was both robust to noise and invariant to geometric transformations. Moment invariants have also been used as region-based moments which are invariant to transformations based on Hu's work [95]. Del Bimbo *et al.* [28] retrieved images containing specified 2-D shapes using an elastic matching technique. Nastar [149] used the image shape spectrum for shape-based retrieval. Liu and Sclaroff [134] first over-segmented images and then tried to find groupings of these segments to fit deformable shape models. The fitting cost function included a region color compatibility term, region/model area overlap term, and a deformation term. They presented examples on banana, leaf and fish images.

Some researchers tried to develop features for a specific application domain. These include the face features by Bach *et al.* [16], fingerprint minutiae patterns by Jain and Hong [103], and the skin color features by Fleck *et al.* [73]. The latter system developed a procedure for finding naked people in color images using a flesh filter that looked for large areas of skin color. They further used a geometric analyzer to group regions in certain spatial relationships that were typical of limbs connected to torsos. Jain *et al.* [104] used both face and fingerprint features to prune the database in an automatic person identification system.

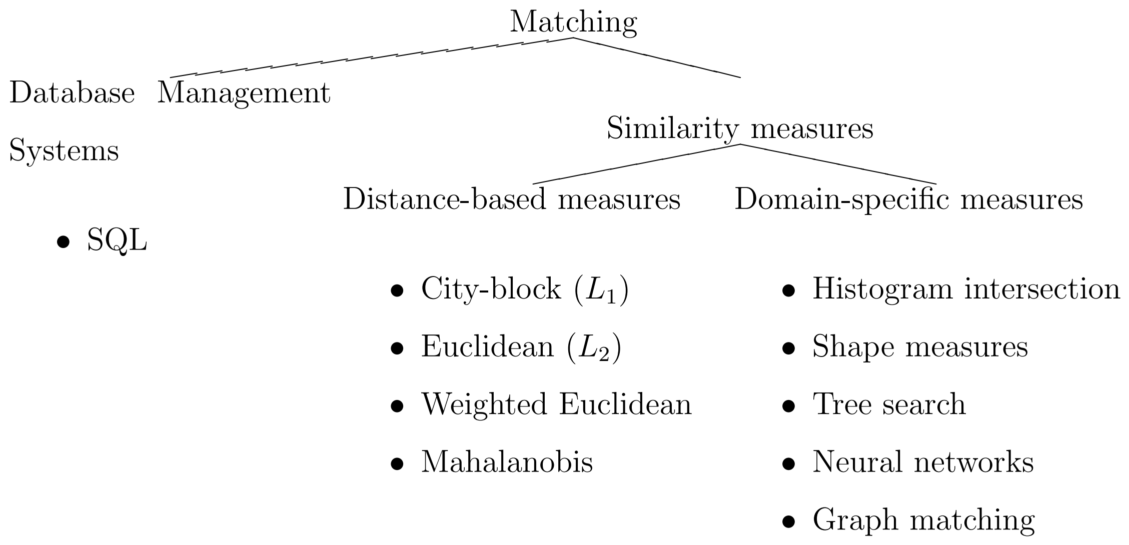
2.2 Region-Based Query Systems

More recent approaches developed region-based query systems which involve image segmentation based on color and texture but the region segmentation algorithms are still too slow to be used in an image retrieval application. Requirements for segmentation accuracy are quite different for shape features and other features. For shape features, an accurate segmentation is highly desirable but a coarse segmentation may be sufficient to find regions to compute other features.

Rui *et al.* [169] proposed a segmentation algorithm based on clustering and grouping in spatial color-texture space. The user defined where the object of interest was and the algorithm tried to group regions into meaningful objects. Carson *et al.* [38] developed a region-based query system called "Blobworld" by first grouping pixels into regions based on color and texture using expectation-maximization and minimum description length principles, then by describing these regions using color, texture, location and shape properties. Ma and Manjunath [135] described a system called "Netra" that also used color, texture, shape and spatial location information. They developed an "edge flow model" that identified the direction of changes in the feature values to segment the image into non-overlapping segments and computed the color, texture, shape and location information for each region. The two latter approaches performed automatic segmentation. Other approaches for image segmentation include graph-based approaches [184, 185, 71, 72] and a non-parametric clustering approach [156].

2.3 Image Matching

After each image or region is associated with a feature vector, the next major step is to use measures to find similarities between images. Researchers have used different methods to match the query image to one or more of the database images. These methods can be divided into the following categories:



Text-based retrieval systems with relational databases [69] used SQL queries to find matches between images [193, 183]. The main disadvantage of SQL queries is that they can only support exact matches between manually assigned keywords.

In systems where images are represented by feature vectors, distance-based methods are used as similarity measures to perform content-based retrieval. A distance measure is used to rank the database images in ascending order of their distances to the query image, which is assumed to correspond to a descending order of similarity. Each image is assumed to be represented as a point by its feature vector in the high-dimensional feature space and the nearest neighbor rule is used to retrieve images. Popular distance measures have been the Euclidean (L_2) distance [74, 160, 131, 188], the city-block (L_1) distance [136, 188], the general Minkowsky L_p distance [178], the weighted Euclidean distance [19, 172], the Cauchy distance [180], and the Mahalanobis distance [160, 188]. Some systems which include domain-specific features used similarity measures that are specific to those features. These include histogram intersection [188], Chamfer measure for shape matching [31], Hausdorff distance for object matching [168], tree search, self organizing maps [212] and graph matching [98].

The nearest neighbor rule is straightforward given the feature vectors but it is limited in the sense that it cannot make use of training data for higher-level similarities and features cannot always map visually similar images into nearby locations in the feature space. Moghaddam and Pentland [146] first projected image feature vectors to a subset of their principal components, then estimated the distribution of these feature vectors using a multivariate Gaussian or a mixture of Gaussians, and finally used these estimates to perform maximum likelihood detection of faces, facial features and hands. In [145], they based similarity on image difference vectors, used principal components projection to reduce dimensionality, defined the class-conditional densities to be Gaussian, and used the *a posteriori* probabilities as similarity measures for face recognition. They showed that the probabilistic approach was much more powerful than the geometric distance-based approach.

Recently, Rubner *et al.* [167] used the earth mover's distance which computed the distance between two distributions that were represented by signatures. The signatures were sets of weighted features that captured the distributions. The earth mover's distance was defined as the minimum amount of work needed to change one signature into the other. The notion of "work" was based on the user-defined ground distance which was the distance between two feature vectors. Useful properties of the earth mover's distance were that the sizes of two signatures could be different and also the sum of weights of one signature could be different than the sum of weights of the other, i.e. it allowed partial matches.

2.4 Feature Combination

It soon became apparent that no single feature vector can achieve significant performance for all images. Some systems allowed the user to query the database using multiple feature vectors. One simple method to combine multiple feature vectors was to append different vectors and treat the result as a big global feature vector [131]. In the IBM's QBIC Project, Niblack *et al.* [74] used features like color, texture and shape that were computed for each object in an image as well as for each image. In [13], they developed semi-automatic tools to aid manual outlining of the objects during database population. The Virage image search engine [15] allowed retrieval based on color, composition, texture, and structure measures. Jain and Vailaya [107] described a system for content-based retrieval of trademark images using a weighted combination of color and shape features. Smith [188] developed a system that used color, texture and spatial location information for image retrieval. Vailaya and Jain [201, 200] compared the effectiveness of different features like color histogram, color coherence vector, discrete cosine transform coefficients, edge direction histogram and edge direction coherence vector in classifying images into two classes: city and landscape, using a weighted nearest neighbor classifier. Edge-based features performed better than the others. They suggested building a hierarchical classifier that uses multiple two-class classifiers for image grouping. Berman and Shapiro [24] developed the Flexible Image Database System to perform fast indexing using linear or Boolean combinations of multiple color and texture feature vectors.

Vailaya *et al.* [200] used a hierarchical classifier to first classify images into the city or landscape classes using edge information, then to classify landscape images into forests, mountains and sunset/sunrise classes using color information. Dy *et al.* [68] used a similar approach by two-level classifiers. They tried to classify a query using features that best differentiate the major classes and then specialized the query to that class by using the features that best distinguish the images within the chosen major class.

Neural networks have also been used as a tool for feature combination. Haering *et al.* [87, 86] used a neural network with features like color, roughness, directionality, co-occurrence features, Fourier features, Gabor features and fractal features as input and trained it to detect deciduous trees. Oh *et al.* [152, 153] also used neural network classifiers to combine different features in a handwriting recognition application.

Bayesian networks were also used as another class of algorithms that used classifiers to combine different features. Schroder *et al.* [177] used a naive Bayesian classifier to link user interests and signal models to iteratively learn user-specific land cover types in a remote sensing image archive. Kumar and Desai [124] used a discrete variable Bayesian network to identify types of the segments of an image. Vasconcelos and Lippman [204] used the fact that movie production usually has specific conventions and structure, and used a Bayesian framework to incorporate this structure in video summarization and classification.

Recently, Tieu and Viola [196] used boosting for image retrieval. They first extracted a very large number of highly selective features and then used boosting to train classifiers that used only a small subset of these features according to positive and negative query images.

A more detailed review about feature combination is given in Chapter 7.

2.5 Relevance Feedback

Recently, relevance feedback has been a popular technique for improving retrieval performance. It was first applied as a post-processing method in the document retrieval literature [175]. The main idea is to include the human user in the retrieval loop. An initial search in the database is done using the original query input by the user. Upon being presented the results of this search, the user labels some of the results as relevant and irrelevant according to his/her information needs. The goal is to incorporate this feedback information into the database search in terms of iterative retrievals. Possible methods for relevance feedback can be summarized as:

Post-processing

Text-based retrieval

- Term selection
- Term weighting
- Creating user model

Feature-based retrieval

- Vector space model
- Choosing
  - Features
  - Similarity measures
  - Search engines
- Weighting
  - Features
  - Similarity measures
- Others
  - Self organizing maps
  - Creating user model
  - Reorganizing retrieval results
  - Feature density estimation

In the document retrieval literature, relevance feedback is usually done by modifying the input keywords (also called terms or attributes) according to the keywords associated with the relevant documents. Crestani [53] used neural networks to perform relevance feedback. The goal was to learn from the users' feedback and modify the query terms. The input of the neural network was the original query terms and the terms pertinent to the set of relevant documents, and the output was the terms of the modified query. These terms were selected from the set of all possible terms which were ranked based on the weights obtained from the output of a 3-layer feed-forward neural network trained using the back-propagation learning algorithm. Biron *et al.* [29] investigated two symbolic approaches to information retrieval: Genetic algorithms where the query was represented as a weighted Boolean function of the terms, and an approach which tried to use knowledge of users' past browsing behavior to enhance the representation of documents in order to influence future retrieval. The set of relevant documents, selected by the user, were used by Ribeiro-Neto and Assis [162] to tune the original query. This was achieved by formulating the modified query as a conjunction of the original query and the pertinent attributes extracted from the relevant set. The newly formulated query was then normalized through transformation to its disjunctive form. Finally the negated terms were eliminated to avoid instability of the system. Deogun *et al.* [57] used "user-oriented clustering" where documents which were judged as jointly relevant (marked together as relevant in user feedback) frequently by users were placed in the same cluster. This approach also used keywords for document representation. In [25], they introduced a graph-based representation where the nodes of the graph were the clusters and the arcs between the nodes represented the weights. The data was clustered in a given number of clusters based on the minimization of the sum of weights, which were measures of similarities between the clusters. The weights were modified with each accumulated information from the user. Voigt [208] presented a document browsing agent that assumed that documents that were viewed long enough were relevant for the user so that they would be more easily accessible in the next browsing session. Here the relevancy was defined between documents and users, not between documents themselves. The main disadvantage of this approach was that it heavily depended on the assumption that "information that was relevant at some point in time would still be important in the future", and manually set thresholds on viewing time were used to determine relevancy. Croft [54] described a document retrieval system where the relevance feedback used "term selection" (choosing keywords from relevant documents to add to the list of original keywords) and "term weighting" (assigning relative importance to them).

Although some of these approaches performed effectively for document retrieval, they are not directly applicable to content-based image retrieval. The complexity of the visual characterization of images and the large number of images in a database make manual keyword assignment process infeasible. However, initial work on relevance feedback for image retrieval tried to apply techniques from information (specifically document) retrieval literature. One of the most commonly used approaches is the vector space model [173, 47, 82] where a new query vector is formed as a weighted linear combination of the feature vectors of the original query, images in the relevance set (with a positive weight) and images in the irrelevance set (with a negative weight). Rui *et al.* [173] applied well established text retrieval techniques to image retrieval. Two factors, “component importance” and “inverse collection importance”, were proposed for images in accordance to the factors “term frequency” and “inverse document frequency” in text retrieval. The vector space model was used for relevance feedback. They also used Gaussian normalization to put equal emphasis to each feature component, and then used the inverse of the standard deviation of each component for the images in the relevant feedback set as weights. They concluded that the approach adopted from text retrieval performed better than Gaussian normalization but the latter was more robust to unknown feature components. Chua *et al.* [47] described relevance feedback techniques for color-based image retrieval. First, they found significant colors in the images, then, they selected the ones that frequently occur in relevant images fed back by the user. These selected colors were used as the new query. They also used color coherence histograms as feature vectors and formed new queries as weighted combinations of the color coherence histograms for the original query and the relevant images (vector space model). Gevers and Smeulders [82] also used the vector space model.

Keywords for each image are also used [186, 52] in the same way they are used in document retrieval. The new query keywords are formed by taking the union of all the keywords for the original query and the images in the retrieval set. The keywords in the irrelevance set are sometimes removed from this large set of keywords. Simonnot and Smail [186] used a model where a video was decomposed into three levels and keywords were assigned to each level. User gave positive and negative weights to these predefined keywords. Boolean queries were performed for pre-selection in the database and final matching was done using the vector space model. Cox *et al.* [52] used hidden semantic attributes in the PicHunter system. The attributes were called "hidden" because the user did not specify any keywords during searching, they were just used by the system for image similarity. The main disadvantage of this system was that the attributes had to be assigned to each image manually.

Some systems made use of the feedback by switching between multiple feature vectors and/or similarity measures according to the comparisons between the rankings of the system and the rankings of the user. Rui *et al.* [174] developed a system that, given a set of similarity measures, selected the similarity measure which minimized the sum of the differences between the ranks of the retrieved images and the ranks of the relevant images selected by the user. They also proposed an alternative to hard switching between different similarity measures; using a weighted sum of all similarity measures. Benitez *et al.* [20] used relevance feedback to rank features of search engines and later to select the set of features from a given search engine that gave the best results for a given query. All the search engines were initially given similar weights and then these weights were increased or decreased based on the images they retrieved and their relevance to the user.

Another popular approach is to give weights to each feature and/or similarity measure and update the weights according to the positive and negative feedback from the user [172, 159, 170]. The MARS system [172] supported a multimedia object model where multiple image representations with dynamically updated weights were used. An object was represented in terms of visual features such as color, texture, and shape in one level, and specific implementations of these feature types such as color histograms, co-occurrence matrices, Fourier descriptors in the next level. The system computed the overall similarity between images using a weighted linear combination of different representations at each level. Weights at each level were updated independently according to the user feedback in terms of positive and negative scores for each image. Rui and Huang [170] criticized that weight updating was usually done heuristically and formulated an optimization problem by minimizing the weighted sum of distances for the training images. They used the generalized Euclidean distance (a Mahalanobis-like distance but a weight matrix was used instead of the covariance matrix) for feature vector distances. Then, a weighted sum of distances was used as the final similarity. As solutions to the optimization problem, the new query vector is the weighted average of the training feature vectors, the new weight matrix was the inverse of the weighted covariance matrix, and the new weights for the distances were inversely proportional to the total distance of the training images. Their experiments on COREL images showed that this formulation performed better than the vector space model and another approach that used the generalized Euclidean distance.

Other approaches include using self organizing feature maps [143], creating a probabilistic user model [51, 50, 155], using a rule-based model [32], modifying the distance measure [43], reorganizing the retrieval results [195, 44, 42] and feature density estimation [150, 141]. Minka and Picard [143] used machine learning in terms of self organizing feature maps to automatically select and combine available features based on positive and negative examples from the user. PicHunter [51] used a series of display/action iterations to help the user find a specific target image in the database. A Bayesian framework was developed, where the probability that each image in the database being equal to the target image was updated after every iteration. A user model to compute these probabilities was constructed according to the experiments with different subjects. Performance was measured in terms of the number of iterations required to find the specific target image. In [50], the feedback was in the form of relative judgments between displayed images. Using this feedback, the system tried to minimize the amount of iterations to find the target image using a nondeterministic comparison searching algorithm. Park and Lee [155] used relevance feedback to capture the users' interest and then to create a user profile. The user profile was used to filter the documents in the database. Bouet and Djeraba [32] used a rule based approach to select the discriminatory features according to the user feedback to refine the query. Conditional probabilities and implication intensities were used to evaluate the rules in the form "What is the probability to get 'b' proposition true when 'a' proposition is true?". The main disadvantage was that the rules depended on manually set thresholds. Chen *et al.* [43] used relevance feedback to prune and reorganize the search space. The reorganization was done by a modification of the distance measure in a way to take into consideration the relevant set (worm hole distance). After this reorganization, the search space became warped around the relevant set in such a way that the distance between two documents from the relevant set in this space was zero. In [195, 44], they clustered the relevant shots in a video were to find natural groupings of them. A dendogram (binary tree) was formed from these shots, which was then cut to produce a fixed number of clusters. The most similar shots to the centroids of the clusters were found and were presented using the similarity pyramid proposed in [42]. Nastar *et al.* [150] tried to estimate the distribution of relevant images from the examples provided by the user by simultaneously minimizing the probability of retrieving irrelevant images. The density estimation was done by first assuming that the feature components were independent and had Gaussian distributions. Then, the parameters for each Gaussian were determined according to an error term defined as a combination of relevant images not covered and irrelevant images covered by the Gaussian. A new query feature vector was generated by drawing random values from the estimated densities for each feature. In [141], they used likelihood values to retrieve images where the class-conditional distributions for the relevance and irrelevance classes were estimated using a Parzen window estimator with a Gaussian kernel at the points that corresponded to images that were labeled relevant and irrelevant by the user. Experiments were shown on the Columbia database and feedback improved results over iterations.

Relevance feedback is currently a very hot topic in image retrieval. Researchers are trying to develop feedback methods to combine information from different feature vectors and improve the performance. Practical database retrieval systems also involve other issues like graphical user interface design and indexing problems in very large databases. Barros *et al.* [18] investigated the effect of triangle inequality using single keys and pairs of keys in reducing the number of comparisons to search the database. Berman and Shapiro [23] first extended the triangle inequality to multiple distance measures and then investigated the performances of different key selection algorithms like random selection, selection according to density variance, selection according to separation, a greedy thresholding algorithm and clustering. Data structures like hash tables, k-d trees, R-trees, R^+-trees, R^*-trees, SS-trees, SR-trees, MVP-trees and M-trees for indexing are also available [187, 183]. Additional literature review about specific problems are given in the following chapters.

Chapter 3

FEATURE EXTRACTION AND NORMALIZATION

3.1 Introduction

As discussed in Section 1.4, the first level of the retrieval process is image representation as pre-processing. Low-level features have been the most popular image representation methods because they are usually well-defined and therefore can be easily computed from raw image data. As reviewed in Chapter 2, many features have been proposed in the literature and complex image retrieval systems use features that are generated by many different feature extraction algorithms with different kinds of sources. However, not all of these features have the same range. Popular distance measures, for example the Euclidean distance, implicitly assign more weighting to features with large ranges than those with small ranges. Feature normalization is required to approximately equalize ranges of the features and make them have approximately the same effect in the computation of similarity. In most of the database retrieval literature, the normalization methods were usually not mentioned or only the Gaussian assumption was used [136, 131, 150, 172]. The Mahalanobis distance [63] also involves normalization in terms of the covariance matrix and produces results related to likelihood when the features have a Gaussian distribution.

This chapter discusses five normalization methods; linear scaling to unit range, linear scaling to unit variance, transformation using the cumulative distribution function, rank normalization and normalization by fitting distributions. The goal is to independently normalize each feature component to the [0,1] range. Effectiveness of different normalization methods are measured in two ways. As being part of the classification framework proposed in Section 1.4, this chapter presents results using the class separability criterion. In the following chapter, effectiveness will be investigated in combination with different similarity measures. A normalization method is preferred over the others according to these empirical results. Even though there is no single best normalization method for all databases, normalization after fitting distributions is usually among the best and class separability appears to be an effective measure for choosing the normalization method that gives the best retrieval performance.

In this dissertation we use low-level features computed globally for the whole image as its representation. A more complex representation can be to use segmentation methods to find hypothetically meaningful regions in the image and base the queries on these regions. Our main goal is to develop similarity methods so only global low-level features are considered in the rest of this dissertation. However, all of the algorithms that will be proposed can be directly applicable to features computed from regions.

The rest of the chapter first summarizes the features we use, then, describes the details of the normalization methods, and finally, presents experiments on choosing the best normalization methods for different feature vectors for the given databases.

3.2 Feature Extraction

Each image is represented by multiple feature vectors in our system. The following texture and color features are implemented:

1. Line-angle-ratio statistics (LAR)

   Line-angle-ratio statistics [2, 4, 7, 9] use a texture histogram computed from the spatial relationships between lines as well as the properties of their surroundings. Spatial relationships are represented by the angles between intersecting line pairs and properties of the surroundings are represented by the ratios of the mean gray levels inside and outside the regions spanned by those angles. The texture histogram is computed by first performing vector quantization to form non-uniform cells in the feature space, and then by counting the number of angle-ratio pairs that are assigned to each cell. Line-angle-ratio statistics result in a 20-dimensional feature vector.

2. Co-occurrence variances (COOC)

   Variances of gray level spatial dependencies [2, 3, 4, 7, 9] use second-order statistics of gray levels of pixels and are computed from the co-occurrence matrices for different spatial relationships. First, co-occurrence matrices are computed for five different pixel distances and four orientations, then, the variances (also called contrast) are computed from these matrices to form a 20-dimensional feature vector.

3. Gabor features (GABOR)

   A successful application of Gabor texture features were used in the University of California at Santa Barbara's Netra project [136]. The image is first filtered by a Gabor filter bank of 5 scales and 6 orientations, and then the means and variances of the filtered image values are computed to form a 60-dimensional feature vector.

4. Moments features (MOMENTS)

   Moments features were used by a group at Tampere University of Technology as textural features for image retrieval [41]. The image is first filtered by moment filters of up to 3rd order (which makes 9 2-dimensional filters), and then a 36-dimensional vector of the means, variances, medians and absolute median deviations of the filtering results are used for image representation.

5. Tamura features (TAMURA)

Tamura's texture features were used in IBM's QBIC project [74]. These features were developed as computational approximations to the visual texture properties found to be important in psychology studies. QBIC uses a 4-dimensional vector which consists of the coarseness, contrast, directionality and dominant orientation features.

6. Color histograms (COLHIST)

   Color histogram has been the most commonly used color feature representation. It estimates the joint probability of the intensities of the three color channels [192]. We use the HSV color space [84] and compute a $3 \times 3 \times 3$ histogram which results in a 27-dimensional feature vector for each image in the VisTex Database. A $4 \times 4 \times 4$ histogram is used as a 64-dimensional feature vector for the COREL Database.

Alternatively, all the features computed using the feature extraction algorithms listed above can be collected in a big feature vector and an optimum partition of this vector can be used as sub-vectors. This approach is computationally expensive but an optimum partition found as part of the classification problem can improve the overall results. In the rest of the dissertation, the feature vectors computed as above are used for image representation but all of the proposed algorithms are directly applicable for alternative partitions of feature vectors.

3.3 Feature Normalization

The following methods can be used to independently normalize each feature component to the [0,1] range.

3.3.1 Linear Scaling to Unit Range (Norm.1)

Given a lower bound l and an upper bound u for a feature component $x \in \mathbb{R}$,

$$\tilde{x} = \frac{x - l}{u - l} \tag{3.1}$$

results in $\tilde{x}$ being in the [0,1] range.

A problem may arise when the unnormalized feature vector of a query image has one or more of the feature values out of the $[u, l]$ range. To avoid an out of range [0,1] after normalization, we can truncate the out-of-range values to either 0 or 1. Therefore, the normalization rule for the mapping $x \in [l, u] \rightarrow \tilde{x} \in [0, 1]$ becomes

$$\tilde{x} = \begin{cases} 0 & \text{if} \quad x < l \\ \frac{x-l}{u-l} & \text{if} \quad l \leq x \leq u \\ 1 & \text{if} \quad x > u. \end{cases} \tag{3.2}$$

3.3.2 Linear Scaling to Unit Variance (Norm.2)

Another normalization procedure is to transform the feature component $x \in \mathbb{R}$ to a random variable with zero mean and unit variance as

$$\tilde{x} = \frac{x - \mu}{\sigma} \tag{3.3}$$

where μ and σ are the sample mean and the sample standard deviation of that feature respectively [105].

If we assume that each feature has a Gaussian distribution, the probability of $\tilde{x}$ being in the [-1,1] range is 68%. An additional shift and rescaling as

$$\tilde{x} = \frac{\frac{x-\mu}{3\sigma} + 1}{2} \tag{3.4}$$

guarantees 99% of $\tilde{x}$ to be in the [0,1] range. We can then truncate the out-of-range components to either 0 or 1.

3.3.3 Transformation Using the Cumulative Distribution Function (Norm.3)

Given a random variable $x \in \mathbb{R}$ with cumulative distribution function $F_x(x)$, the random variable $\tilde{x}$ resulting from the transformation $\tilde{x} = F_x(x)$ will be uniformly distributed in the [0,1] range [154].

3.3.4 Rank Normalization (Norm.4)

Given the sample for a feature component for all images as $x_1, \dots, x_n \in \mathbb{R}$, first we find the order statistics $x^{(1)}, \dots, x^{(n)}$ and then replace each image's feature value by its corresponding normalized rank as

$$\tilde{x}_i = \frac{\operatorname*{rank}_{x_1,\dots,x_n}(x_i) - 1}{n-1} \tag{3.5}$$

where x_i is the feature value for the i'th image. This procedure uniformly maps all feature values to the [0,1] range. When there are more than one image with the same feature value, especially after quantization, they are assigned the average rank for that value.

3.3.5 Normalization After Fitting Distributions (Norm.5)

The transformations in Section 3.3.2 assumed that a feature has a Gaussian(μ, σ^2) distribution. The sample values can be used to find better estimates for the feature distributions. Then, these estimates can be used to find normalization methods based particularly on these distributions. Huang and Mumford [96] also tried to fit distributions to single pixel statistics, derivative statistics, joint pixel statistics and joint wavelet feature statistics using the minimum mean square error as the fitting criterion. However, they only tried to develop models and did not use this information in normalization.

The following sections describe how to fit Gaussian, Lognormal, Exponential and Gamma densities to a random sample. We also give the difference distributions because the image similarity measures use feature differences. After estimating the parameters of a distribution, the cut-off value that includes 99% of the feature values is found and the sample values are scaled and truncated so that each feature component have the same range. Given the cut-off value δ_x, the normalization rule for the mapping $x \in [0, \infty) \rightarrow \tilde{x} \in [0, 1]$ is

$$\tilde{x} = \begin{cases} \frac{x}{\delta_x} & \text{if} \quad 0 \leq x \leq \delta_x \\ 1 & \text{if} \quad x > \delta_x. \end{cases} \tag{3.6}$$

Since the original feature values are positive, we use only the positive section of the Gaussian density after fitting. Lognormal, Exponential and Gamma densities are defined for random variables with only positive values. Other distributions that are commonly encountered in the statistics literature are the Uniform, χ^2 and Weibull (which are special cases of Gamma), Beta (which is defined only for [0,1]) and Cauchy (whose moments do not exist). Although these distributions can also be used by first estimating the parameters and then finding the cut-off values, we will show that the distributions used in this chapter can quite generally model features from different feature extraction algorithms.

To measure how well a fitted distribution resembles the sample data (goodness-of-fit), we use the Kolmogorov-Smirnov test statistic [36, 161] which is defined as the maximum value of the absolute difference between the cumulative distribution function estimated from the sample and the one calculated from the fitted distribution. After estimating the parameters for different distributions, we compute the Kolmogorov-Smirnov statistic for each distribution and choose the one with the smallest value as the best fit to our sample. This is done for each feature component independently.

Fitting a Gaussian(μ, σ^2) Density

Let $x_1, \ldots, x_n \in \mathbb{R}$ be a random sample from a population with density $\frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$, $-\infty < x < \infty,\ -\infty < \mu < \infty,\ \sigma > 0$. The likelihood function for the parameters μ and σ^2 is

$$L(\mu, \sigma^2 | x_1, \ldots, x_n) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\sum_{i=1}^n (x_i - \mu)^2 / 2\sigma^2}. \tag{3.7}$$

To find the values of μ and σ^2 that maximize this likelihood function, we take its logarithm as

$$\log L(\mu, \sigma^2 | x_1, \ldots, x_n) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^n (x_i - \mu)^2 \tag{3.8}$$

and equate the derivatives to 0 as

$$\frac{\partial}{\partial \mu} \log L(\mu, \sigma^2 | x_1, \ldots, x_n) = \frac{1}{\sigma^2} \sum_{i=1}^n (x_i - \mu) = 0 \tag{3.9}$$

$$\frac{\partial}{\partial \sigma^2} \log L(\mu, \sigma^2 | x_1, \ldots, x_n) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=1}^n (x_i - \mu)^2 = 0. \tag{3.10}$$

Then, the maximum likelihood estimators (MLE) of μ and σ^2 can be derived as

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i \qquad \text{and} \qquad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})^2. \tag{3.11}$$

The cut-off value δ_x that includes 99% of the feature values can be found as

$$\begin{gathered} P(x \leq \delta_x) = P\left(\frac{x - \hat{\mu}}{\hat{\sigma}} \leq \frac{\delta_x - \hat{\mu}}{\hat{\sigma}} \right) = 0.99 \\ \Longrightarrow \delta_x = \hat{\mu} + 2.4\hat{\sigma}. \end{gathered} \tag{3.12}$$

Let x and y be two i.i.d. random variables with a Gaussian(μ, σ^2) distribution. Using moment generating functions, we can easily show that their difference $z = x - y$ has a Gaussian($0, 2\sigma^2$) distribution. The cut-off value δ_z that includes 99% of the feature differences can be found as

$$\begin{gathered} P(|z| \leq \delta_z) = P\left(\left| \frac{z}{\sqrt{2}\hat{\sigma}} \right| \leq \frac{\delta_z}{\sqrt{2}\hat{\sigma}} \right) = 0.99 \\ \Longrightarrow \delta_z = 3\sqrt{2}\hat{\sigma}. \end{gathered} \tag{3.13}$$

Fitting a Lognormal(μ, σ^2) Density

Let $x_1, \dots, x_n \in \mathbb{R}$ be a random sample from a population with density $\frac{1}{\sqrt{2\pi}\sigma} \frac{e^{-(\log x - \mu)^2/2\sigma^2}}{x}$, $x \geq 0,\ -\infty < \mu < \infty,\ \sigma > 0$. The likelihood function for the parameters μ and σ^2 is

$$L(\mu, \sigma^2 | x_1, \dots, x_n) = \frac{1}{(2\pi\sigma^2)^{n/2}} \frac{e^{-\sum_{i=1}^n (\log x_i - \mu)^2/2\sigma^2}}{\prod_{i=1}^n x_i}. \tag{3.14}$$

To find the values of μ and σ^2 that maximize this likelihood function, we take its logarithm as

$$\log L(\mu, \sigma^2 | x_1, \dots, x_n) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^n (\log x_i - \mu)^2 - \sum_{i=1}^n \log x_i \tag{3.15}$$

and equate the derivatives to 0 as

$$\frac{\partial}{\partial\mu} \log L(\mu, \sigma^2 | x_1, \dots, x_n) = \frac{1}{\sigma^2}\sum_{i=1}^n (\log x_i - \mu) = 0 \tag{3.16}$$

$$\frac{\partial}{\partial\sigma^2} \log L(\mu, \sigma^2 | x_1, \dots, x_n) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^n (\log x_i - \mu)^2 = 0. \tag{3.17}$$

Then, the MLEs of μ and σ^2 can be derived as

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^n \log x_i \qquad \text{and} \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^n (\log x_i - \hat{\mu})^2. \tag{3.18}$$

In other words, we can take the natural logarithm of each sample point and treat the new data as a sample from a Gaussian(μ, σ^2) distribution [39].

The cut-off value δ_x that includes 99% of the feature values can be found as

$$\begin{aligned} P(x \leq \delta_x) = P(\log x \leq \log \delta_x) = P\left(\frac{\log x - \hat{\mu}}{\hat{\sigma}} \leq \frac{\log \delta_x - \hat{\mu}}{\hat{\sigma}}\right) = 0.99 \\ \implies \delta_x = e^{\hat{\mu} + 2.4\hat{\sigma}}. \end{aligned} \tag{3.19}$$

Fitting an Exponential(λ) Density

Let $x_1, \dots, x_n \in \mathbb{R}$ be a random sample from a population with density $\frac{1}{\lambda}e^{-x/\lambda}$, $x \geq 0,\ \lambda > 0$. The likelihood function for the parameter λ is

$$L(\lambda | x_1, \dots, x_n) = \frac{1}{\lambda^n} e^{-\sum_{i=1}^n x_i/\lambda}. \tag{3.20}$$

To find the value of λ that maximizes this likelihood function, we take its logarithm as

$$\log L(\lambda|x_1,\ldots,x_n) = -n\log\lambda - \frac{1}{\lambda}\sum_{i=1}^{n} x_i \tag{3.21}$$

and equate its derivative to 0 as

$$\frac{\partial}{\partial\lambda}\log L(\lambda|x_1,\ldots,x_n) = -\frac{n}{\lambda} + \frac{1}{\lambda^2}\sum_{i=1}^{n} x_i = 0. \tag{3.22}$$

Then, the MLE of λ can be derived as

$$\hat{\lambda} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{3.23}$$

The cut-off value δ_x that includes 99% of the feature values can be found as

$$\begin{aligned} P(x \le \delta_x) = 1 - e^{-\delta_x/\hat{\lambda}} = 0.99 \\ \Longrightarrow \delta_x = -\hat{\lambda}\log 0.01. \end{aligned} \tag{3.24}$$

Let x and y be two i.i.d. random variables with an Exponential(λ) distribution. Their joint distribution is

$$p_{xy}(x,y) = \frac{1}{\lambda^2}e^{-(x+y)/\lambda}, \qquad x \ge 0,\ y \ge 0. \tag{3.25}$$

To find the distribution of their difference $z = x - y$, we define a second random variable as $w = x$. Then the joint distribution of z and w becomes

$$p_{zw}(z,w) = p_{xy}(w, w-z) = \frac{1}{\lambda^2}e^{-(2w-z)/\lambda}, \qquad z \le w,\ w \ge 0. \tag{3.26}$$

The marginal distribution of z can be found as

$$\begin{aligned} p_z(z) &= \int_{-\infty}^{\infty} p_{zw}(z,w)dw \\ &= \int_{\max\{0,z\}}^{\infty} \frac{1}{\lambda^2}e^{-(2w-z)/\lambda}dw \\ &= \begin{cases} \frac{1}{2\lambda}e^{z/\lambda} & \text{if } z < 0 \\ \frac{1}{2\lambda}e^{-z/\lambda} & \text{if } z \ge 0 \end{cases} \\ &= \frac{1}{2\lambda}e^{-|z|/\lambda}, \qquad -\infty < z < \infty. \end{aligned} \tag{3.27}$$

Similar to the previous case, the MLE of λ can be derived as

$$\hat{\lambda} = \frac{1}{n}\sum_{i=1}^{n}|z_i|. \tag{3.28}$$

The cut-off value δ_z that includes 99% of the feature differences can be found as

$$\begin{aligned} P(|z| \leq \delta_z) = 1 - e^{-\delta_z/\hat{\lambda}} &= 0.99 \\ \Longrightarrow \delta_z &= -\hat{\lambda}\log 0.01. \end{aligned} \tag{3.29}$$

Fitting a Gamma(α, β) Density

Let $x_1, \ldots, x_n \in \mathbb{R}$ be a random sample from a population with density $\frac{1}{\Gamma(\alpha)\beta^\alpha}x^{\alpha-1}e^{-x/\beta}$, $x \geq 0,\ \alpha, \beta > 0$. Since closed forms for the MLEs of the parameters α and β do not exist[1], we use the method of moments (MOM) estimators [39]. The first two sample moments are

$$m_1 = \frac{1}{n}\sum_{i=1}^{n}x_i \qquad \text{and} \qquad m_2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2. \tag{3.30}$$

To find the MOM estimates for α and β, we equate the first two population moments

$$E[x] = \alpha\beta \qquad \text{and} \qquad E[x^2] = \alpha\beta^2 + \alpha^2\beta^2 \tag{3.31}$$

to the sample moments and solve the following system of equations

$$\frac{1}{n}\sum_{i=1}^{n}x_i = \alpha\beta \tag{3.32}$$

$$\frac{1}{n}\sum_{i=1}^{n}x_i^2 = \alpha\beta^2 + \alpha^2\beta^2. \tag{3.33}$$

The MOM estimators for α and β can be derived as

$$\hat{\alpha} = \frac{\left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2}{\left(\frac{1}{n}\sum_{i=1}^{n}x_i^2\right) - \left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2}, \tag{3.34}$$

$$\hat{\beta} = \frac{\left(\frac{1}{n}\sum_{i=1}^{n}x_i^2\right) - \left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2}{\left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)} \tag{3.35}$$

[1]MLEs of Gamma parameters can be derived in terms of the "Digamma" function and can be computed numerically [36, 161].

and can be easily computed using the sample mean and the sample variance.

It can be shown [39] that when $x \sim$ Gamma(α, β) with an integer α, $P(x \leq \delta_x) = P(y \geq \alpha)$ where $y \sim$ Poisson(δ_x/β). Then the cut-off value δ_x that includes 99% of the feature values can be found as

$$\begin{aligned} P(x \leq \delta_x) = \sum_{y=\hat{\alpha}}^{\infty} e^{-\delta_x/\hat{\beta}} \frac{(\delta_x/\hat{\beta})^y}{y!} &= 1 - \sum_{y=0}^{\hat{\alpha}-1} e^{-\delta_x/\hat{\beta}} \frac{(\delta_x/\hat{\beta})^y}{y!} = 0.99 \\ \Longrightarrow \sum_{y=0}^{\hat{\alpha}-1} e^{-\delta_x/\hat{\beta}} \frac{(\delta_x/\hat{\beta})^y}{y!} &= 0.01. \end{aligned} \tag{3.36}$$

Johnson *et al.* [112] represents equation (3.36) as

$$P(x \leq \delta_x) = e^{-\delta_x/\hat{\beta}} \sum_{j=0}^{\infty} \frac{(\delta_x/\hat{\beta})^{\hat{\alpha}+j}}{\Gamma(\hat{\alpha}+j+1)} \tag{3.37}$$

Another way to find δ_x is to use the Incomplete Gamma function [1, p.260],[161, sec. 6.2] as

$$P(x \leq \delta_x) = I_{\delta_x/\hat{\beta}}(\hat{\alpha}). \tag{3.38}$$

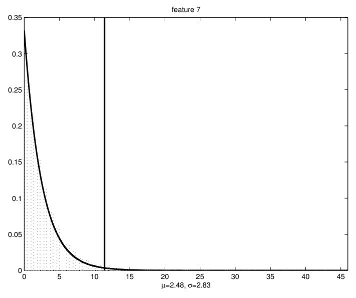
Note that $\hat{\alpha}$ does not have to be an integer in (3.38).

Let x and y be two i.i.d. random variables with a Gamma(α, β) distribution. The distribution of $z = x - y$ can be found as [190, p.356]

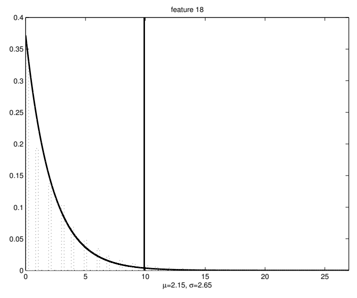
$$p_z(z) = \frac{z^{(2\alpha-1)/2}}{(2\beta)^{(2\alpha-1)/2}} \frac{1}{\pi^{1/2}} \frac{1}{\beta\Gamma(\alpha)} K_{\alpha-1/2}(z/\beta) \ , \quad -\infty < z < \infty \tag{3.39}$$

where $K_m(u)$ is the modified Bessel function of the second kind of order m ($m \geq 0$, integer) [190, p.419],[161, sec. 6.6].

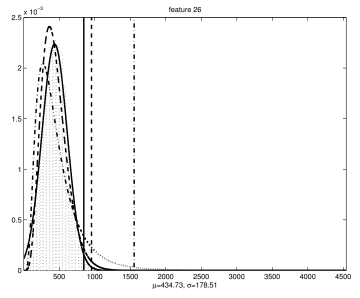
Histograms and fitted distributions for some example features are given in Figure 3.1. Each feature is modeled by its best fit according to the Kolmogorov-Smirnov statistic. These plots show that many features from different feature extraction algorithms can be modeled by the distributions that we presented in Section 3.3.5.



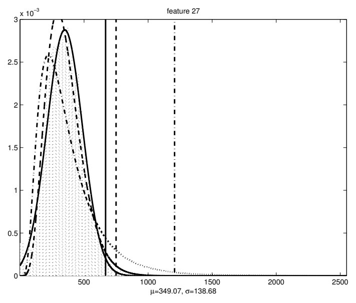
(a) Line-angle-ratio (Best fit: Exponential)



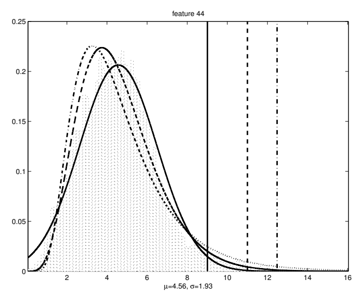
(b) Line-angle-ratio (Best fit: Exponential)



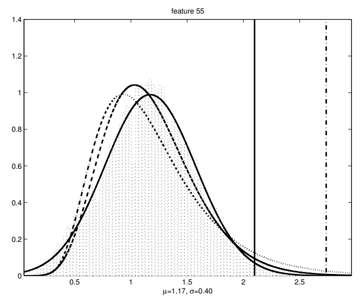
(c) Co-occurrence (Best fit: Normal)



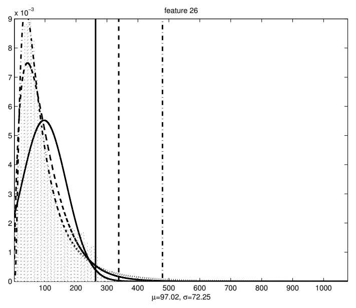
(d) Co-occurrence (Best fit: Normal)



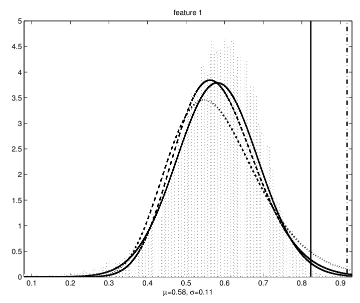
(e) Gabor (Best fit: Gamma)



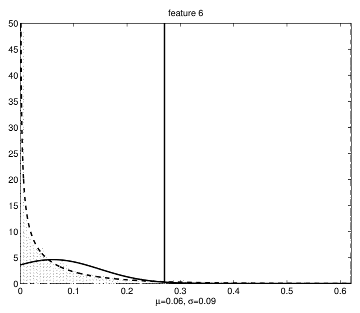
(f) Gabor (Best fit: Normal)



(g) Moments (Best fit: Lognormal)



(h) Tamura (Best fit: Normal)



(i) Color histogram (Best fit: Gamma)

Figure 3.1: Feature histograms and fitted distributions for example features. An Exponential model (solid line) is used for the line-angle-ratio features and Normal (solid line), Lognormal (dash-dot line) and Gamma (dashed line) models are used for others. Vertical lines show the 99% cut-off point for each distribution.

3.4 Choosing the Best Normalization Method

We use two empirical methods to choose the best performing normalization method for a particular data set. The first one compares retrieval performances which are evaluated using average precision and recall computed from the groundtruth. The second one is based on class separability. Two classical approaches to linear data transformations, the principal components analysis and the discriminant analysis, use scatter matrices to find projections that efficiently represent or efficiently separate the data respectively. Therefore, scatter information is an important measure of class separability [64, 78].

Assume we have n data vectors $\mathbf{x_1}, \dots, \mathbf{x_n} \in \mathbb{R}^{(q \times 1)}$ and m classes $\mathcal{C}_1, \dots, \mathcal{C}_m$. Let $n_1, \dots, n_m$ be the number of data vectors assigned to these m classes. A within-class scatter matrix measures the scatter of data vectors around their respective class means. Given a scatter matrix $\mathbf{S_i} \in \mathbb{R}^{(q \times q)}$ for the i'th class $\mathcal{C}_i$ as

$$\mathbf{S_i} = \sum_{\mathbf{x} \in \mathcal{C}_i} (\mathbf{x} - \boldsymbol{\mu_i})(\mathbf{x} - \boldsymbol{\mu_i})^T \tag{3.40}$$

where $\boldsymbol{\mu_i} = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{C}_i} \mathbf{x}$, the within-class scatter matrix $\mathbf{S_W}$ is computed as

$$\mathbf{S_W} = \sum_{i=1}^{m} \mathbf{S_i}. \tag{3.41}$$

On the other hand, the between-class scatter matrix $\mathbf{S_B} \in \mathbb{R}^{(q \times q)}$ is the scatter of the class means around the total mean

$$\mathbf{S_B} = \sum_{i=1}^{m} n_i (\boldsymbol{\mu_i} - \boldsymbol{\mu_0})(\boldsymbol{\mu_i} - \boldsymbol{\mu_0})^T \tag{3.42}$$

where $\boldsymbol{\mu_0}$ is the total mean and is given by $\boldsymbol{\mu_0} = \frac{1}{n} \sum_{i=1}^{m} n_i \boldsymbol{\mu_i}$. The total scatter matrix $\mathbf{S_T} \in \mathbb{R}^{(q \times q)}$ is the scatter of all data vectors regardless of their class assignments and is computed as

$$\mathbf{S_T} = \sum_{\mathbf{x}} (\mathbf{x} - \boldsymbol{\mu_0})(\mathbf{x} - \boldsymbol{\mu_0})^T = \mathbf{S_W} + \mathbf{S_B}. \tag{3.43}$$

We need a scalar measure of the "size" of a scatter matrix. The simplest scalar measure of a scatter matrix is its trace. The trace measures the square of the scattering radius because it is proportional to the sum of the variances in the coordinate directions. One possible measure is the trace of $\mathbf{S_W}$. Similarly, the determinant measures the square of the scattering volume since it is proportional to the product of the variances in the directions of the principal axis. Hence, another possible measure is the determinant of $\mathbf{S_W}$. Both measures often give the same results but the latter is invariant to changes in the scale of the axes, so it is preferable over the former [64]. It is also known that the eigenvalues of $\mathbf{S_W}^{-1}\mathbf{S_B}$ are invariant under nonsingular linear transformations of the data. Therefore, an appealing criterion becomes

$$\varsigma = \ln |\mathbf{S_W}^{-1}(\mathbf{S_W} + \mathbf{S_B})| \tag{3.44}$$

where $|\cdot|$ represents the determinant. ς gives a number which is large when the between-class scatter is large and the within-class scatter is small. ($\mathbf{S_W}^{-1}\mathbf{S_B}$ cannot be used directly because $\mathbf{S_B}$ will be singular if either m or $n - m$ is less than or equal to the dimensionality.)

3.5 Experiments

The normalization methods described in Section 3.3 were used to normalize the components of the feature vectors in all databases and class separability was computed for each case. The results are given in Table 3.1. In these experiments, each groundtruth group was considered a class for a particular database. The results show that there was no single best method. However, normalization after fitting distributions was usually among the best three methods. Other experiments that used all possible fitted distributions (not only the best fits) showed that the ordering of fitting-based normalization methods according to class separability was also consistent with their ordering according to fitting accuracy in terms of the Kolmogorov-Smirnov statistic. We will show in the following chapter that these results are also consistent with the

Table 3.1: Class separability for each normalization method where each groundtruth group is considered a class. The normalization methods used in the experiments are linear scaling to unit range (Norm.1), linear scaling to unit variance (Norm.2), transformation using the cumulative distribution function (Norm.3), rank normalization (Norm.4), and normalization after fitting distributions (Norm.5). The best method, i.e. the one resulting in the largest class separability, for each feature vector for each database is marked by a box and will be used as the normalization method for that feature vector in the rest of the dissertation.

		Class separability $\ln \lvert \mathbf{S_W}^{-1}(\mathbf{S_W}+\mathbf{S_B})\rvert$				
Database	Feature	Norm.1	Norm.2	Norm.3	Norm.4	Norm.5
ISL	LAR+COOC	$\boxed{7.5744}$	7.3806	6.5429	6.5879	7.5110
	GABOR	10.7486	10.7645	10.1630	10.1749	$\boxed{10.9418}$
	MOMENTS	5.5497	$\boxed{5.5781}$	5.1550	5.1566	5.5718
	TAMURA	$\boxed{2.8758}$	2.8664	2.7003	2.6926	$\boxed{2.8758}$
VisTex	LAR+COOC	$\boxed{28.4957}$	26.7647	22.0479	21.9172	27.3390
	GABOR	42.3886	41.9468	43.0962	$\boxed{43.1182}$	42.0865
	MOMENTS	18.4746	18.2974	15.5114	15.4498	$\boxed{18.6321}$
	TAMURA	5.7108	$\boxed{5.8555}$	5.7230	5.7498	5.6916
	COLHIST	35.6390	39.7095	37.2992	$\boxed{40.5325}$	35.6190
COREL	LAR+COOC	6.4210	6.7886	6.4931	6.2059	$\boxed{6.9714}$
	GABOR	8.7609	8.9270	8.6802	8.6691	$\boxed{8.9490}$
	MOMENTS	5.7522	$\boxed{5.7815}$	5.2468	5.2597	5.7577
	TAMURA	2.4694	$\boxed{2.4809}$	2.3717	2.0291	2.4693
	COLHIST	12.2795	$\boxed{13.8496}$	13.6783	12.4768	12.2848

retrieval performance results. Therefore, class separability appears to be an effective measure for choosing the normalization method that gives the best retrieval performance and this fact strengthens our motivation for a classification-based framework for image retrieval. The best normalization method for each feature vector for each database will be used as the normalization method for that vector in the rest of the dissertation.

Chapter 4

SIMILARITY MEASURES

4.1 Introduction

Initial work on content-based retrieval focused on using low-level features like color and texture for image representation. After each image is associated with a feature vector and these vectors are normalized, distance measures that compute distances between these feature vectors are used to find similarities between images with the assumption that images that are close to each other in the feature space are also visually similar. A distance measure is used to rank the database images in ascending order of their distances to the query image, which is assumed to correspond to a descending order of similarity.

A similarity measure for content-based retrieval should be efficient enough to match similar images as well as being able to discriminate dissimilar ones. Feature vectors usually exist in a very high-dimensional space. Due to this high dimensionality, their parametric characterization is usually not studied, and non-parametric approaches like the nearest neighbor rule are used for retrieval. In geometric similarity measures like the nearest neighbor rule, no assumption is made about the probability distribution of the features and similarity is based on the distances between feature vectors in the feature space. Given this fact, Euclidean distance has been the most widely used distance measure [74, 160, 131, 188]. Other popular measures have been the weighted Euclidean distance [19, 172], the city-block (L_1) distance [136, 188], the general Minkowsky L_p distance [178], the Cauchy distance [180], and the Mahalanobis distance [160, 188]. The L_1 distance was also used under the name "histogram intersection" [188]. Berman and Shapiro [21] used polynomial combinations of predefined distance measures to create new distance measures. Moghaddam and Pentland [146, 145] used multivariate Gaussians to estimate the feature vector or feature difference vector distributions in the principal components space and used these estimates to perform maximum likelihood detection of faces, facial features and hands, and maximum *a posteriori* detection for face recognition. They showed that probabilistic similarity performed much better than the geometric similarity with distances.

This chapter presents a probabilistic approach for image retrieval with the motivation and problem definition given in Section 1.4. We describe likelihood-based similarity measures that compute the likelihood of two images being similar or dissimilar, one being the query image and the other one being an image in the database. First, we define two classes, the relevance class and the irrelevance class, and then derive the likelihood values from a Bayesian classifier. We use three different methods to estimate the class-conditional probabilities used in the classifier. The first method uses a multivariate Gaussian assumption, the second one uses independently fitted distributions for each feature, and the last one uses mixtures of Gaussians. The performances of these methods are compared to the performances of the commonly used geometric approaches in the form of the L_p metric (e.g., city-block (L_1) and Euclidean (L_2) distances) in ranking the images in the database. We also describe a classification-based criterion to select the best performing p for the L_p metric.

This probabilistic setting can also be interpreted as a mapping from the high-dimensional feature space to the two-dimensional probability space. We examine the effects of operating in the feature space versus operating in the probability space by training nine different classifiers in both spaces. This corresponds to a two-level modeling of probability. In the first level, class-conditional probabilities are computed for the feature vectors. Since these probabilities are only estimates of the true probabilities, the classifiers trained in the probability space implicitly perform a second level modeling of probabilities to compensate for errors in modeling probabilities in the feature space. Similarity in the probability space corresponds to likelihoods of the two classes and similarity in the feature space corresponds to distances between feature vectors. We show that the probabilistic similarity measures perform significantly better than the geometric similarity measures, and simple linear classifiers can be effectively trained in the two-dimensional probability space while complex non-linear classifiers have to be designed in the high-dimensional feature space.

The rest of the chapter is organized as follows. First, the probability models for computing the likelihood values are described in Section 4.2. Then, the nearest neighbor rule with the Minkowsky L_p metric is described in Section 4.3. Section 4.4 discusses operating in the feature space versus operating in the probability space. Finally, extensive experiments for both classification and retrieval performance evaluation are given in Section 4.5. The effects of the normalization methods that were described in Chapter 3 are also studied in coordination with the similarity measures described in this chapter.

4.2 Probabilistic Similarity Measures

As mentioned in Section 1.4, the choice for the Bayes classifier comes from the fact that it minimizes the classification error given that we know the true class-conditional distributions and the prior probabilities. However, these distributions are not exactly known in practice but can be estimated from training data.

Equation (1.5) defined the likelihood ratio as the discriminant function for classification. Given two images with q-dimensional feature vectors $\mathbf{x}$ and $\mathbf{y}$, and their feature difference vector $\mathbf{d} = \mathbf{x} - \mathbf{y} \in \mathbb{R}^{(q \times 1)}$, the likelihood ratio is defined as

$$\Delta(\mathbf{d}) = \frac{p(\mathbf{d}|\mathcal{A})}{p(\mathbf{d}|\mathcal{B})}. \tag{4.1}$$

In the following sections, we describe three models to estimate the class-conditional distributions. The motivation for estimating $p(\mathbf{d}|\mathcal{A})$ and $p(\mathbf{d}|\mathcal{B})$ in terms of feature difference vectors comes from the assumption that similarity between images can be based on the closeness of their feature values, i.e. similar images have similar feature values (therefore, difference vectors have a small scatter) and dissimilar images have relatively different feature values (difference vectors have a large scatter).

4.2.1 Multivariate Gaussian (MVG)

We assume that the feature difference vectors for the relevance class have a multivariate Gaussian density with mean $\boldsymbol{\mu}_{\mathcal{A}}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathcal{A}}$, i.e.

$$p(\mathbf{d}|\mathcal{A}) = p(\mathbf{d}|\boldsymbol{\mu}_{\mathcal{A}}, \boldsymbol{\Sigma}_{\mathcal{A}}) = \frac{1}{(2\pi)^{q/2}|\boldsymbol{\Sigma}_{\mathcal{A}}|^{1/2}} e^{-(\mathbf{d}-\boldsymbol{\mu}_{\mathcal{A}})^T \boldsymbol{\Sigma}_{\mathcal{A}}^{-1} (\mathbf{d}-\boldsymbol{\mu}_{\mathcal{A}})/2}. \tag{4.2}$$

Similarly, the feature difference vectors for the irrelevance class are assumed to have a multivariate Gaussian density with sample mean $\boldsymbol{\mu}_{\mathcal{B}}$ and sample covariance matrix $\boldsymbol{\Sigma}_{\mathcal{B}}$, i.e.

$$p(\mathbf{d}|\mathcal{B}) = p(\mathbf{d}|\boldsymbol{\mu}_{\mathcal{B}}, \boldsymbol{\Sigma}_{\mathcal{B}}) = \frac{1}{(2\pi)^{q/2}|\boldsymbol{\Sigma}_{\mathcal{B}}|^{1/2}} e^{-(\mathbf{d}-\boldsymbol{\mu}_{\mathcal{B}})^T \boldsymbol{\Sigma}_{\mathcal{B}}^{-1} (\mathbf{d}-\boldsymbol{\mu}_{\mathcal{B}})/2}. \tag{4.3}$$

The likelihood ratio in Equation (4.1) becomes

$$\Delta(\mathbf{d}) = \frac{p(\mathbf{d}|\boldsymbol{\mu}_{\mathcal{A}}, \boldsymbol{\Sigma}_{\mathcal{A}})}{p(\mathbf{d}|\boldsymbol{\mu}_{\mathcal{B}}, \boldsymbol{\Sigma}_{\mathcal{B}})}. \tag{4.4}$$

Given training feature difference vectors $\mathbf{d_1}, \ldots, \mathbf{d_n} \in \mathbb{R}^{(q\times 1)}$ for a class, the sample mean and the sample covariance for that class can be computed using the multivariate versions of the MLEs given in Section 3.3.5 as

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{d}_{\boldsymbol{i}} \qquad \text{and} \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{d}_{\boldsymbol{i}} - \hat{\boldsymbol{\mu}})(\mathbf{d}_{\boldsymbol{i}} - \hat{\boldsymbol{\mu}})^T. \tag{4.5}$$

To simplify the computation of the likelihood ratio in Equation (4.4), we take its logarithm, eliminate some constants, and use

$$\Delta'(\mathbf{d}) = (\mathbf{d} - \boldsymbol{\mu}_{\mathcal{A}})^T \boldsymbol{\Sigma}_{\mathcal{A}}^{-1} (\mathbf{d} - \boldsymbol{\mu}_{\mathcal{A}}) - (\mathbf{d} - \boldsymbol{\mu}_{\mathcal{B}})^T \boldsymbol{\Sigma}_{\mathcal{B}}^{-1} (\mathbf{d} - \boldsymbol{\mu}_{\mathcal{B}}) \tag{4.6}$$

to rank the database images in ascending order of these values which corresponds to a descending order of similarity. This ranking is equivalent to ranking in descending order using the likelihood values in Equation (4.4).

4.2.2 Independently Fitted Distributions (FIT)

A common assumption with data in high dimensions is to treat feature components as independent. When this assumption is used, marginal distributions can be fitted to individual feature components and their multiplication gives the joint likelihood values.

Let f be a feature difference component with training values $f_1, \ldots, f_n \in \mathbb{R}$. Since we are using feature differences, we need symmetric distributions[1] with both positive and negative ranges. Among the distributions [70] that satisfy this criterion, we fit f either a Gaussian distribution $\frac{1}{\sqrt{2\pi}\sigma} e^{-(f-\mu)^2/2\sigma^2}$, $-\infty < \mu < \infty,\ \sigma > 0$, with maximum likelihood estimates (MLEs)

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} f_i \qquad \text{and} \qquad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (f_i - \hat{\mu})^2, \tag{4.7}$$

or a Double Exponential distribution (a.k.a. the Laplace distribution) $\frac{1}{2\lambda} e^{-|f-\mu|/\lambda}$, $-\infty < \mu < \infty,\ \lambda > 0$, with MLEs

$$\hat{\mu} = \text{median}(f_1, \ldots, f_n) \qquad \text{and} \qquad \hat{\lambda} = \frac{1}{n} \sum_{i=1}^{n} |f_i - \hat{\mu}|, \tag{4.8}$$

or a Logistic distribution $\frac{e^{-(f-\mu)/\tau}}{\tau(1+e^{-(f-\mu)/\tau})^2}$, $-\infty < \mu < \infty,\ \tau > 0$, with method of moments estimators (MOMs)

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} f_i \qquad \text{and} \qquad \hat{\tau}^2 = \frac{3}{\pi^2} \left(\frac{1}{n} \sum_{i=1}^{n} f_i^2 - \hat{\mu}^2 \right) \tag{4.9}$$

according to the Kolmogorov-Smirnov statistic described in Section 3.3.5. We observed that most of the best fits were Gaussians. This can be explained by a weak application of the Central Limit Theorem which states that the summation of a number of independent random variables is asymptotically Gaussian. If the random variables

[1]Similarity relationship is symmetric. I.e. if image ξ_i is similar to image ξ_j, image ξ_j is equally similar to image ξ_i. Therefore, we are using symmetric training and testing image pair sets in the experiments (see Section 1.5.2).

are i.i.d. with a smooth distribution, a very small number of terms in the summation is sufficient for a good approximation [154]. Although the number of variables in our case is only two, a Gaussian was chosen as the best fit for most of the cases.

After the fitted distributions are obtained for a feature difference vector $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_q)^T \in \mathbb{R}^{(q \times 1)}$, the joint distribution for the relevance class is given as

$$\begin{aligned} p(\mathbf{d}|\mathcal{A}) &= p(\mathbf{d}|\mu_{\mathcal{A}t}, \sigma^2_{\mathcal{A}t}, t \in \mathcal{T}_{\mathcal{A}}, \mu_{\mathcal{A}u}, \lambda_{\mathcal{A}u}, u \in \mathcal{U}_{\mathcal{A}}, \mu_{\mathcal{A}v}, \tau_{\mathcal{A}v}, v \in \mathcal{V}_{\mathcal{A}}) \\ &= \prod_{t \in \mathcal{T}_{\mathcal{A}}} \frac{1}{\sqrt{2\pi\sigma^2_{\mathcal{A}t}}} e^{-\frac{(\mathbf{d}_t - \mu_{\mathcal{A}t})^2}{2\sigma^2_{\mathcal{A}t}}} \prod_{u \in \mathcal{U}_{\mathcal{A}}} \frac{1}{2\lambda_{\mathcal{A}u}} e^{-\frac{|\mathbf{d}_u - \mu_{\mathcal{A}u}|}{\lambda_{\mathcal{A}u}}} \prod_{v \in \mathcal{V}_{\mathcal{A}}} \frac{e^{-(\mathbf{d}_v - \mu_{\mathcal{A}v})/\tau_{\mathcal{A}v}}}{\tau_{\mathcal{A}v}(1 + e^{-(\mathbf{d}_v - \mu_{\mathcal{A}v})/\tau_{\mathcal{A}v}})^2} \end{aligned} \tag{4.10}$$

where $\mathcal{T}_{\mathcal{A}}, \mathcal{U}_{\mathcal{A}}$ and $\mathcal{V}_{\mathcal{A}}$ are the sets of indices for components with best fits as Gaussians, Double Exponentials and Logistics respectively. Similarly, the joint distribution for the irrelevance class is given as

$$\begin{aligned} p(\mathbf{d}|\mathcal{B}) &= p(\mathbf{d}|\mu_{\mathcal{B}t}, \sigma^2_{\mathcal{B}t}, t \in \mathcal{T}_{\mathcal{B}}, \mu_{\mathcal{B}u}, \lambda_{\mathcal{B}u}, u \in \mathcal{U}_{\mathcal{B}}, \mu_{\mathcal{B}v}, \tau_{\mathcal{B}v}, v \in \mathcal{V}_{\mathcal{B}}) \\ &= \prod_{t \in \mathcal{T}_{\mathcal{B}}} \frac{1}{\sqrt{2\pi\sigma^2_{\mathcal{B}t}}} e^{-\frac{(\mathbf{d}_t - \mu_{\mathcal{B}t})^2}{2\sigma^2_{\mathcal{B}t}}} \prod_{u \in \mathcal{U}_{\mathcal{B}}} \frac{1}{2\lambda_{\mathcal{B}u}} e^{-\frac{|\mathbf{d}_u - \mu_{\mathcal{B}u}|}{\lambda_{\mathcal{B}u}}} \prod_{v \in \mathcal{V}_{\mathcal{B}}} \frac{e^{-(\mathbf{d}_v - \mu_{\mathcal{B}v})/\tau_{\mathcal{B}v}}}{\tau_{\mathcal{B}v}(1 + e^{-(\mathbf{d}_v - \mu_{\mathcal{B}v})/\tau_{\mathcal{B}v}})^2} \end{aligned} \tag{4.11}$$

where $\mathcal{T}_{\mathcal{B}}, \mathcal{U}_{\mathcal{B}}$ and $\mathcal{V}_{\mathcal{B}}$ are the sets of indices for components with best fits as Gaussians, Double Exponentials and Logistics respectively. The likelihood ratio in Equation (4.1) becomes

$$\Delta(\mathbf{d}) = \frac{p(\mathbf{d}|\mu_{\mathcal{A}t}, \sigma^2_{\mathcal{A}t}, t \in \mathcal{T}_{\mathcal{A}}, \mu_{\mathcal{A}u}, \lambda_{\mathcal{A}u}, u \in \mathcal{U}_{\mathcal{A}}, \mu_{\mathcal{A}v}, \tau_{\mathcal{A}v}, v \in \mathcal{V}_{\mathcal{A}})}{p(\mathbf{d}|\mu_{\mathcal{B}t}, \sigma^2_{\mathcal{B}t}, t \in \mathcal{T}_{\mathcal{B}}, \mu_{\mathcal{B}u}, \lambda_{\mathcal{B}u}, u \in \mathcal{U}_{\mathcal{B}}, \mu_{\mathcal{B}v}, \tau_{\mathcal{B}v}, v \in \mathcal{V}_{\mathcal{B}})}. \tag{4.12}$$

Instead of computing the complete likelihood ratio, we take its logarithm, eliminate some constants, and use

$$\begin{aligned}\Delta'(\mathbf{d}) = \frac{1}{2}\sum_{t\in\mathcal{T}_\mathcal{A}} \frac{(\mathbf{d}_t-\mu_{\mathcal{A}t})^2}{\sigma^2_{\mathcal{A}t}} + \sum_{u\in\mathcal{U}_\mathcal{A}} \frac{|\mathbf{d}_u-\mu_{\mathcal{A}u}|}{\lambda_{\mathcal{A}u}} + \\ \sum_{v\in\mathcal{V}_\mathcal{A}} \left[\frac{(\mathbf{d}_v-\mu_{\mathcal{A}v})}{\tau_{\mathcal{A}v}} + 2\log(1+e^{-(\mathbf{d}_v-\mu_{\mathcal{A}v})/\tau_{\mathcal{A}v}})\right] - \\ \frac{1}{2}\sum_{t\in\mathcal{T}_\mathcal{B}} \frac{(\mathbf{d}_t-\mu_{\mathcal{B}t})^2}{\sigma^2_{\mathcal{B}t}} - \sum_{u\in\mathcal{U}_\mathcal{B}} \frac{|\mathbf{d}_u-\mu_{\mathcal{B}u}|}{\lambda_{\mathcal{B}u}} - \\ \sum_{v\in\mathcal{V}_\mathcal{B}} \left[\frac{(\mathbf{d}_v-\mu_{\mathcal{B}v})}{\tau_{\mathcal{B}v}} + 2\log(1+e^{-(\mathbf{d}_v-\mu_{\mathcal{B}v})/\tau_{\mathcal{B}v}})\right]\end{aligned} \tag{4.13}$$

to rank the database images.

The assumption of independent features is used quite often to simplify computations although features are usually correlated. Whitening [78] is a linear transform that can be used to make the components of a vector uncorrelated with variances equal to unity. The eigenvalue decomposition of the sample covariance matrix can be used for whitening. After finding the eigenvalues $\lambda_1,\ldots,\lambda_q$ and eigenvectors $\boldsymbol{\nu_1},\ldots,\boldsymbol{\nu_q}$ of the covariance matrix $\mathbf{\Sigma}$ of the random variable $\mathbf{x}\in\mathbb{R}^{(q\times 1)}$, the transformation

$$\mathbf{y} = \mathbf{\Lambda}^{-1/2}\mathbf{\Phi}^T(\mathbf{x}-\boldsymbol{\mu}) \tag{4.14}$$

with $\boldsymbol{\mu}$ being the sample mean of $\mathbf{x}$, $\Lambda = \text{diag}(\lambda_1,\ldots,\lambda_q)$ being the diagonal eigenvalue matrix and $\mathbf{\Phi} = (\boldsymbol{\nu_1},\ldots,\boldsymbol{\nu_q})$ being the eigenvector matrix such that $\mathbf{\Sigma\Phi} = \mathbf{\Phi\Lambda}$, results in the random variable y with zero mean and unit covariance matrix.

The distribution of each uncorrelated feature component can then be estimated after applying the whitening transform[2]. Uncorrelatedness can be considered as a weaker form of independence because independence implies uncorrelatedness but the reverse is not true unless the random variable is a Gaussian. In other words, uncorrelatedness corresponds to independence in second order statistics.

[2]A special case of this is the multivariate Gaussian when all components are fit marginal Gaussians because a multivariate Gaussian is invariant to orthogonal transformations.

Another linear transform, called the Independent Component Analysis (ICA) [99, 64], tries to find the axes on which data is independent in as many statistical orders as possible. ICA can be formulated in terms of the model

$$\mathbf{x} = \mathbf{M}\mathbf{y} + \boldsymbol{\mu} \tag{4.15}$$

where the components of the random variable $\mathbf{x}$ (i.e. the observed variables) are assumed to be generated from the components of the random variable $\mathbf{y}$ (i.e. the latent variables) according to the mixing matrix $\mathbf{M}$. $\boldsymbol{\mu}$ is the mean of the observed variables. Given the observations for $\mathbf{x}$, the goal is to estimate $\mathbf{M}$ and $\mathbf{y}$. Then, the transformation $\mathbf{y} = \mathbf{M}^{-1}(\mathbf{x} - \boldsymbol{\mu})$ can be used and the joint distribution can be estimated by multiplying the estimates of the marginal distributions of the components of $\mathbf{y}$.

ICA algorithms differ in terms of the criterion function to measure the independence of the components. The unmixing matrix $\mathbf{M}^{-1}$ can then be found by minimizing the given criterion. Welling and Weber [209] modeled the distributions of independent components with one-dimensional mixtures of Gaussians and developed an Expectation-Maximization (EM) algorithm to estimate both the mixing matrix and the mixture parameters.

Hyvarinen and Oja [100] related independentness to non-Gaussianity. In the case of Gaussian variables, we can estimate the mixing matrix up to an orthogonal transformation because Gaussian random variables are invariant to orthogonal transformation and the resulting symmetric Gaussian densities do not contain any information of higher than second-order. To maximize the non-Gaussianity of the variables under transformation, they suggested using measures like kurtosis (which is zero for Gaussian variables but is non-zero for most of the others), negentropy (because a Gaussian variable has the largest entropy among all random variables with equal variances), and approximations of negentropy using cosine-hyperbolic or some other exponential functions. Another possible criterion to measure independence is the minimization of mutual information which is equivalent to the Kullback-Leibler distance between the joint density and the product of the marginal densities. Whitening can be used as a pre-processing step both for reducing the number of parameters to be estimated and for dimensionality reduction.

We used the FastICA algorithm [100] that is based on fixed-point iterations for finding a maximum of non-Gaussianity which is measured by an approximation of negentropy using tangent-hyperbolic functions. However, the results were not better than the case where the distributions for individual components were estimated without applying the transformation. Even though ICA was successfully applied to particular signals like sound signals [209], and in problems like blind source separation, financial data analysis and electroencephalogram (EEG) data [99, 100], it did not perform well for our high-dimensional feature vectors because of overfitting problems. Hyvarinen *et al.* [101] discussed a similar problem of overfitting where different ICA algorithms produced estimates of the source signals that were zero everywhere except single spikes and bumps when the training samples were insufficient. They pointed out that this problem is much more likely to occur if the source signals are not i.i.d. in time and have strong time-dependencies. In the experiments in Section 4.5, we estimate the distributions of individual feature components using normalized raw data without any pre-processing.

4.2.3 Mixtures of Gaussians (GMIX)

Parametric density estimation methods assume a specific form for the density function, like a multivariate Gaussian in Section 4.2.1, and the problem reduces to finding the estimates of the parameters of this specific form. However, the assumed form can be quite different from the true density. On the other hand, non-parametric approaches usually require a large amount of training data and computations can be quite demanding when the data size increases. To combine the advantages of both parametric and non-parametric approaches, we can use mixture models [30] which are semi-parametric models that are not restricted to a particular form but also have a fixed number of parameters independent of the size of the data set. This section describes a combination of the Expectation-Maximization (EM) and Minimum Description Length (MDL) algorithms for the estimation of mixtures of Gaussians.

Let $\mathcal{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ be a set of vectors independent and identically distributed (i.i.d.) in $\mathbb{R}^{(q\times 1)}$ with distribution $p(\mathbf{x}|\boldsymbol{\Theta})$. The likelihood function for the parameter set $\boldsymbol{\Theta}$ is

$$L(\boldsymbol{\Theta}|\mathcal{X}) = p(\mathcal{X}|\boldsymbol{\Theta}) = \prod_{i=1}^{n} p(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\Theta}). \tag{4.16}$$

The maximum likelihood estimate of $\boldsymbol{\Theta}$ is the one that maximizes $L(\boldsymbol{\Theta}|\mathcal{X})$, or often $\log L(\boldsymbol{\Theta}|\mathcal{X})$ because it may be analytically easier. A mixture model is a linear combination of m densities,

$$p(\mathbf{x}|\boldsymbol{\Theta}) = \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}|\boldsymbol{\theta_j}), \quad \alpha_j \geq 0, \quad \sum_{j=1}^{m} \alpha_j = 1 \tag{4.17}$$

where $\boldsymbol{\Theta} = (\alpha_1, \ldots, \alpha_m, \boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_m})$. The log-likelihood function becomes

$$\log L(\boldsymbol{\Theta}|\mathcal{X}) = \log \prod_{i=1}^{n} p(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\Theta}) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta_j}) \right). \tag{4.18}$$

We cannot obtain an analytical solution for $\boldsymbol{\Theta}$ by simply setting the derivatives of $\log L(\boldsymbol{\Theta}|\mathcal{X})$ to zero because of the logarithm of the sum. However, the Expectation-Maximization (EM) algorithm provides a solution to this problem by simplifying the likelihood function by assuming the existence of and values for additional but hidden parameters [26]. We will first present the abstract form of the EM algorithm and then give the solution for a Gaussian mixture.

Expectation-Maximization (EM) Algorithm

In addition to the observed data $\mathcal{X}$, we assume that a complete data set $\mathcal{Z} = (\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})\}$ exists i.i.d. with a joint density function

$$p(\mathbf{z}|\boldsymbol{\Theta}) = p(\mathbf{x}, \mathbf{y}|\boldsymbol{\Theta}) = p(\mathbf{y}|\mathbf{x}, \boldsymbol{\Theta})p(\mathbf{x}|\boldsymbol{\Theta}) \tag{4.19}$$

where $\mathcal{Y}$ is the hidden data.

The EM algorithm first finds the expected value of the complete data log-likelihood $\log L(\mathbf{\Theta}|\mathcal{X},\mathcal{Y}) = \log p(\mathcal{X},\mathcal{Y}|\mathbf{\Theta})$ with respect to the unknown data $\mathcal{Y}$ given the observed data $\mathcal{X}$ and the current parameter estimates $\mathbf{\Theta}$. We define

$$Q(\mathbf{\Theta},\mathbf{\Theta}^{(i-1)}) = E\left[\log p(\mathcal{X},\mathcal{Y}|\mathbf{\Theta})|\mathcal{X},\mathbf{\Theta}^{(i-1)}\right] \tag{4.20}$$

where $\mathbf{\Theta}^{(i-1)}$ is the current set of parameter estimates that are used to evaluate the expectation and $\mathbf{\Theta}$ is the set of parameters that are optimized to increase the objective function Q. The E-step (expectation step) of the algorithm consists of the evaluation of the expectation in Equation (4.20) as

$$E\left[\log p(\mathcal{X},\mathcal{Y}|\mathbf{\Theta})|\mathcal{X},\mathbf{\Theta}^{(i-1)}\right] = \int \log p(\mathcal{X},\mathbf{y}|\mathbf{\Theta})p(\mathbf{y}|\mathcal{X},\mathbf{\Theta}^{(i-1)})d\mathbf{y}. \tag{4.21}$$

The M-step (maximization step) of the algorithm maximizes the expectation in Equation (4.20) to find

$$\mathbf{\Theta}^{(i)} = \arg\max_{\mathbf{\Theta}} Q(\mathbf{\Theta},\mathbf{\Theta}^{(i-1)}). \tag{4.22}$$

These two steps are repeated iteratively. Each iteration is guaranteed to increase the log-likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function [140].

For a mixture model, we assume the hidden data $\mathcal{Y}$ to be $\mathcal{Y} = \{y_i\}_{i=1}^n$ where y_i corresponds to which mixture component generated the data vector $\mathbf{x}_{\boldsymbol{i}}$, i.e. $y_i = k$ if the i'th data vector was generated by the k'th mixture component. Then, the log-likelihood becomes

$$\begin{aligned}\log L(\mathbf{\Theta}|\mathcal{X},\mathcal{Y}) &= \log p(\mathcal{X},\mathcal{Y}|\mathbf{\Theta}) \\ &= \sum_{i=1}^{n} \log(p(\mathbf{x}_{\boldsymbol{i}}|y_i,\boldsymbol{\theta}_{\boldsymbol{i}})p(y_i|\boldsymbol{\theta}_{\boldsymbol{i}})) \\ &= \sum_{i=1}^{n} \log(\alpha_{y_i}p_{y_i}(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{y_i}})).\end{aligned} \tag{4.23}$$

Assume we have the initial parameter estimates $\boldsymbol{\Theta}^{(g)} = (\alpha_1^{(g)}, \ldots, \alpha_m^{(g)}, \theta_1^{(g)}, \ldots, \theta_m^{(g)})$,

$$p(y_i|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}) = \frac{\alpha_{y_i}^{(g)} p_{y_i}(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{y_i}}^{(g)})}{p(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\Theta}^{(g)})} = \frac{\alpha_{y_i}^{(g)} p_{y_i}(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{y_i}}^{(g)})}{\sum_{j=1}^m \alpha_j^{(g)} p_j(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{j}}^{(g)})} \tag{4.24}$$

and

$$p(y|\mathcal{X}, \boldsymbol{\Theta}^{(g)}) = \prod_{i=1}^n p(y_i|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}). \tag{4.25}$$

Then, $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(g)})$ takes the form [26]

$$\begin{aligned} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(g)}) &= \sum_y \log p(\mathcal{X}, y|\boldsymbol{\Theta}) p(y|\mathcal{X}, \boldsymbol{\Theta}^{(g)}) \\ &= \sum_{j=1}^m \sum_{i=1}^n \log(\alpha_j p_j(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{j}})) p(j|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}) \\ &= \sum_{j=1}^m \sum_{i=1}^n \log(\alpha_j) p(j|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}) + \sum_{j=1}^m \sum_{i=1}^n \log(p_j(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{j}})) p(j|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}) \end{aligned} \tag{4.26}$$

where α_j and $p_j(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta}_{\boldsymbol{j}})$ are from the mixture model in Equation (4.17) and $p(j|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)})$ is given in Equation (4.24). We can then maximize the two sets of summations for α_j and $\boldsymbol{\theta}_{\boldsymbol{j}}$ independently because they are not related.

To find the expression for α_j with the constraint that $\sum_{j=1}^m \alpha_j = 1$, we introduce a Lagrange multiplier and take the derivative as

$$\frac{\partial}{\partial \alpha_j} \left[\sum_{j=1}^m \sum_{i=1}^n \log(\alpha_j) p(j|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}) + \lambda \left(\sum_{j=1}^m \alpha_j - 1 \right) \right] = 0 \tag{4.27}$$

which gives

$$\alpha_j = \frac{1}{n} \sum_{i=1}^n p(j|\mathbf{x}_{\boldsymbol{i}}, \boldsymbol{\Theta}^{(g)}). \tag{4.28}$$

EM for Mixture of Gaussians

We can obtain analytical expressions for $\boldsymbol{\theta}_{\boldsymbol{j}}$ for the special case of a Gaussian mixture where $\boldsymbol{\theta}_{\boldsymbol{j}} = (\boldsymbol{\mu}_{\boldsymbol{j}}, \boldsymbol{\Sigma}_{\boldsymbol{j}})$ and

$$p_j(\mathbf{x}|\boldsymbol{\theta}_{\boldsymbol{j}}) = p_j(\mathbf{x}|\boldsymbol{\mu}_{\boldsymbol{j}}, \boldsymbol{\Sigma}_{\boldsymbol{j}}) = \frac{1}{(2\pi)^{q/2}|\boldsymbol{\Sigma}_{\boldsymbol{j}}|^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu}_{\boldsymbol{j}})^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\boldsymbol{j}})/2}, \quad j = 1, \ldots, m. \tag{4.29}$$

The second term in Equation (4.26) becomes

$$\sum_{j=1}^{m}\sum_{i=1}^{n}\log(p_j(\mathbf{x}_{\boldsymbol{i}}|\boldsymbol{\theta_j}))p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)}) =$$
$$\sum_{j=1}^{m}\sum_{i=1}^{n}\left(-\frac{1}{2}\log(|\boldsymbol{\Sigma_j}|)-\frac{1}{2}(\mathbf{x}_{\boldsymbol{i}}-\boldsymbol{\mu_j})^T\boldsymbol{\Sigma_j}^{-1}(\mathbf{x}_{\boldsymbol{i}}-\boldsymbol{\mu_j})\right)p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)}). \tag{4.30}$$

Equating its derivative with respect to $\boldsymbol{\mu_j}$ to zero gives

$$\boldsymbol{\mu_j} = \frac{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})\mathbf{x}_{\boldsymbol{i}}}{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})}. \tag{4.31}$$

The estimate for $\boldsymbol{\Sigma_j}$ can also be obtained as the solution to the maximization of Equation (4.26). We consider five models for the covariance matrix $\boldsymbol{\Sigma_j}$:

1. $\boldsymbol{\Sigma_j} = \sigma^2\mathbf{I}$, all components having the same spherical covariance matrix:

$$\sigma^2 = \frac{1}{nq}\sum_{j=1}^{m}\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})\|\mathbf{x}_{\boldsymbol{i}}-\boldsymbol{\mu_j}\|^2. \tag{4.32}$$

2. $\boldsymbol{\Sigma_j} = \sigma_j^2\mathbf{I}$, each component having an individual spherical covariance matrix:

$$\sigma_j^2 = \frac{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})\|\mathbf{x}_{\boldsymbol{i}}-\boldsymbol{\mu_j}\|^2}{q\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})}. \tag{4.33}$$

3. $\boldsymbol{\Sigma_j} = \mathrm{diag}(\{\sigma_{jk}^2\}_{k=1}^{q})$, each component having an individual diagonal covariance matrix:

$$\sigma_{jk}^2 = \frac{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})(\mathbf{x}_{\boldsymbol{i}k}-\boldsymbol{\mu_j}_k)^2}{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})}. \tag{4.34}$$

4. $\boldsymbol{\Sigma_j} = \boldsymbol{\Sigma}$, each component having the same full covariance matrix:

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}},\boldsymbol{\Theta}^{(g)})(\mathbf{x}_{\boldsymbol{i}}-\boldsymbol{\mu_j})(\mathbf{x}_{\boldsymbol{i}}-\boldsymbol{\mu_j})^T. \tag{4.35}$$

5. $\mathbf{\Sigma}_{\boldsymbol{j}}$, each component having an individual full covariance matrix, the unconstrained case:

$$\mathbf{\Sigma}_{\boldsymbol{j}} = \frac{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}}, \mathbf{\Theta}^{(g)})(\mathbf{x}_{\boldsymbol{i}} - \boldsymbol{\mu}_{\boldsymbol{j}})(\mathbf{x}_{\boldsymbol{i}} - \boldsymbol{\mu}_{\boldsymbol{j}})^T}{\sum_{i=1}^{n} p(j|\mathbf{x}_{\boldsymbol{i}}, \mathbf{\Theta}^{(g)})}. \tag{4.36}$$

Equations (4.28), (4.31) and (4.32) - (4.36) perform both expectation and maximization steps simultaneously. There are also other possibilities for the covariance matrices which can be further parameterized in terms of their eigenvalue decompositions. Celeux and Govaert [40] presented either closed forms or iterative solutions for those cases.

The iterations for the EM algorithm proceed by using the current estimates as the initial estimates for the next iteration. The k-means algorithm [64] can be used to determine the initial configuration. The priors are computed from the proportion of examples belonging to each cluster. The means are the cluster centroids. The covariance matrices are calculated as the sample covariance of the points associated with each cluster. As a stopping criterion for the EM algorithm, we can use a threshold for the number of iterations or we can stop if the change in log-likelihood between two iterations is less than another threshold.

Minimum Description Length

The EM algorithm requires the number of mixture components to be given. Our goal is to find the best Gaussian mixture that describes the training data. One possible choice as the goodness-of-fit is the log-likelihood, as used above. However, when we perform a maximum likelihood estimation of the mixture, there is no bound in the complexity of the model. The likelihood will keep increasing as the number of components increase; the limit being a component for each training vector. On the other hand, smaller models are known to have better generalization ability [30].

The Minimum Description Length (MDL) Principle [163, 164, 114, 59, 60] tries to find a compromise between the model complexity (still having a good data approximation) and the complexity of the data approximation (while using a simple model). Under the MDL principle, the best model is the one that minimizes the sum of the model's complexity $\mathcal{L}(\mathcal{M})$ and the efficiency of the description of the training data with respect to that model $\mathcal{L}(\mathcal{X}|\mathcal{M})$, i.e.

$$\mathcal{L}(\mathcal{X}, \mathcal{M}) = \mathcal{L}(\mathcal{M}) + \mathcal{L}(\mathcal{X}|\mathcal{M}). \tag{4.37}$$

According to Shannon, the shortest code-length to encode data $\mathcal{X}$ with a distribution $p(\mathcal{X}|\mathcal{M})$ under model $\mathcal{M}$ is given by

$$\mathcal{L}(\mathcal{X}|\mathcal{M}) = -\log L(\mathcal{M}|\mathcal{X}) = -\log p(\mathcal{X}|\mathcal{M}) \tag{4.38}$$

where $L(\mathcal{M}|\mathcal{X})$ is the likelihood function for model $\mathcal{M}$ given the sample $\mathcal{X}$. The model complexity is measured as the number of bits required to describe the model parameters. According to Rissanen [163, 164], the code-length to encode $\kappa_{\mathcal{M}}$ real-valued parameters characterizing n data points is

$$\mathcal{L}(\mathcal{M}) = \frac{\kappa_{\mathcal{M}}}{2} \log n \tag{4.39}$$

where $\kappa_{\mathcal{M}}$ is the number of free parameters in model $\mathcal{M}$ and n is the size of the sample used to estimate those parameters.

For a Gaussian mixture model with m components having covariance matrices as given in the previous section, the total number of free parameters for each case are:

1. $\boldsymbol{\Sigma_j} = \sigma^2\mathbf{I}$:

$$\kappa_{\mathcal{M}} = (m-1) + mq + 1 \tag{4.40}$$

2. $\boldsymbol{\Sigma_j} = \sigma_j^2\mathbf{I}$:

$$\kappa_{\mathcal{M}} = (m-1) + mq + m \tag{4.41}$$

3. $\boldsymbol{\Sigma_j}$, diagonal:

$$\kappa_{\mathcal{M}} = (m-1) + mq + mq \tag{4.42}$$

4. $\boldsymbol{\Sigma_j} = \boldsymbol{\Sigma}$:

$$\kappa_{\mathcal{M}} = (m-1) + mq + \frac{q(q+1)}{2} \tag{4.43}$$

5. $\boldsymbol{\Sigma_j}$, full:

$$\kappa_{\mathcal{M}} = (m-1) + mq + m\frac{q(q+1)}{2} \tag{4.44}$$

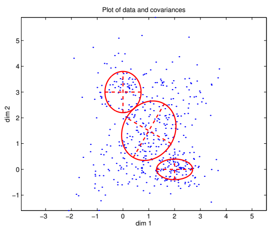
where q is the dimension of the data vectors. The first term describes the mixture weights $\{\alpha_j\}_{j=1}^m$, the second term describes the means $\{\boldsymbol{\mu_j}\}_{j=1}^m$ and the third term describes the covariance matrices $\{\boldsymbol{\Sigma_j}\}_{j=1}^m$. Hence, the best m can be found as

$$m^* = \arg\min_m \left[\frac{1}{2}\kappa_{\mathcal{M}} \log n - \sum_{i=1}^n \log \left(\sum_{j=1}^m \alpha_j p_j(\mathbf{x_i} | \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \right) \right]. \tag{4.45}$$

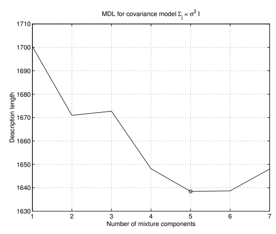
Equation (4.37) can also be used to choose the most appropriate covariance model to a given data. The results for the maximum likelihood estimation of a mixture of Gaussians are summarized in Table 4.1. Example fits for two-dimensional synthetic data are given in Figure 4.1. Models 4 and 5 with full covariance matrices gave the correct number, 3, of the components in the synthetic mixture.

A possible problem for the EM algorithm described above is that there may exist parameter values for which the likelihood goes to infinity [30]. This problem occurs when there are small groups of data points which are close to each other and this may make some of the Gaussian components collapse onto these points. In these cases, the covariance matrix in Equation (4.29) is singular. Several techniques for dealing with the problems of singularities have been proposed [30]. One approach is to constrain the components to have equal covariance matrices, i.e. models 1 and 4 above. Alternatively, variance flooring [142] imposes lower bounds for the variance parameters. If a variance value drops below its corresponding floor value during EM iterations, it is set back to that floor value.

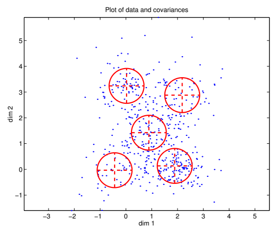
We also consider applying principal components analysis (PCA) to the high-dimensional vectors and fitting mixtures in the low-dimensional projected space. PCA



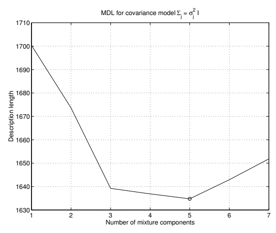
(a) True mixture



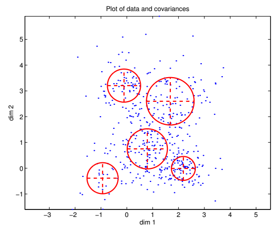
(b) Description length for model 1



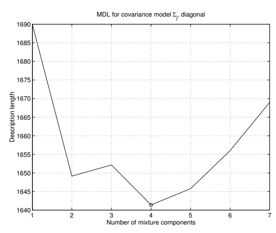
(c) Mixture plot for model 1



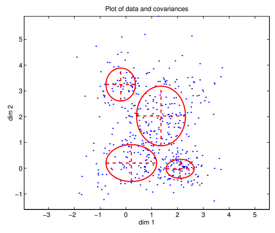
(d) Description length for model 2



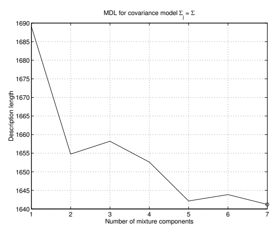
(e) Mixture plot for model 2



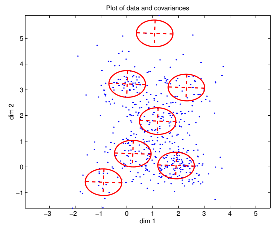
(f) Description length for model 3



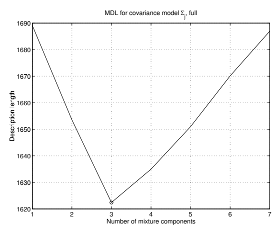
(g) Mixture plot for model 3



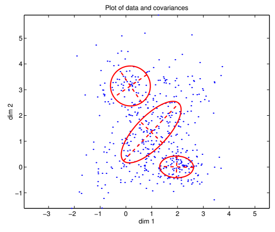
(h) Description length for model 4



(i) Mixture plot for model 4



(j) Description length for model 5



(k) Mixture plot for model 5

Figure 4.1: Example fits for a sample from a mixture of three bivariate Gaussians. Plots show description length vs. number of components and also fitted Gaussians as ellipses at one standard deviations. Models used are 1: $\boldsymbol{\Sigma_j} = \sigma^2\mathbf{I}$, 2: $\boldsymbol{\Sigma_j} = \sigma_j^2\mathbf{I}$, 3: $\boldsymbol{\Sigma_j} = \mathrm{diag}(\{\sigma_{jk}^2\}_{k=1}^q)$, 4: $\boldsymbol{\Sigma_j} = \boldsymbol{\Sigma}$, 5: $\boldsymbol{\Sigma_j}$, full. Using MDL with model 5 could capture the true number of components and model 5 also gave the smallest description length.

Table 4.1: Maximum likelihood estimates and numbers of free parameters for a mixture of m Gaussians $p(\mathbf{x}|\mathbf{\Theta}) = \sum_{j=1}^{m} \alpha_j p_j(\mathbf{x}|\boldsymbol{\mu_j}, \mathbf{\Sigma_j})$ for the sample $\mathbf{x_1}, \ldots, \mathbf{x_n} \in \mathbb{R}^{(q \times 1)}$.

Variable	Estimate (M-step)	# of free parameters
$p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})$	$\frac{\alpha_j^{(g)} p_j(\mathbf{x_i}\|\boldsymbol{\mu_j}^{(g)}, \mathbf{\Sigma_j}^{(g)})}{\sum_{k=1}^{m} \alpha_k^{(g)} p_k(\mathbf{x_i}\|\boldsymbol{\mu_k}^{(g)}, \mathbf{\Sigma_k}^{(g)})}$	
α_j	$\frac{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})}{n}$	$(m-1)$
$\boldsymbol{\mu_j}$	$\frac{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})\mathbf{x_i}}{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})}$	mq
$\mathbf{\Sigma_j} = \sigma^2 \mathbf{I}$	$\sigma^2 = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})\\|\mathbf{x_i} - \boldsymbol{\mu_j}\\|^2}{nq}$	1
$\mathbf{\Sigma_j} = \sigma_j^2 \mathbf{I}$	$\sigma_j^2 = \frac{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})\\|\mathbf{x_i} - \boldsymbol{\mu_j}\\|^2}{q\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})}$	m
$\mathbf{\Sigma_j} = \mathrm{diag}(\{\sigma_{jk}^2\}_{k=1}^{q})$	$\sigma_{jk}^2 = \frac{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})(\mathbf{x_i}_k - \boldsymbol{\mu_j}_k)^2}{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})}$	mq
$\mathbf{\Sigma_j} = \mathbf{\Sigma}$	$\mathbf{\Sigma} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})(\mathbf{x_i} - \boldsymbol{\mu_j})(\mathbf{x_i} - \boldsymbol{\mu_j})^T}{n}$	$\frac{q(q+1)}{2}$
$\mathbf{\Sigma_j}$, full	$\mathbf{\Sigma_j} = \frac{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})(\mathbf{x_i} - \boldsymbol{\mu_j})(\mathbf{x_i} - \boldsymbol{\mu_j})^T}{\sum_{i=1}^{n} p(j\|\mathbf{x_i}, \mathbf{\Theta}^{(g)})}$	$m\frac{q(q+1)}{2}$

tries to find a linear projection that best represents the data in a least-squares sense. The best k-dimensional projection directions are the eigenvectors corresponding to the k largest eigenvalues of the sample covariance matrix. One way of choosing the best k is to do an exhaustive search but this requires a lot of computation for each k value and for each number of components so we use the following approach. Since we are using both a multivariate Gaussian and a mixture of Gaussians models to estimate the distribution of the same training data, we want these two models to have equal number of parameters both for a fair comparison and also to avoid overfitting for the mixture model. The multivariate Gaussian model has $q + q(q+1)/2$ free parameters, the mixture model with different diagonal covariance matrices for each component has $m - 1 + 2mk$ free parameters, the mixture model with the same full covariance matrix for each component has $m - 1 + mk + k(k+1)/2$ free parameters and the mixture model with different full covariance matrices for each component has $m - 1 + mk + mk(k+1)/2$ free parameters. Therefore, k values that make all of these models have the same number of parameters can be found as

$$k = \begin{cases} \min\{\frac{-2m+2+3q+q^2}{4m}, q\} & \text{if } \boldsymbol{\Sigma_j} \text{ is different and diagonal} \\ -\frac{1}{2} - m + \frac{1}{2}\sqrt{9 - 4m + 4m^2 + 12q + 4q^2} & \text{if } \boldsymbol{\Sigma_j} \text{ is same and full} \\ \frac{-3m+\sqrt{m^2+8m+12mq+4mq^2}}{2m} & \text{if } \boldsymbol{\Sigma_j} \text{ is different and full} \end{cases} \tag{4.46}$$

where q is the dimension of the data vectors and m is the number of components.

After finding the number of components for Gaussian mixtures for the relevance and irrelevance classes using MDL separately, the class-conditional distributions become

$$\begin{aligned} p(\mathbf{d}|\mathcal{A}) &= p(\mathbf{d}|\alpha_{\mathcal{A}1}, \dots, \alpha_{\mathcal{A}m_{\mathcal{A}}}, \boldsymbol{\mu_{\mathcal{A}1}}, \dots, \boldsymbol{\mu_{\mathcal{A}m_{\mathcal{A}}}}, \boldsymbol{\Sigma_{\mathcal{A}1}}, \dots, \boldsymbol{\Sigma_{\mathcal{A}m_{\mathcal{A}}}}) \\ &= \sum_{j=1}^{m_{\mathcal{A}}} \frac{\alpha_{\mathcal{A}j}}{(2\pi)^{k/2}|\boldsymbol{\Sigma_{\mathcal{A}j}}|^{1/2}} e^{-(\mathbf{d}-\boldsymbol{\mu_{\mathcal{A}j}})^T \boldsymbol{\Sigma_{\mathcal{A}j}^{-1}}(\mathbf{d}-\boldsymbol{\mu_{\mathcal{A}j}})/2} \end{aligned} \tag{4.47}$$

for the relevance class and

$$\begin{aligned} p(\mathbf{d}|\mathcal{B}) &= p(\mathbf{d}|\alpha_{\mathcal{B}1}, \dots, \alpha_{\mathcal{B}m_{\mathcal{B}}}, \boldsymbol{\mu_{\mathcal{B}1}}, \dots, \boldsymbol{\mu_{\mathcal{B}m_{\mathcal{B}}}}, \boldsymbol{\Sigma_{\mathcal{B}1}}, \dots, \boldsymbol{\Sigma_{\mathcal{B}m_{\mathcal{B}}}}) \\ &= \sum_{j=1}^{m_{\mathcal{B}}} \frac{\alpha_{\mathcal{B}j}}{(2\pi)^{k/2}|\boldsymbol{\Sigma_{\mathcal{B}j}}|^{1/2}} e^{-(\mathbf{d}-\boldsymbol{\mu_{\mathcal{B}j}})^T \boldsymbol{\Sigma_{\mathcal{B}j}^{-1}}(\mathbf{d}-\boldsymbol{\mu_{\mathcal{B}j}})/2} \end{aligned} \tag{4.48}$$

for the irrelevance class where $m_{\mathcal{A}}$ and $m_{\mathcal{B}}$ are the number of components in the mixtures for the relevance and irrelevance classes respectively. The likelihood ratio in Equation (4.1) becomes

$$\Delta(d) = \frac{p(\mathbf{d}|\alpha_{\mathcal{A}1}, \dots, \alpha_{\mathcal{A}m_{\mathcal{A}}}, \boldsymbol{\mu_{\mathcal{A}1}}, \dots, \boldsymbol{\mu_{\mathcal{A}m_{\mathcal{A}}}}, \boldsymbol{\Sigma_{\mathcal{A}1}}, \dots, \boldsymbol{\Sigma_{\mathcal{A}m_{\mathcal{A}}}})}{p(\mathbf{d}|\alpha_{\mathcal{B}1}, \dots, \alpha_{\mathcal{B}m_{\mathcal{B}}}, \boldsymbol{\mu_{\mathcal{B}1}}, \dots, \boldsymbol{\mu_{\mathcal{B}m_{\mathcal{B}}}}, \boldsymbol{\Sigma_{\mathcal{B}1}}, \dots, \boldsymbol{\Sigma_{\mathcal{B}m_{\mathcal{B}}}})} \tag{4.49}$$

and we can then use

$$\Delta'(\mathbf{d}) = \log \sum_{j=1}^{m_{\mathcal{B}}} \alpha_{\mathcal{B}j} p_{\mathcal{B}j}(\mathbf{d}|\boldsymbol{\mu_{\mathcal{B}j}}, \boldsymbol{\Sigma_{\mathcal{B}j}}) - \log \sum_{j=1}^{m_{\mathcal{A}}} \alpha_{\mathcal{A}j} p_{\mathcal{A}j}(\mathbf{d}|\boldsymbol{\mu_{\mathcal{A}j}}, \boldsymbol{\Sigma_{\mathcal{A}j}}) \tag{4.50}$$

to rank the database images. No further simplification is available except using Cholesky decompositions [161] of the covariance matrices.

The previous sections described different density models to estimate the class-conditional probabilities in the likelihood ratio in Equation (4.1) to compute similarities between the query image and the images in the database. The following section describes the commonly used geometric similarity measures.

4.3 Geometric Similarity Measures

The nearest neighbor rule has been the most commonly used similarity measure in image retrieval. It is a geometric similarity measure where each image n in the database is assumed to be represented by its feature vector $\mathbf{y}_{\boldsymbol{n}}$ in the q-dimensional feature space. Given the feature vector $\mathbf{x}$ for the input query, the goal is to find the $\mathbf{y}$'s which are the closest neighbors of $\mathbf{x}$ according to a distance measure ρ. Then, the k-nearest neighbors of $\mathbf{x}$ will be retrieved as the most relevant ones.

The problem of finding the k-nearest neighbors can be formulated as follows. Given the set $\mathcal{Y} = \{\mathbf{y}_{\boldsymbol{n}} | \mathbf{y}_{\boldsymbol{n}} \in \mathbb{R}^{(q\times 1)}, n = 1, \ldots, N\}$ and feature vector $\mathbf{x} \in \mathbb{R}^{(q\times 1)}$, find the set of images $\mathcal{U} \subseteq \{1, \ldots, N\}$ such that $|\mathcal{U}| = k$ and

$$\rho(\mathbf{x}, \mathbf{y}_{\boldsymbol{u}}) \leq \rho(\mathbf{x}, \mathbf{y}_{\boldsymbol{v}}) \ , \quad \forall u \in \mathcal{U}, v \in \{1, \ldots, N\} \backslash \mathcal{U} \tag{4.51}$$

where N being the number of images in the database. Then, images in the set $\mathcal{U}$ are retrieved as the result of the query.

4.3.1 The Minkowsky L_p Metric

The Minkowsky L_p metric [151] is one of the most popular similarity measures used in image retrieval. The original Minkowsky L_p metric is defined for $p \geq 1$. We show below that a modified version of the original metric also becomes a metric for $0 < p < 1$.

Definition 1 *L_p metric for $p \geq 1$:*

$$\rho_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{q} |\mathbf{x}_i - \mathbf{y}_i|^p\right)^{1/p} \tag{4.52}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{(q\times 1)}$ and $\mathbf{x}_i$ and $\mathbf{y}_i$ are the i'th components of the vectors $\mathbf{x}$ and $\mathbf{y}$ respectively.

Lemma 1 $(x+y)^p \leq x^p + y^p$ *for* $0 < p < 1,\ x, y \in \mathbb{R},\ x, y > 0$.

Proof:

Let $f(x,y) = x^p + y^p - (x+y)^p$.

$$\frac{\partial f}{\partial x} = px^{p-1} - p(x+y)^{p-1} = p[x^{p-1} - (x+y)^{p-1}] > 0$$

$$\frac{\partial f}{\partial y} = py^{p-1} - p(x+y)^{p-1} = p[y^{p-1} - (x+y)^{p-1}] > 0$$

$f(x,0) = f(0,y) = 0$ (Boundary conditions).

Therefore, $f(x,y) \geq 0$ for $x > 0$ and $y > 0$.

Definition 2 L_p *metric for* $0 < p < 1$*:*

$$\rho_p(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{q} |\mathbf{x}_i - \mathbf{y}_i|^p. \tag{4.53}$$

i) Positivity: $\rho_p(\mathbf{x},\mathbf{y}) \geq 0$ and if $\mathbf{x} = \mathbf{y}$, then $\rho_p(\mathbf{x},\mathbf{y}) = 0$.

ii) Strictly positivity: If $\rho_p(\mathbf{x},\mathbf{y}) = 0$, $|\mathbf{x}_i - \mathbf{y}_i| = 0\ \forall i$, i.e. $\mathbf{x}_i = \mathbf{y}_i\ \forall i$.

iii) Symmetry: $\rho_p(\mathbf{x},\mathbf{y}) = \rho_p(\mathbf{y},\mathbf{x})$.

iv) Triangle inequality:

$$\begin{aligned}
\rho_p(\mathbf{x},\mathbf{y}) &= \sum_{i=1}^{q} |\mathbf{x}_i - \mathbf{y}_i|^p \\
&= \sum_{i=1}^{q} |\mathbf{x}_i - \mathbf{z}_i + \mathbf{z}_i - \mathbf{y}_i|^p \\
&\leq \sum_{i=1}^{q} (|\mathbf{x}_i - \mathbf{z}_i| + |\mathbf{z}_i - \mathbf{y}_i|)^p \\
&\leq \sum_{i=1}^{q} (|\mathbf{x}_i - \mathbf{z}_i|^p + |\mathbf{z}_i - \mathbf{y}_i|^p) \quad \text{(Using Lemma 1)} \\
&= \sum_{i=1}^{q} |\mathbf{x}_i - \mathbf{z}_i|^p + \sum_{i=1}^{q} |\mathbf{z}_i - \mathbf{y}_i|^p \\
&= \rho_p(\mathbf{x},\mathbf{z}) + \rho_p(\mathbf{z},\mathbf{y})
\end{aligned}$$

We use the form in Equation (4.53) to rank the database images since the power $1/p$ in Equation (4.52) does not affect the ranks. We will describe how we choose which p to use in the following section.

4.3.2 Choosing p

Commonly used forms of the L_p metric are the city-block (L_1) distance and the Euclidean (L_2) distance. Note that the L_1 metric implicitly assumes that the components of the feature difference vector are i.i.d. Double Exponentials and L_2 metric implicitly assumes that they are i.i.d. Gaussians. Sclaroff *et al.* [178] used L_p metrics with a selection criterion based on the relevance feedback from the user. The best L_p metric for each query was chosen as the one that minimized the mean distance between the relevant images. However, no study of the performance of this selection criterion was presented.

Within our classification framework, we use a linear classifier to choose the best p value for the L_p metric. Given training sets of feature vector pairs $(\mathbf{x},\mathbf{y})$ for the relevance and irrelevance classes, first, the distances ρ_p are computed as in Equation

(4.53). Then, from the histograms of ρ_p for the relevance class $\mathcal{A}$ and the irrelevance class $\mathcal{B}$, a threshold θ is selected for classification. This corresponds to a likelihood ratio test where the class-conditional densities are estimated by the histograms.

After the threshold is selected, the classification rule becomes

$$\text{assign } (\mathbf{x},\mathbf{y}) \text{ to } \begin{cases} \text{class } \mathcal{A} & \text{if } \rho_p(\mathbf{x},\mathbf{y}) < \theta \\ \text{class } \mathcal{B} & \text{if } \rho_p(\mathbf{x},\mathbf{y}) \geq \theta. \end{cases} \tag{4.54}$$

We use a minimum error decision rule with equal priors, i.e. the threshold is the intersecting point of the two histograms. The best p value is then chosen as the one that minimizes the classification error which is defined as 0.5 misdetection + 0.5 false alarm.

4.4 Feature Space vs. Probability Space

Sample size is very important in building classifiers. Duin [65] discussed the effects of the curse of dimensionality and sample size on classification error. Although it was traditionally thought that it is necessary to fill the feature space with more objects than its dimensionality to obtain a classifier that can generalize well, he argued that it is possible to build reliable classifiers in very small sample size problems. He used a kernel mapping to map the high-dimensional feature space to a low-dimensional kernel space. This mapping allowed building simple linear classifiers in the kernel space that corresponded to complex non-linear classifiers in the feature space when non-linear kernels were used. An important observation was that when the number of features was less than the sample size, there was statistical adaptation where the set of parameters could be estimated accurately, then, the generalization error became maximum when the sample size became equal to the number of features where the structure became too complex (too many parameters) and statistical adaptation was poor (too poor noise averaging), but after that, better results could be obtained because of structural adaptation. In [158], Pekalska and Duin used prototype objects for dimensionality reduction where each object was represented by its distances to the prototype objects and constructed decision rules based on the distances of training objects to the prototype objects. They concluded that it was possible to build reliable classifiers in low-dimensional spaces even though the feature space was sparse.

As discussed in Section 1.4, the proposed probabilistic setting can also be interpreted as a mapping from the high-dimensional feature space to the two-dimensional probability space. Therefore, classification can be done either in the feature space using the feature difference vector $\mathbf{d}$, or in the probability space using the class-conditional probabilities $p(\mathbf{d}|\mathcal{A})$ and $p(\mathbf{d}|\mathcal{B})$ as new features estimated using the models that were described in the previous sections.

Levels of classification in the probability space and the feature space are summarized in Figures 4.2 and 4.3 respectively. The pattern recognition literature [78, 64, 106] provides many choices for a classifier. We use nine classifiers [66, 67]:

1. *Linear classifier* assuming Gaussian densities with equal covariance matrices for both classes,

2. *Quadratic classifier* also assuming Gaussian densities but with different covariance matrices for each class,

3. *Fisher's linear classifier* where the posterior probabilities are computed by normalizing the distances to the discriminant using a sigmoid,

4. *Logistic linear classifier* which is computed by maximizing the likelihood criterion using the sigmoid function,

5. *Scaled nearest mean classifier* assuming spherical covariance matrices for each class and assigning each object to the class of the closest mean where the posterior probabilities are computed by normalizing the distances to the means using a sigmoid,

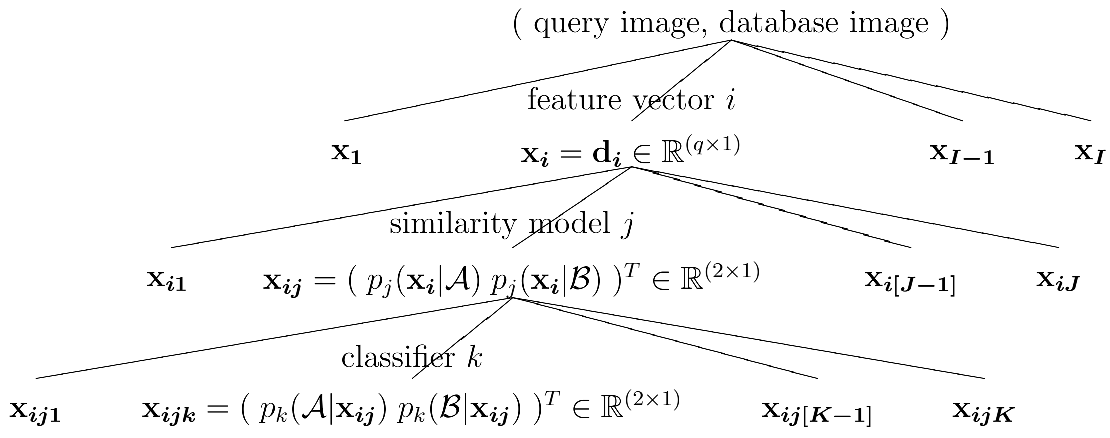

(query image, database image)
feature vector i
$\mathbf{x_1}$
$\mathbf{x_i} = \mathbf{d_i} \in \mathbb{R}^{(q\times 1)}$
$\mathbf{x_{I-1}}$
$\mathbf{x_I}$
similarity model j
$\mathbf{x_{i1}}$
$\mathbf{x_{ij}} = (\ p_j(\mathbf{x_i}|\mathcal{A})\ p_j(\mathbf{x_i}|\mathcal{B})\ )^T \in \mathbb{R}^{(2\times 1)}$
$\mathbf{x_{i[J-1]}}$
$\mathbf{x_{iJ}}$
classifier k
$\mathbf{x_{ij1}}$
$\mathbf{x_{ijk}} = (\ p_k(\mathcal{A}|\mathbf{x_{ij}})\ p_k(\mathcal{B}|\mathbf{x_{ij}})\ )^T \in \mathbb{R}^{(2\times 1)}$
$\mathbf{x_{ij[K-1]}}$
$\mathbf{x_{ijK}}$

Figure 4.2: Levels of classification in the probability space for a system with I feature vectors, J similarity models and K classifiers. $\mathbf{x}$ values represent the measurements in each level, $\mathbf{d}$ values represent the feature difference vectors and p values are probabilities. In the feature vector level, there are I measurements $\mathbf{x_i}, 1 \leq i \leq I$. Then, J similarity models map each $\mathbf{x_i}$ to the measurements $\mathbf{x_{ij}}, 1 \leq j \leq J$, which contain the class-conditional probabilities. Finally, for each $\mathbf{x_{ij}}$, K classifiers output the measurements $\mathbf{x_{ijk}}, 1 \leq k \leq K$, which contain the posterior probabilities.

6. *The nearest neighbor classifier* where the posterior probabilities are proportional to the distances between the object and the closest objects from each class,

7. *Parzen classifier* with a Gaussian kernel for each object,

8. *Binary decision tree classifier* where the posterior probabilities are estimated using the class frequencies in the end nodes,

9. *Feed-forward neural network classifier* with three hidden units that were trained using back-propagation where the posterior probabilities are computed from the normalized output values.

In a system with I feature representations (feature vectors), J similarity models and K classifiers, there are $I \times J \times K$ possible configurations for classification in the probability space and $I \times K$ possible configurations for classification in the feature

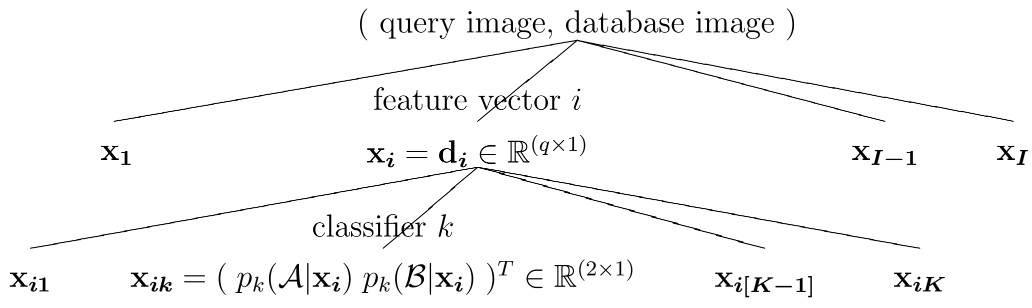
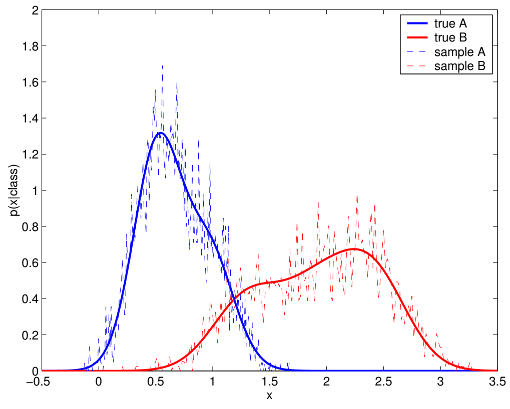

(query image, database image)
feature vector i
$\mathbf{x_1}$
$\mathbf{x_i} = \mathbf{d_i} \in \mathbb{R}^{(q\times 1)}$
$\mathbf{x_{I-1}}$
$\mathbf{x_I}$
classifier k
$\mathbf{x_{i1}}$
$\mathbf{x_{ik}} = (\ p_k(\mathcal{A}|\mathbf{x_i})\ p_k(\mathcal{B}|\mathbf{x_i})\ )^T \in \mathbb{R}^{(2\times 1)}$
$\mathbf{x_{i[K-1]}}$
$\mathbf{x_{iK}}$

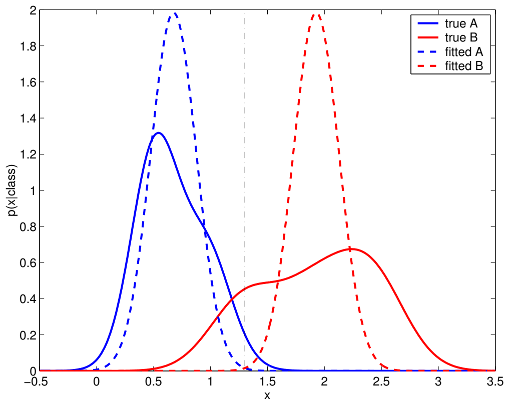
Figure 4.3: Levels of classification in the feature space for a system with I feature vectors and K classifiers. $\mathbf{x}$ values represent the measurements in each level, $\mathbf{d}$ values represent the feature difference vectors and p values are probabilities. For each of the I measurements $\mathbf{x_i}, 1 \leq i \leq I$, in the feature vector level, K classifiers output the measurements $\mathbf{x_{ik}}, 1 \leq k \leq K$, which contain the posterior probabilities.

space. Similarity can then be computed as likelihood in the probabilistic setting as in Section 4.2 instead of computing distances in the geometric setting as in Section 4.3.

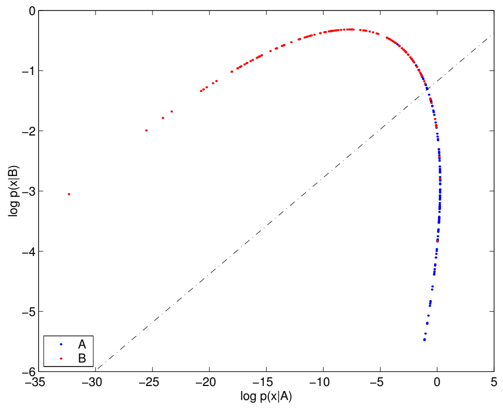
The class-conditional probabilities computed using parametric density models in the high-dimensional feature space are only estimates of the true probabilities (because of imperfect density modeling, quantization, dimensionality, etc.). The classifiers trained in the two-dimensional space of class-conditional probabilities impose a second level modeling of probability, i.e. "probability of probability", to compensate for errors in modeling probabilities in the feature space. This two-level modeling is illustrated with one- and two-dimensional synthetic data generated using mixtures of three univariate and bivariate Gaussians in Figures 4.4 and 4.5 respectively. Classification error is first computed in the original signal space using a linear Gaussian classifier. Then, this error is compared to the error computed using a new classifier in the space formed by the class-conditional probabilities. Even though the final probability of error was still higher than the true Bayes error, it was always smaller in the probability space than in the original signal space in experiments done using different sample sizes.



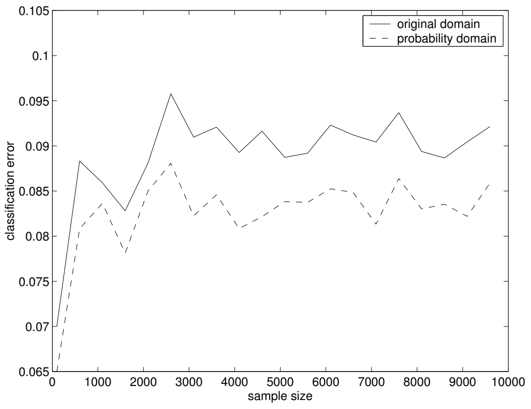
(a) True distributions (solid) and 2500 samples (dashed) for each of relevance (blue) and irrelevance (red) classes (Bayes error, $P_e = 0.0828$)



(b) True (solid) and estimated (dashed) distributions for relevance (blue) and irrelevance (red) classes for a linear Gaussian classifier (dash-dot, black) ($P_e = 0.0919$)

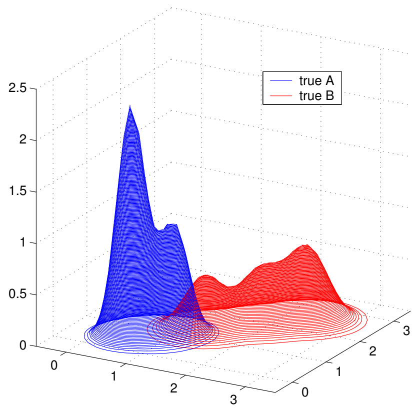


(c) A linear Gaussian classifier (dash-dot, black) trained in the log-probability space for relevance (blue) and irrelevance (red) classes ($P_e = 0.0879$)

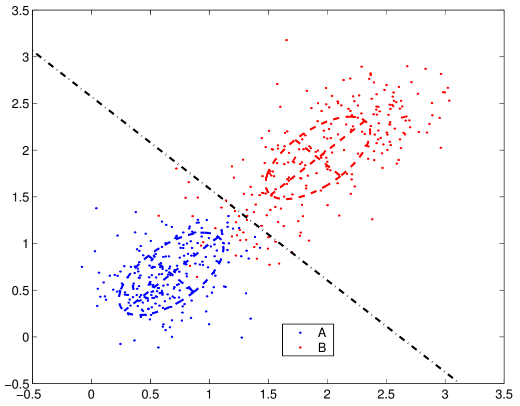


(d) Classification error (y-axis) vs. sample size (x-axis) for linear Gaussian classifiers trained in signal (solid) and log-probability (dashed) spaces

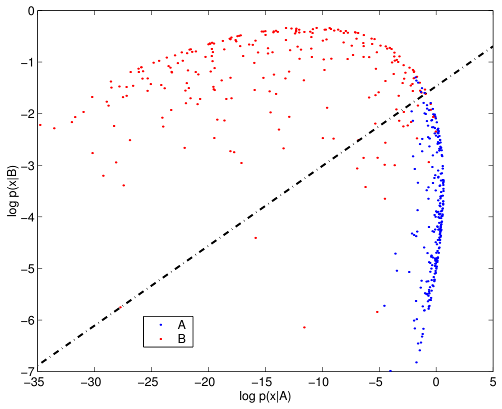
Figure 4.4: Classification in signal and probability spaces for synthetic data generated using mixtures of three univariate Gaussians for both the relevance and irrelevance classes. A linear Gaussian classifier is used for classification in both spaces. Classification errors (P_e) for true and estimated distributions are given in parentheses. Two-level probability modeling always gave a smaller error.



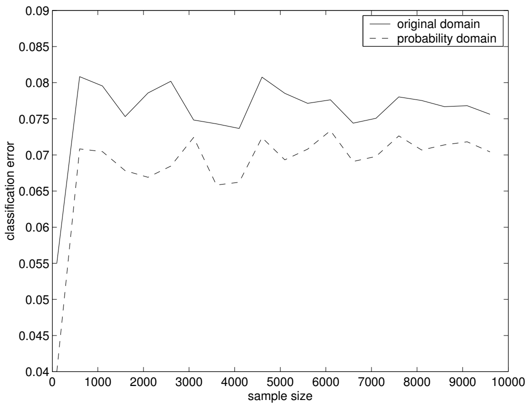
(a) True distributions for each of relevance (blue) and irrelevance (red) classes



(b) 2500 samples and fitted Gaussians as ellipses at one standard deviations for relevance (blue) and irrelevance (red) classes for a linear Gaussian classifier (dash-dot, black) ($P_e = 0.0770$)



(c) A linear Gaussian classifier (dash-dot, black) trained in the log-probability space for relevance (blue) and irrelevance (red) classes ($P_e = 0.0694$)



(d) Classification error (y-axis) vs. sample size (x-axis) for linear Gaussian classifiers trained in signal (solid) and log-probability (dashed) spaces

Figure 4.5: Classification in signal and probability spaces for synthetic data generated using mixtures of three bivariate Gaussians for both the relevance and irrelevance classes. A linear Gaussian classifier is used for classification in both spaces. Classification errors (P_e) for estimated distributions are given in parentheses. Two-level probability modeling always gave a smaller error.

Similar approaches for multi-level modeling in the literature usually require complex classifiers because they are trained directly in the feature space. For example, Chou and Shapiro [45] developed a hierarchical multiple classifier algorithm where component classifiers were trained on clusters of training data and then super-classifiers were trained on the outputs of the component classifiers. However, even though they could reduce the amount of training by clustering the input features, complex classifiers like neural networks and decision trees had to be used for both component and super-classifiers because of the high-dimensional input data, as opposed to our case where simple models (e.g. multivariate Gaussian, linear Gaussian classifiers) can be trained by doing both dimensionality reduction and probability modeling simultaneously. Huang and Suen [97] also used a two-level combination of neural network classifiers for handwritten numerals. Bilmes and Kirchhoff [27] developed directed graphical models to combine the outputs of two classifiers. They argued that classifier combination is a statistical process where directed graphical models are a rich language to describe its underlying assumptions. They used models with five variables; a variable for the target class, two hidden variables for the outputs of two classifiers and two variables for the inputs of those classifiers. They used discrete probability tables to estimate the conditional probabilities for the relationships between the variables. In experiments where the individual classifiers were neural networks, they found that some of the combination rules with more relaxed independence assumptions gave better performances than the sum and product rules. Duin and Tax [67] also used a two-level classifier combination. First, classifiers like Gaussian, nearest neighbor, Parzen, neural network and decision tree classifiers were trained in the original feature space. Then, Gaussian, nearest mean and nearest neighbor classifiers were trained using the outputs of the previous classifiers. They compared the performances of these second level of classifiers to the performances of fixed combination rules (e.g. product, sum), and observed that combination using the second level of classifiers worked well for combining classifiers which were not primarily designed to give posterior probabilities (e.g. neural networks). Other approaches for classifier combination include training classifiers in the original feature space and then using fixed combination rules (e.g. product, sum, majority voting) to combine the outputs of these classifiers [126, 119, 127, 58, 120, 118] but those approaches correspond to only one level of modeling in the framework of Figures 4.2 and 4.3. We will investigate classifier combination in more detail in Chapter 7.

The improvements obtained by the two-level modeling proposed in this section are more significant than the cases for the synthetic data in Figures 4.4 and 4.5 when the initial signal (feature) space has a much higher dimension. The experiments are discussed in the following section.

4.5 Experiments

4.5.1 Classification Performance

Figures 4.6 - 4.8 show plots of the class-conditional log-probabilities for the relevance and irrelevance classes under different models for different databases. Each plot corresponds to $\mathbf{x}_{ij}, 1 \leq i \leq 5, 1 \leq j \leq 3$ in Figure 4.2. LAR, COOC, GABOR, MOMENTS, TAMURA and COLHIST are the feature representations described in Section 3.2, and MVG, FIT and GMIX are the probability models described in this chapter. For the mixture of Gaussians (GMIX) model, we constrained each component to have either different diagonal covariance matrices, or the same full covariance matrix, or different full covariance matrices. The number of components was chosen using the Minimum Description Length (MDL) criterion. The plots in Figure 4.6 are given for the training set of the ISL Database. From a total of 600 images in 7 groups in the groundtruth, 30 randomly selected images from each group were used for training and 55 randomly selected images from each group were used for testing. The plots in Figure 4.7 are given for the training set of the VisTex Database. From a total of 736 images in 46 groups in the groundtruth, 6 randomly selected images from each group were used for training and 10 randomly selected images from each group were used for testing. The plots in Figure 4.8 are given for the training set of the COREL Database. From a total of 1,575 images in 18 groups in the groundtruth, 30 randomly selected images from each group were used for training and 39 randomly selected images from each group were used for testing. The protocol for training and testing image pair generation was described in Section 1.5.2. All results in this section are given for the normalization methods that performed the best in terms of class separability for each feature representation as discussed in Section 3.4. Other normalization methods gave similar results but are omitted due to space considerations.

Nine classifiers were trained in the two-dimensional probability space and the decision boundaries for some of them are also shown in the figures. We can see that some of the feature vector and similarity model combinations result in a better separation between the training relevance class (red) and irrelevance class (blue) points. To compare the classification performance in the probability space and the feature space, we also trained classifiers in the feature space and computed classification errors for testing sets. Classification performances in terms of classification error for training and testing datasets using different classifiers and different feature vectors in the probability and feature spaces are given in Tables 4.2, 4.3 and 4.4 for the ISL Database, VisTex Database and COREL Database respectively. Each number in the first four main columns corresponds to $\mathbf{x}_{\boldsymbol{ijk}}, 1 \leq i \leq 5, 1 \leq j \leq 3, 1 \leq k \leq 9$ in Figure 4.2. Each number in the last main column corresponds to $\mathbf{x}_{\boldsymbol{ik}}, 1 \leq i \leq 5, 1 \leq k \leq 9$ in Figure 4.3.

Simple classifiers like the Logistic linear or Gaussian quadratic classifiers performed often as well as and sometimes better than the complex Parzen, decision tree and neural network classifiers in the probability space. Linear classifiers in the probability space also performed much better than the non-linear classifiers in the feature space although the non-linear classifiers performed better than linear ones in the feature space. This is a very useful result because it allows us to do effec-

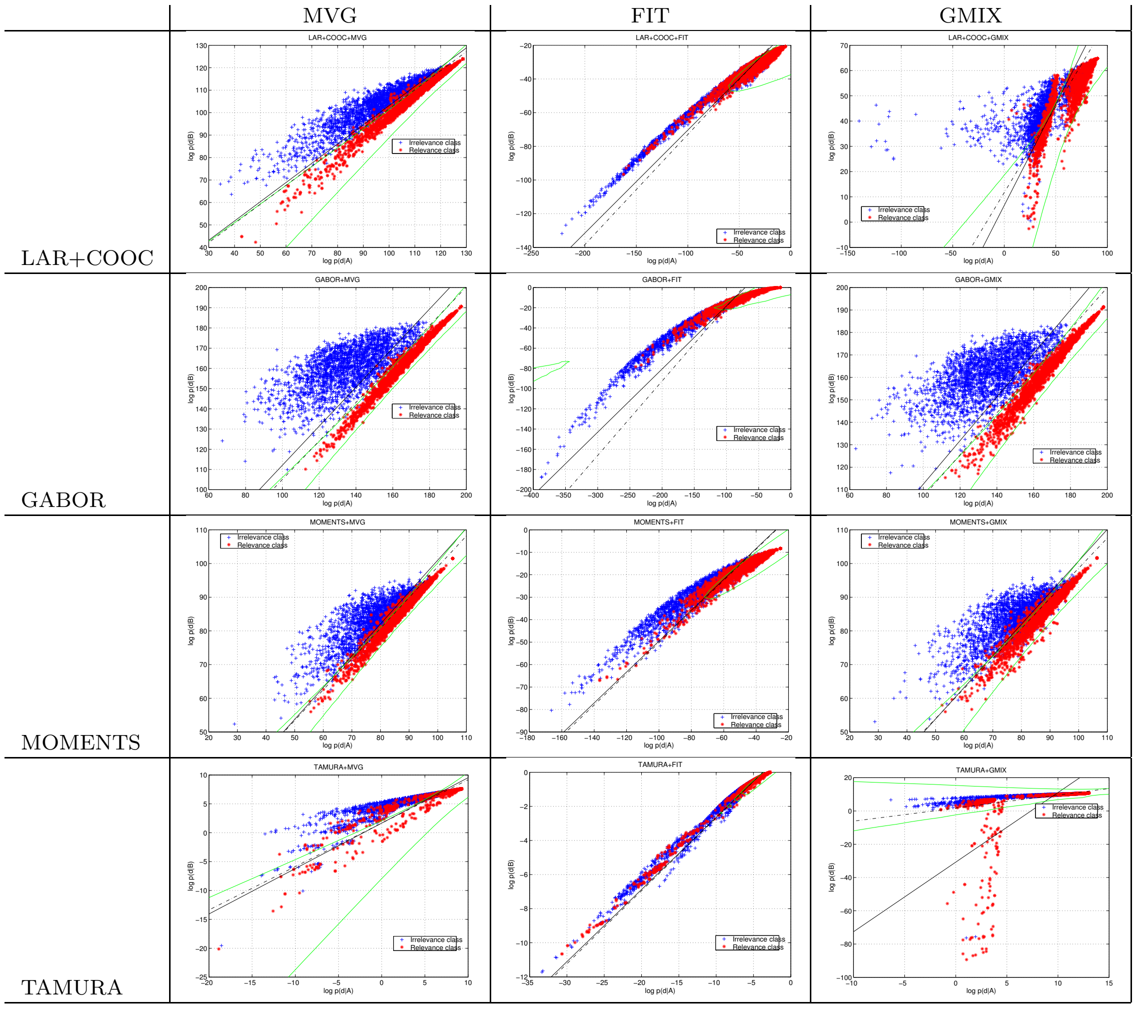


Figure 4.6: Class-conditional log-probabilities for different feature vectors and similarity models for the ISL Database. X-axis shows log-probabilities for the relevance class and y-axis shows log-probabilities for the irrelevance class. Red dots represent training data for the relevance class and blue dots represent training data for the irrelevance class. Solid black line is the Gaussian linear classifier, solid green line is the Gaussian quadratic classifier, and dash-dot black line is the logistic linear classifier.

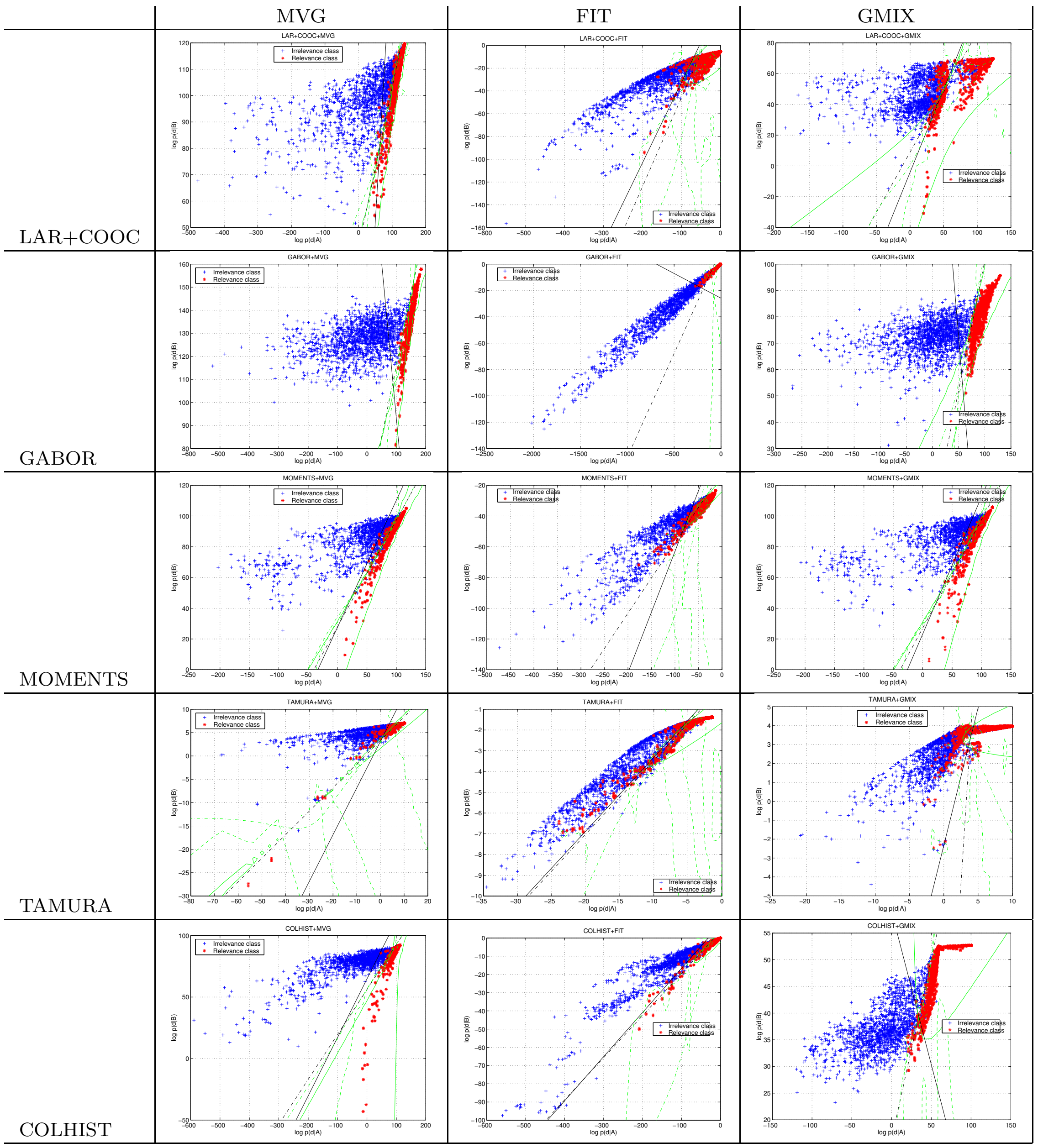


Figure 4.7: Class-conditional log-probabilities for different feature vectors and similarity models for the VisTex Database. X-axis shows log-probabilities for the relevance class and y-axis shows log-probabilities for the irrelevance class. Red dots represent training data for the relevance class and blue dots represent training data for the irrelevance class. Solid black line is the Gaussian linear classifier, solid green line is the Gaussian quadratic classifier, dash-dot black line is the logistic linear classifier, and dash-dot green line is the neural network classifier.

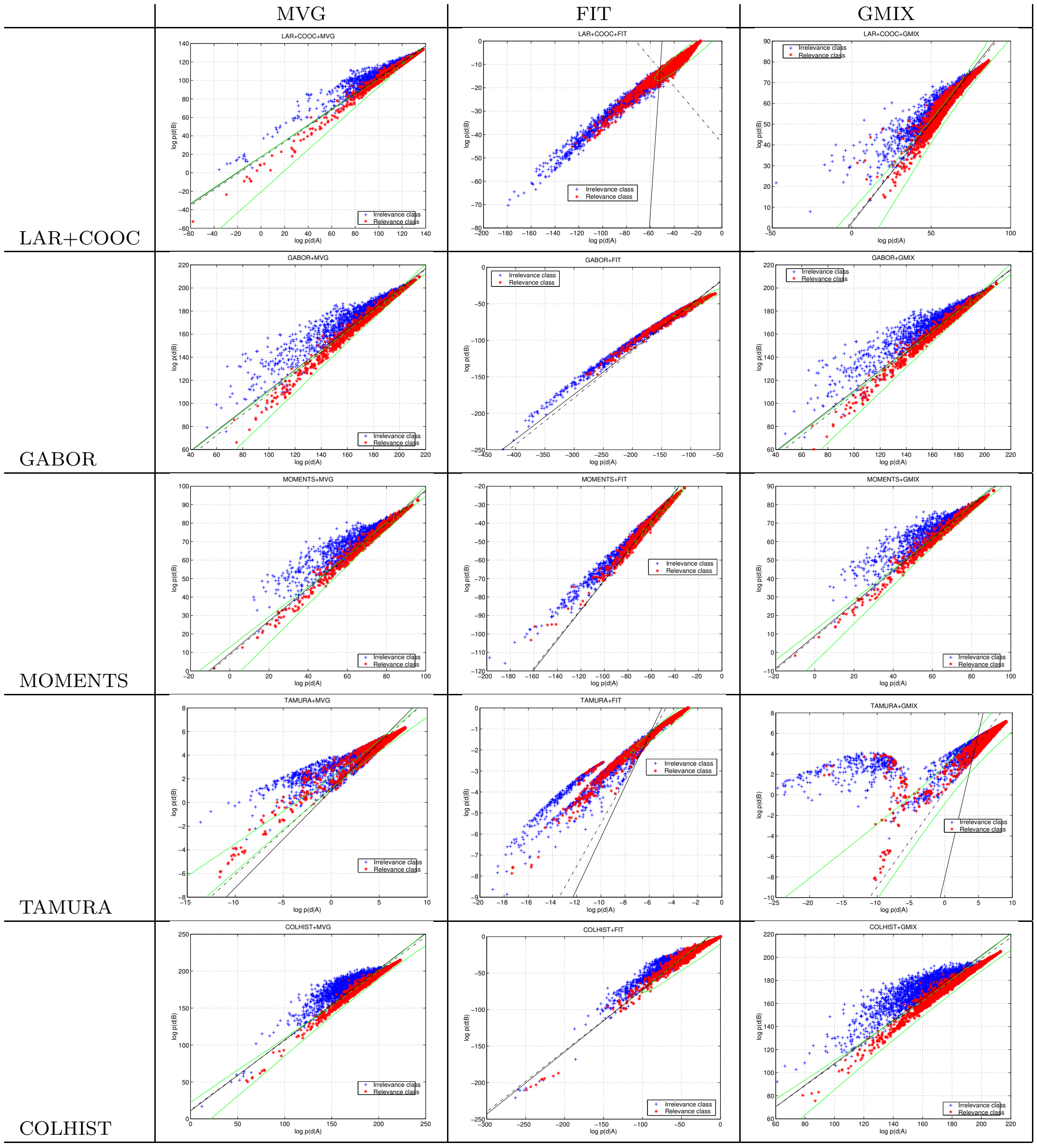


Figure 4.8: Class-conditional log-probabilities for different feature vectors and similarity models for the COREL Database. X-axis shows log-probabilities for the relevance class and y-axis shows log-probabilities for the irrelevance class. Red dots represent training data for the relevance class and blue dots represent training data for the irrelevance class. Solid black line is the Gaussian linear classifier, solid green line is the Gaussian quadratic classifier, and dash-dot black line is the logistic linear classifier.

Table 4.2: Classification performance in terms of classification error for training and testing datasets using different classifiers and different feature vectors in the probability and feature spaces for the ISL Database. The best performing classifiers (that gave the smallest classification errors) for particular feature vectors and similarity models are marked by boxes.

		Classification error in probability space						Classification error in feature space	
		MVG		FIT		GMIX			
Feature	Classifier	Training	Testing	Training	Testing	Training	Testing	Training	Testing
LAR+COOC	Gaussian linear classifier	0.105397	0.211759	0.314048	0.313365	0.238333	$\boxed{0.324557}$	0.479444	0.495207
	Gaussian quadratic classifier	0.080079	0.177473	0.340714	0.328194	$\boxed{0.232460}$	0.349233	$\boxed{0.103889}$	$\boxed{0.203849}$
	Fisher's linear classifier	0.105397	0.211759	0.314048	0.313365	0.238333	$\boxed{0.324557}$	0.479444	0.495207
	Logistic linear classifier	$\boxed{0.067063}$	$\boxed{0.169327}$	$\boxed{0.313968}$	$\boxed{0.311452}$	0.233889	0.325950	0.479444	0.495230
	Scaled nearest mean classifier	0.348810	0.376623	0.337698	0.316222	0.351825	0.365407	0.484762	0.495514
GABOR	Gaussian linear classifier	0.078810	0.117898	0.245238	0.248076	0.077222	0.123424	0.476111	0.493955
	Gaussian quadratic classifier	0.017937	0.096269	0.293889	0.288784	0.021587	0.098323	$\boxed{0.023889}$	$\boxed{0.151405}$
	Fisher's linear classifier	0.078810	0.117898	0.245238	0.248076	0.077222	0.123424	0.476111	0.493955
	Logistic linear classifier	$\boxed{0.016111}$	$\boxed{0.090697}$	$\boxed{0.244762}$	$\boxed{0.246659}$	$\boxed{0.019444}$	$\boxed{0.092326}$	0.476190	0.493979
	Scaled nearest mean classifier	0.199921	0.302621	0.262143	0.260142	0.189365	0.301015	0.484841	0.490697
MOMENTS	Gaussian linear classifier	0.161349	0.227013	0.251587	0.256246	0.161905	0.236198	0.479841	0.496293
	Gaussian quadratic classifier	0.129286	0.218819	0.253889	0.263943	0.149683	0.233412	$\boxed{0.128413}$	$\boxed{0.238229}$
	Fisher's linear classifier	0.161349	0.227013	0.251587	0.256246	0.161905	0.236198	0.479841	0.496293
	Logistic linear classifier	$\boxed{0.120635}$	$\boxed{0.217096}$	$\boxed{0.249841}$	$\boxed{0.255325}$	$\boxed{0.131905}$	$\boxed{0.232940}$	0.479841	0.496316
	Scaled nearest mean classifier	0.326587	0.364604	0.286190	0.299032	0.316667	0.361511	0.487222	0.492963
TAMURA	Gaussian linear classifier	0.268730	0.288028	0.283571	0.295986	0.290635	0.310484	0.483968	0.493341
	Gaussian quadratic classifier	0.288810	0.306824	$\boxed{0.274841}$	$\boxed{0.294050}$	0.478730	0.480567	$\boxed{0.257778}$	$\boxed{0.275466}$
	Fisher's linear classifier	0.268730	0.288028	0.283571	0.295986	0.290635	0.310484	0.483968	0.493341
	Logistic linear classifier	$\boxed{0.255317}$	$\boxed{0.272113}$	0.281905	0.294900	$\boxed{0.256587}$	$\boxed{0.274711}$	0.483968	0.493341
	Scaled nearest mean classifier	0.324444	0.337851	0.313413	0.336151	0.315714	0.332326	0.484921	0.493412

Table 4.3: Classification performance in terms of classification error for training and testing datasets using different classifiers and different feature vectors in the probability and feature spaces for the VisTex Database. The best performing classifiers for particular feature vectors and similarity models are marked by boxes.

		Classification error in probability space						Classification error in feature space	
		MVG		FIT		GMIX			
Feature	Classifier	Training	Testing	Training	Testing	Training	Testing	Training	Testing
LAR+COOC	Gaussian linear classifier	0.220109	0.247826	0.196860	0.204022	0.146135	0.200109	0.431461	0.463370
	Gaussian quadratic classifier	0.123792	0.157609	0.202899	0.206739	0.162440	0.217391	0.142512	0.210109
	Fisher's linear classifier	0.220109	0.247826	0.196860	0.204022	0.146135	0.200109	0.431461	0.463370
	Logistic linear classifier	0.112319	$\boxed{0.151848}$	0.174819	$\boxed{0.186522}$	0.150664	$\boxed{0.186848}$	0.431159	0.463696
	Scaled nearest mean classifier	0.269626	0.293913	0.206824	0.224783	0.256341	0.296522	0.424215	0.452609
	Nearest neighbor classifier	$\boxed{0.000000}$	0.224457	$\boxed{0.000000}$	0.264457	$\boxed{0.000000}$	0.246087	$\boxed{0.000000}$	0.200000
	Parzen classifier	0.100543	0.170000	0.150664	0.191630	0.091787	0.194348	0.067029	0.197500
	Binary decision tree classifier	$\boxed{0.000000}$	0.228587	$\boxed{0.000000}$	0.274783	$\boxed{0.000000}$	0.243043	–	–
	Neural network classifier	0.119263	0.156304	0.174215	0.189674	0.146135	0.195870	0.106884	$\boxed{0.165652}$
GABOR	Gaussian linear classifier	0.081522	0.094457	0.153986	0.170543	0.069746	0.091196	0.429650	0.467935
	Gaussian quadratic classifier	0.010266	0.097609	0.072766	0.080761	0.016606	$\boxed{0.066304}$	0.016908	0.148043
	Fisher's linear classifier	0.081522	0.094457	0.153986	0.170543	0.069746	0.091196	0.429650	0.467935
	Logistic linear classifier	0.007850	$\boxed{0.060217}$	0.068539	0.073261	0.008756	0.091630	0.429348	0.468152
	Scaled nearest mean classifier	0.084843	0.123261	0.153986	0.170435	0.080616	0.123370	0.450785	0.467717
	Nearest neighbor classifier	$\boxed{0.000000}$	0.092500	$\boxed{0.000000}$	0.112717	$\boxed{0.000000}$	0.122826	$\boxed{0.000000}$	0.063587
	Parzen classifier	0.000604	0.087826	0.059783	0.077065	0.003019	0.106848	0.004227	$\boxed{0.062935}$
	Binary decision tree classifier	$\boxed{0.000000}$	0.091413	$\boxed{0.000000}$	0.114022	$\boxed{0.000000}$	0.128152	$\boxed{0.000000}$	0.090978
	Neural network classifier	0.006039	0.069130	0.064614	$\boxed{0.070978}$	0.009964	0.069348	0.003321	0.116957
MOMENTS	Gaussian linear classifier	0.226449	0.241848	0.211051	0.208043	0.224940	0.244457	0.442935	0.474130
	Gaussian quadratic classifier	0.130133	0.159022	0.198068	0.196304	0.139493	0.171196	0.127415	0.193587
	Fisher's linear classifier	0.226449	0.241848	0.211051	0.208043	0.224940	0.244457	0.442935	0.474130
	Logistic linear classifier	0.102657	$\boxed{0.152500}$	0.163949	$\boxed{0.162609}$	0.099336	$\boxed{0.164022}$	0.443539	0.473913
	Scaled nearest mean classifier	0.293780	0.295326	0.231582	0.235109	0.291667	0.294457	0.455012	0.468478
	Nearest neighbor classifier	$\boxed{0.000000}$	0.208478	$\boxed{0.000000}$	0.233696	$\boxed{0.000000}$	0.214891	$\boxed{0.000000}$	0.167283
	Parzen classifier	0.082428	0.160326	0.130435	0.175652	0.080616	0.175326	0.021739	$\boxed{0.152283}$
	Binary decision tree classifier	$\boxed{0.000000}$	0.210109	$\boxed{0.000000}$	0.227500	$\boxed{0.000000}$	0.224239	$\boxed{0.000000}$	0.186957
	Neural network classifier	0.094203	0.155000	0.171800	0.179457	0.100543	0.164674	0.088768	0.154348
TAMURA	Gaussian linear classifier	0.213164	0.214674	0.182971	0.178043	0.177536	0.183587	0.432971	0.463804
	Gaussian quadratic classifier	0.190821	0.194565	0.176329	0.173696	0.175423	0.167935	0.175725	0.171522
	Fisher's linear classifier	0.213164	0.214674	0.182971	0.178043	0.177536	0.183587	0.432971	0.463804
	Logistic linear classifier	0.163345	0.164891	0.172705	0.174783	0.176932	0.178370	0.433273	0.463804
	Scaled nearest mean classifier	0.215580	0.213478	0.221618	0.217391	0.172403	$\boxed{0.166413}$	0.442029	0.465326
	Nearest neighbor classifier	$\boxed{0.000000}$	0.228696	$\boxed{0.000000}$	0.225761	$\boxed{0.000000}$	0.247283	$\boxed{0.000000}$	0.223804
	Parzen classifier	0.134964	0.167609	0.104167	0.186739	0.161534	0.166630	0.139493	$\boxed{0.161196}$
	Binary decision tree classifier	$\boxed{0.000000}$	0.235326	$\boxed{0.000000}$	0.239130	$\boxed{0.000000}$	0.243587	–	–
	Neural network classifier	0.161836	$\boxed{0.162935}$	0.163949	$\boxed{0.170109}$	0.168176	0.168043	0.164855	0.179565
COLHIST	Gaussian linear classifier	0.143418	0.142174	0.154891	0.154565	0.095411	0.098804	0.420592	0.451848
	Gaussian quadratic classifier	0.053442	0.067826	0.110507	0.116304	0.082729	0.089565	0.084239	0.094348
	Fisher's linear classifier	0.143418	0.142174	0.154891	0.154565	0.095411	0.098804	0.420592	0.451848
	Logistic linear classifier	0.041667	0.058370	0.094203	$\boxed{0.100652}$	0.048007	0.065543	0.420592	0.451957
	Scaled nearest mean classifier	0.173309	0.169674	0.197766	0.196196	0.122886	0.118587	0.417874	0.436630
	Nearest neighbor classifier	$\boxed{0.000000}$	0.085109	$\boxed{0.000000}$	0.146957	$\boxed{0.000000}$	0.085978	$\boxed{0.000000}$	0.064239
	Parzen classifier	0.035326	0.064239	0.056159	0.108478	0.037440	0.066196	0.010266	$\boxed{0.063261}$
	Binary decision tree classifier	$\boxed{0.000000}$	0.092174	$\boxed{0.000000}$	0.143587	$\boxed{0.000000}$	0.094783	$\boxed{0.000000}$	0.095000
	Neural network classifier	0.040157	$\boxed{0.056957}$	0.113225	0.111413	0.045894	$\boxed{0.064783}$	0.049517	0.102283

Table 4.4: Classification performance in terms of classification error for training and testing datasets using different classifiers and different feature vectors in the probability and feature spaces for the COREL Database. The best performing classifiers (that gave the smallest classification errors) for particular feature vectors and similarity models are marked by boxes.

		Classification error in probability space						Classification error in feature space	
		MVG		FIT		GMIX			
Feature	Classifier	Training	Testing	Training	Testing	Training	Testing	Training	Testing
LAR+COOC	Gaussian linear classifier	0.280710	0.325900	0.288704	0.290927	0.312469	0.364745	0.485741	0.493389
	Gaussian quadratic classifier	0.301451	0.340511	0.305432	0.307217	0.341883	0.380598	$\boxed{0.298611}$	$\boxed{0.335233}$
	Fisher's linear classifier	0.280710	0.325900	0.288704	0.290927	0.312469	0.364745	0.485741	0.493389
	Logistic linear classifier	$\boxed{0.251204}$	$\boxed{0.302469}$	$\boxed{0.279198}$	$\boxed{0.282617}$	$\boxed{0.310648}$	$\boxed{0.361275}$	0.485772	0.493407
	Scaled nearest mean classifier	0.423519	0.428958	0.285586	0.289009	0.413642	0.416247	0.489660	0.492275
GABOR	Gaussian linear classifier	0.226265	0.267660	0.323086	0.338136	0.224352	0.267368	0.484043	0.495215
	Gaussian quadratic classifier	0.229136	0.266272	0.356204	0.355413	0.227716	0.266400	$\boxed{0.227809}$	$\boxed{0.279239}$
	Fisher's linear classifier	0.226265	0.267660	0.323086	0.338136	0.224352	0.267368	0.484043	0.495215
	Logistic linear classifier	$\boxed{0.186019}$	$\boxed{0.249251}$	$\boxed{0.322006}$	$\boxed{0.333187}$	$\boxed{0.183858}$	$\boxed{0.250475}$	0.484043	0.495142
	Scaled nearest mean classifier	0.412191	0.400888	0.342315	0.334740	0.412531	0.399007	0.489938	0.493517
MOMENTS	Gaussian linear classifier	0.291667	0.318559	0.340031	0.339944	0.292870	0.317974	0.486389	0.494430
	Gaussian quadratic classifier	0.306574	0.330046	0.371481	0.361513	0.307994	0.329078	$\boxed{0.296759}$	$\boxed{0.327215}$
	Fisher's linear classifier	0.291667	0.318559	0.340031	0.339944	0.292870	0.317974	0.486389	0.494430
	Logistic linear classifier	$\boxed{0.264105}$	$\boxed{0.305866}$	$\boxed{0.336111}$	$\boxed{0.336584}$	$\boxed{0.265062}$	$\boxed{0.305720}$	0.486389	0.494430
	Scaled nearest mean classifier	0.409167	0.406239	0.372562	0.365859	0.407809	0.404467	0.486821	0.492786
TAMURA	Gaussian linear classifier	0.338272	0.360381	0.335648	0.350975	0.369938	0.373420	0.491574	0.494503
	Gaussian quadratic classifier	0.359228	0.376160	0.368549	0.374187	0.420648	0.427844	$\boxed{0.324475}$	$\boxed{0.340328}$
	Fisher's linear classifier	0.338272	0.360381	0.335648	0.350975	0.369938	0.373420	0.491574	0.494503
	Logistic linear classifier	$\boxed{0.331265}$	0.358006	$\boxed{0.334198}$	$\boxed{0.349240}$	$\boxed{0.346790}$	$\boxed{0.355541}$	0.491574	0.494503
	Scaled nearest mean classifier	0.340340	$\boxed{0.350153}$	0.336265	0.349624	0.370247	0.374370	0.491821	0.493444
COLHIST	Gaussian linear classifier	0.141173	0.184528	0.245031	0.259935	0.139907	0.181861	0.483272	0.493590
	Gaussian quadratic classifier	0.121111	0.173150	0.252778	0.270619	0.118210	0.171963	$\boxed{0.126111}$	$\boxed{0.202334}$
	Fisher's linear classifier	0.141173	0.184528	0.245031	0.259935	0.139907	0.181861	0.483272	0.493590
	Logistic linear classifier	$\boxed{0.117809}$	$\boxed{0.172292}$	$\boxed{0.244074}$	$\boxed{0.254858}$	$\boxed{0.113488}$	$\boxed{0.168237}$	0.483272	0.493590
	Scaled nearest mean classifier	0.332932	0.351870	0.311142	0.312550	0.331667	0.350756	0.485154	0.490759

tive classification by training only simple linear classifiers in the probability space. (These results also agree with those of Duin [65].) The average classification error for the COREL Database was larger than the average classification errors for other databases. This is an expected result because of the complexity of this database.

Among the feature vectors, Gabor and color histogram performed better than others while the line-angle-ratio and co-occurrence feature vectors performed better than moments and Tamura feature vectors. The Logistic linear classifier in the probability space was usually the best performing classifier among all the others in both probability and feature spaces.

Using mixtures of Gaussians did not give an improvement over the single multivariate Gaussian case. This was because of the fact that it was hard to estimate multivariate distributions in the high dimensional space from a small amount of data, and one component usually dominated the others. The multivariate Gaussian model also usually performed better than the independently fitted distributions model because of its handling of the correlations between features. The results of classification in the feature space where the Gaussian quadratic classifier was one of the best performing classifiers also support this observation.

This significant performance of the multivariate Gaussian model shows us that simple models are worth trying before using any of the more complex models because they are often quite effective, do not require extra effort to tune in too many parameters, and do not suffer from the local extrema and convergence problems during the estimation of more complex models. The fact that we often had problems of overfitting in complex models like ICA or Gaussian mixtures shows us that although these models can be effective in many situations, they become impractical in higher dimensions and also when a very large amount of training data is not available. This brings out the question of finding effective models that can both capture the intrinsic structure in high-dimensional data and give a good estimate of its distribution. An interesting research problem can be to develop mixtures of sub-manifolds where both the dimension of each sub-manifold and the number of sub-manifolds in the mixture are determined automatically.

Although most of the classifiers have similar error rates, sets of image pairs misclassified by different classifiers do not necessarily overlap. Classifier performance can be further improved by combining the decisions made by individual classifiers. This will be investigated in Section 7.2.

4.5.2 Choosing p for the L_p Metric

The p values that were used in the retrieval experiments below were chosen using the approach described in Section 4.3.2. For each normalization method, we computed the classification error for p in the range [0.2,5]. The results are given in Figures 4.9 - 4.11. Normalization using transformation with the cumulative distribution function (Norm.3) and rank normalization (Norm.4) gave the smallest classification error in almost all cases. These methods also resulted in relatively flat classification error curves around the best performing p values which showed that a larger range of p values were performing similarly well. Therefore, flat minima are indicative of a more robust method. All the other methods were also more sensitive to the choice of p and both the classification error and the average precision changed fast with smaller changes in p.

The best performing normalization methods had larger p values but the other methods had p values around 1. Given the structure of the L_p metric, a few relatively large differences can effect the results significantly for larger p values. On the other hand, smaller p values are less sensitive to large differences. Therefore, smaller p values tend to make a distance more robust to large differences. This is consistent with the fact that L_1 regression is more robust than least squares with respect to outliers [166]. This shows that the normalization methods other than Norm.3 and Norm.4 resulted in relatively unbalanced feature spaces and smaller p values tried to reduce this effect in the L_p metric. This is also consistent with our earlier experiments where the city-block (L_1) distance performed better than the Euclidean (L_2) distance [8]. Sebe *et al.* [179] fitted Gaussians and Double Exponentials to feature differences and showed that L_2 and L_1 distances correspond to the maximum likelihood estimates using these fitted distributions. They concluded that L_1 performed better than L_2 and this was consistent with the fitting results. In [180] they also used Cauchy distribution fits and the Cauchy-based distance performed better than both L_1 and L_2.

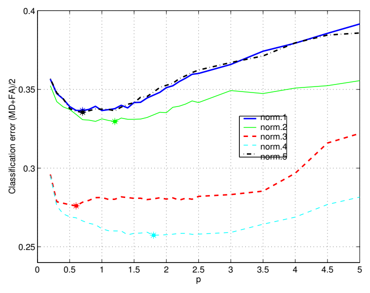
We also did retrieval experiments for different values of p. The values of p that resulted in the smallest classification errors and the largest average precisions were consistent. Therefore, the classification scheme presented in Section 4.3.2 proved effective in deciding which p to use. The p values that gave the smallest classification error for each normalization method were used in the retrieval experiments of the following section.

4.5.3 Retrieval Performance

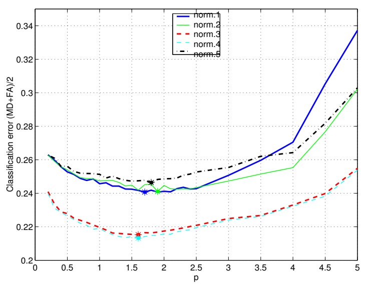
Extensive retrieval experiments to test the effectiveness of different feature vectors and similarity models are done and average precision and recall for all databases are given in Figures 4.12 - 4.17. Each plot shows average precision vs. recall for either a particular feature vector and similarity model combination for all normalization models or a particular feature vector and normalization method for all similarity models. The experiments of this section correspond to the similarity model level in Figure 4.2, i.e. the likelihood ratio computed from the class-conditional probabilities for the relevance and irrelevance classes for a particular similarity model trained using a particular feature vector.

The results of retrieval experiments were consistent with those of the classification experiments discussed in Section 4.5.1. The feature vector and similarity model combinations that gave the smallest classification error also gave the largest precision and recall.

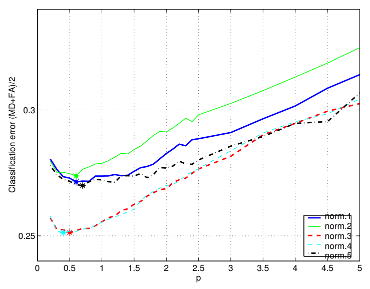
The best results were obtained for the VisTex Database which is the simplest



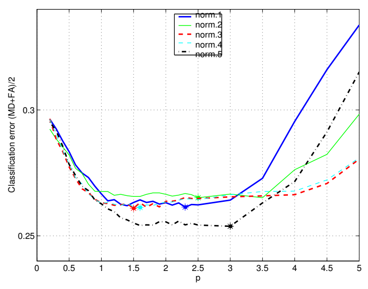
(a) Line-angle-ratio and co-occurrence



(b) Gabor

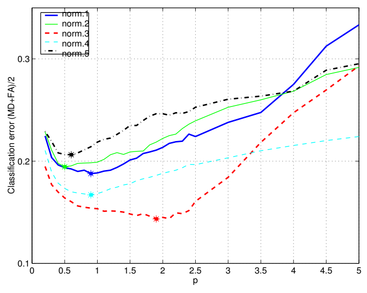


(c) Moments

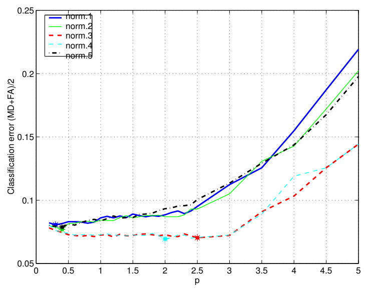


(d) Tamura

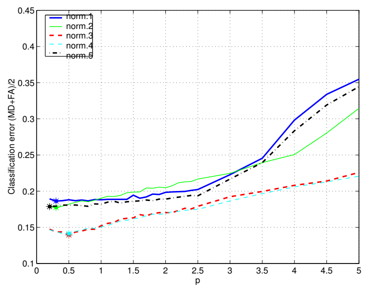
Figure 4.9: Classification error vs. p value in the Minkowsky L_p metric for different normalization methods and feature vectors for the ISL Database. The best p value for each method is marked with a star. The curves represent the normalization methods linear scaling to unit range (Norm.1, blue), linear scaling to unit variance (Norm.2, green), transformation using cumulative distribution function (Norm.3, red), rank normalization (Norm.4, cyan), and normalization after fitting distributions (Norm.5, black).



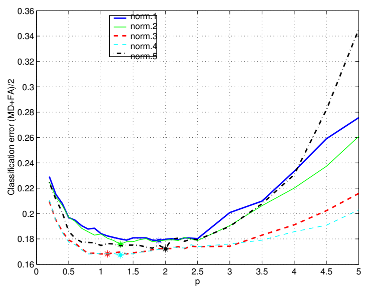
(a) Line-angle-ratio and co-occurrence



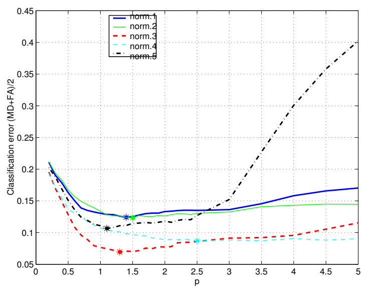
(b) Gabor



(c) Moments

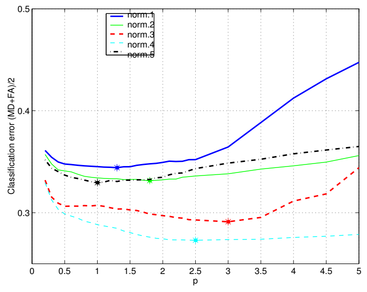


(d) Tamura

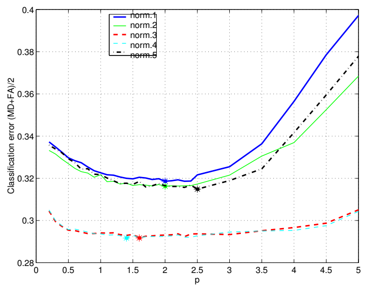


(e) Color histogram

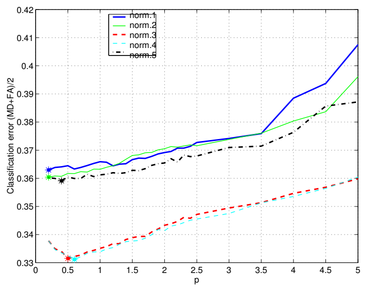
Figure 4.10: Classification error vs. p value in the Minkowsky L_p metric for different normalization methods and feature vectors for the VisTex Database. The best p value for each method is marked with a star. The curves represent the normalization methods linear scaling to unit range (Norm.1, blue), linear scaling to unit variance (Norm.2, green), transformation using cumulative distribution function (Norm.3, red), rank normalization (Norm.4, cyan), and normalization after fitting distributions (Norm.5, black).



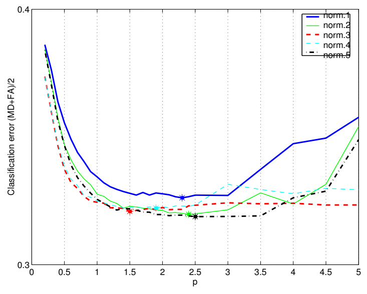
(a) Line-angle-ratio and co-occurrence



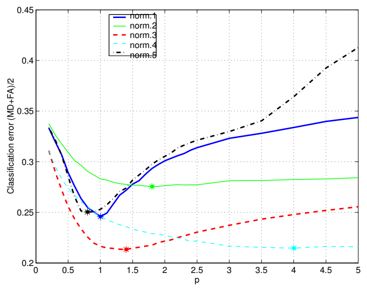
(b) Gabor



(c) Moments



(d) Tamura



(e) Color histogram

Figure 4.11: Classification error vs. p value in the Minkowsky L_p metric for different normalization methods and feature vectors for the COREL Database. The best p value for each method is marked with a star. The curves represent the normalization methods linear scaling to unit range (Norm.1, blue), linear scaling to unit variance (Norm.2, green), transformation using cumulative distribution function (Norm.3, red), rank normalization (Norm.4, cyan), and normalization after fitting distributions (Norm.5, black).

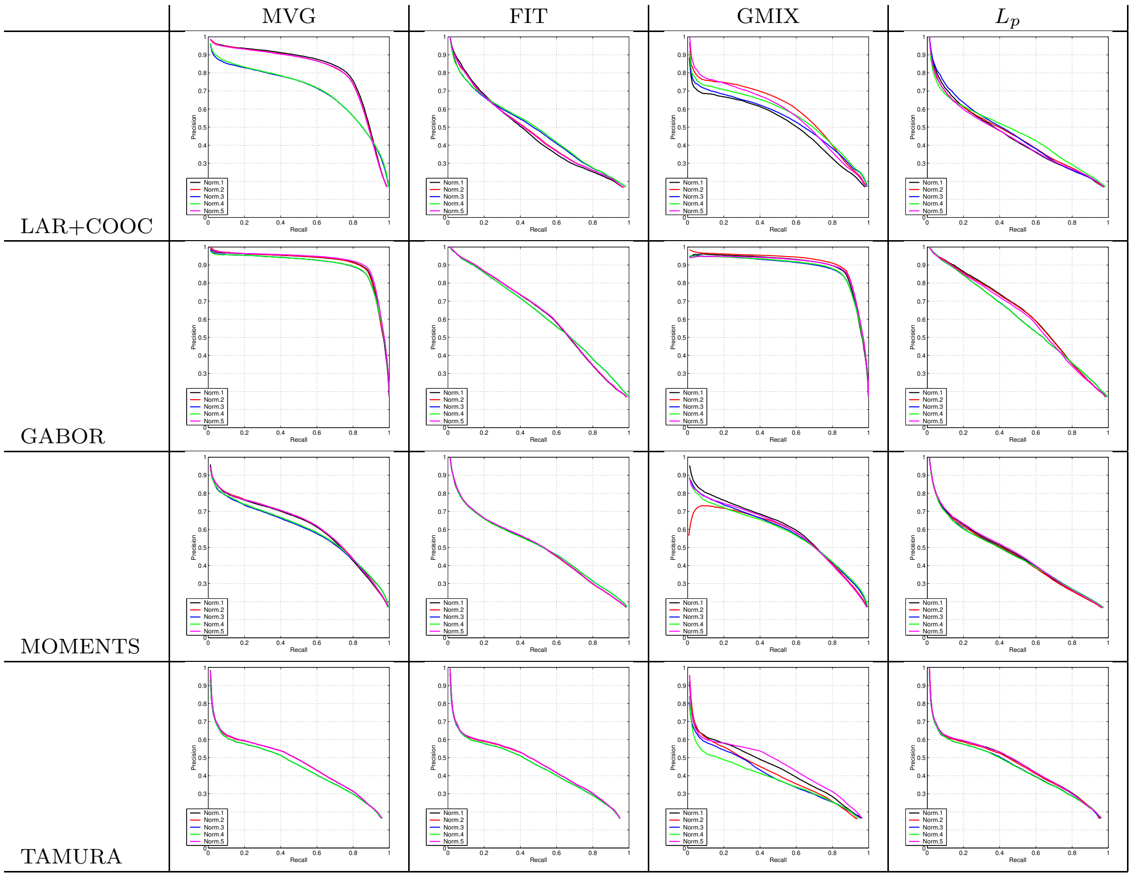


Figure 4.12: Retrieval performance for different normalization methods for the ISL Database. Each plot shows average precision (y-axis) vs. recall (x-axis) for a particular feature vector and similarity model combination. Different curves within the same plot represent the normalization methods linear scaling to unit range (Norm.1, black), linear scaling to unit variance (Norm.2, red), transformation using cumulative distribution function (Norm.3, blue), rank normalization (Norm.4, green), and normalization after fitting distributions (Norm.5, magenta).

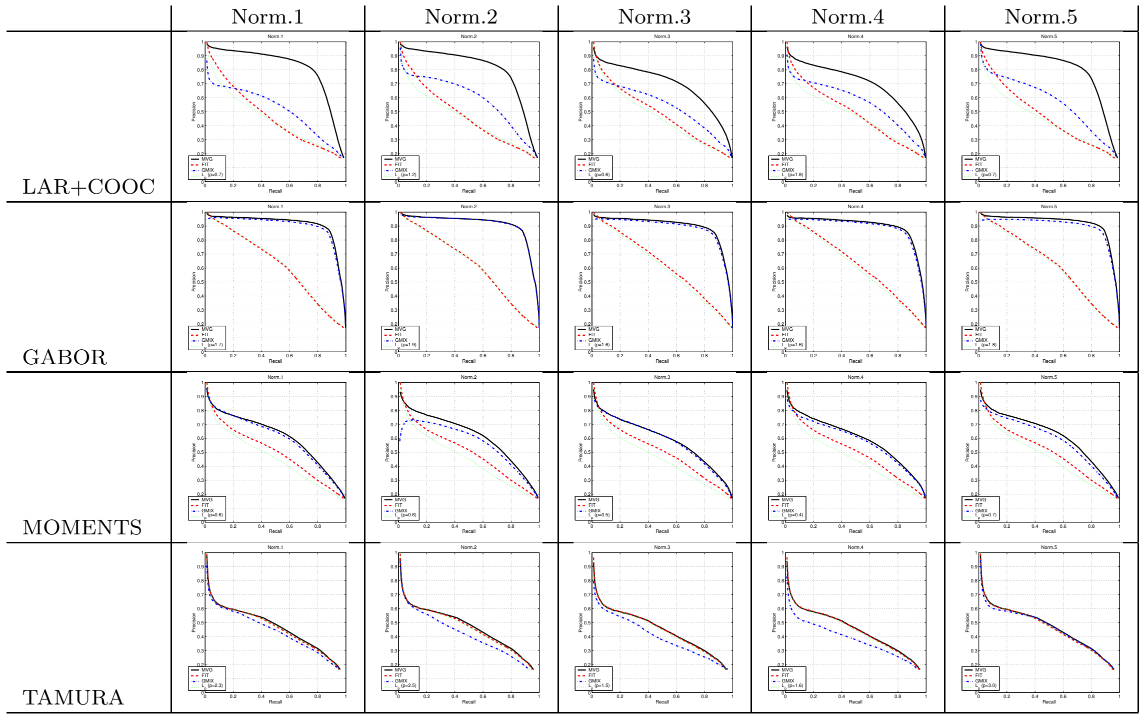


Figure 4.13: Retrieval performance for different similarity models for the ISL Database. Each plot shows average precision (y-axis) vs. recall (x-axis) for a particular feature vector and normalization method. Different curves within the same plot represent the similarity models multivariate Gaussian (black, solid), independently fitted distributions (red, dashed), mixture of Gaussians (blue, dash-dot), and Minkowsky L_p metric (green, dotted).

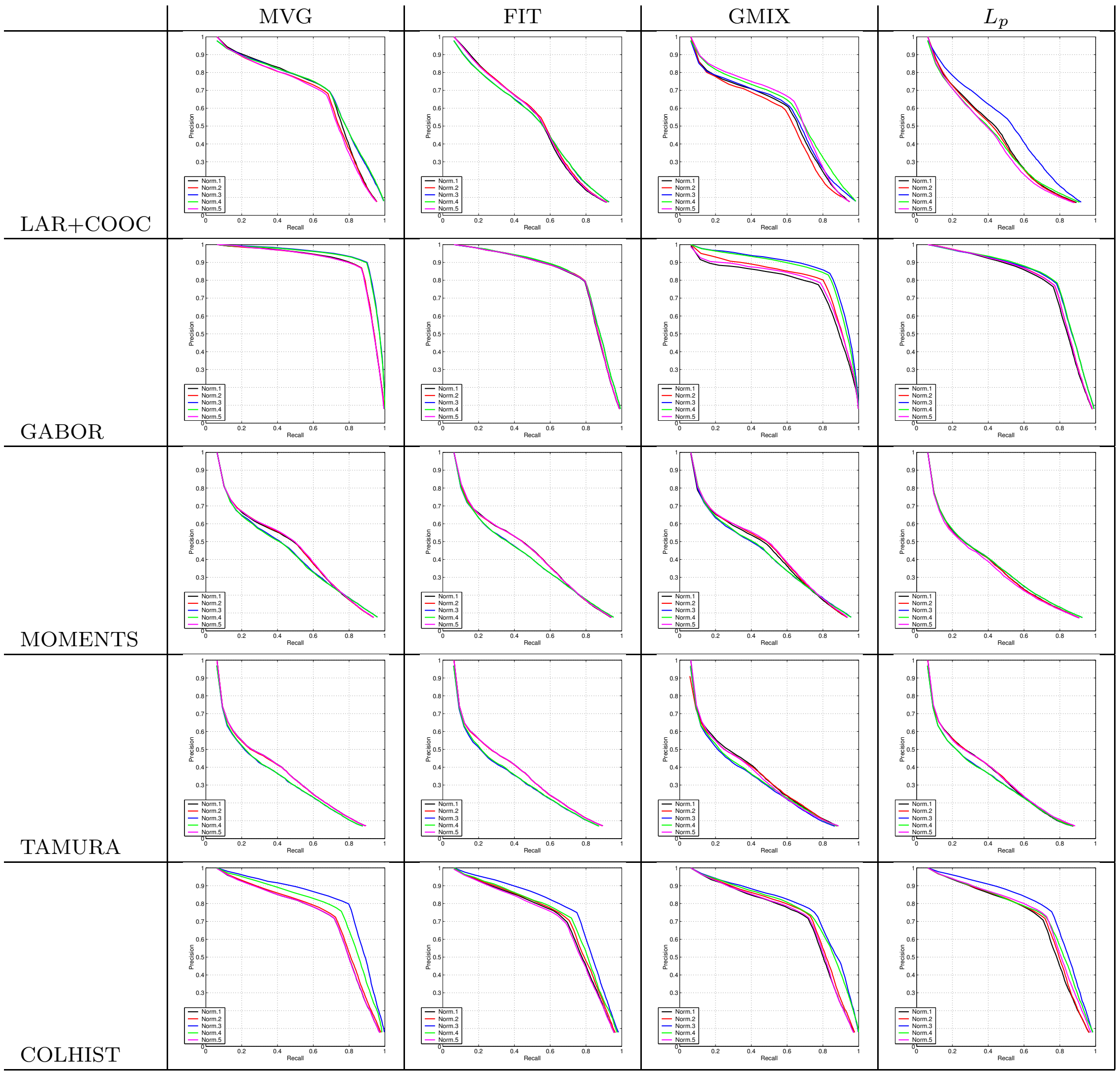


Figure 4.14: Retrieval performance for different normalization methods for the VisTex Database. Each plot shows average precision (y-axis) vs. recall (x-axis) for a particular feature vector and similarity model combination. Different curves within the same plot represent the normalization methods linear scaling to unit range (Norm.1, black), linear scaling to unit variance (Norm.2, red), transformation using cumulative distribution function (Norm.3, blue), rank normalization (Norm.4, green), and normalization after fitting distributions (Norm.5, magenta).

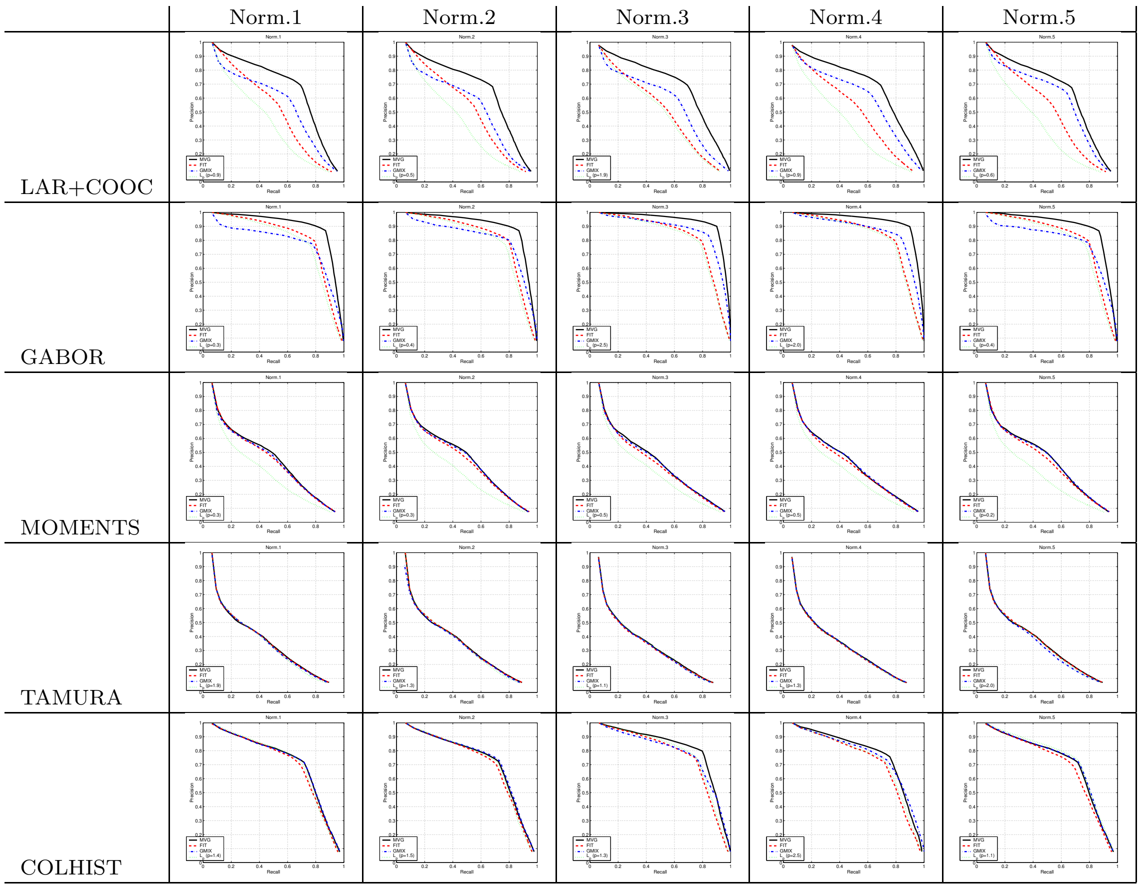


Figure 4.15: Retrieval performance for different similarity models for the VisTex Database. Each plot shows average precision (y-axis) vs. recall (x-axis) for a particular feature vector and normalization method. Different curves within the same plot represent the similarity models multivariate Gaussian (black, solid), independently fitted distributions (red, dashed), mixture of Gaussians (blue, dash-dot), and Minkowsky L_p metric (green, dotted).

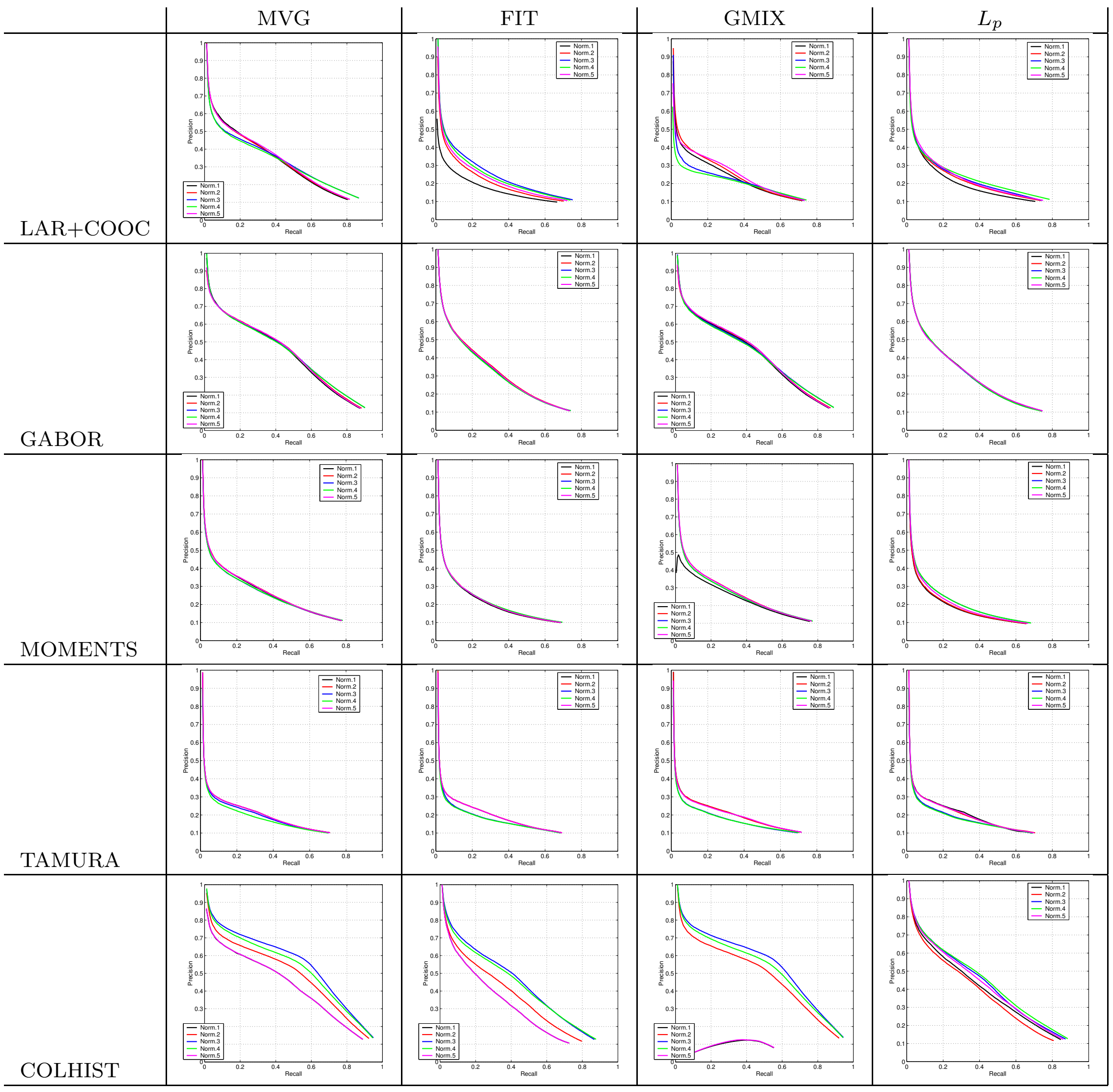


Figure 4.16: Retrieval performance for different normalization methods for the COREL Database. Each plot shows average precision (y-axis) vs. recall (x-axis) for a particular feature vector and similarity model combination. Different curves within the same plot represent the normalization methods linear scaling to unit range (Norm.1, black), linear scaling to unit variance (Norm.2, red), transformation using cumulative distribution function (Norm.3, blue), rank normalization (Norm.4, green), and normalization after fitting distributions (Norm.5, magenta).

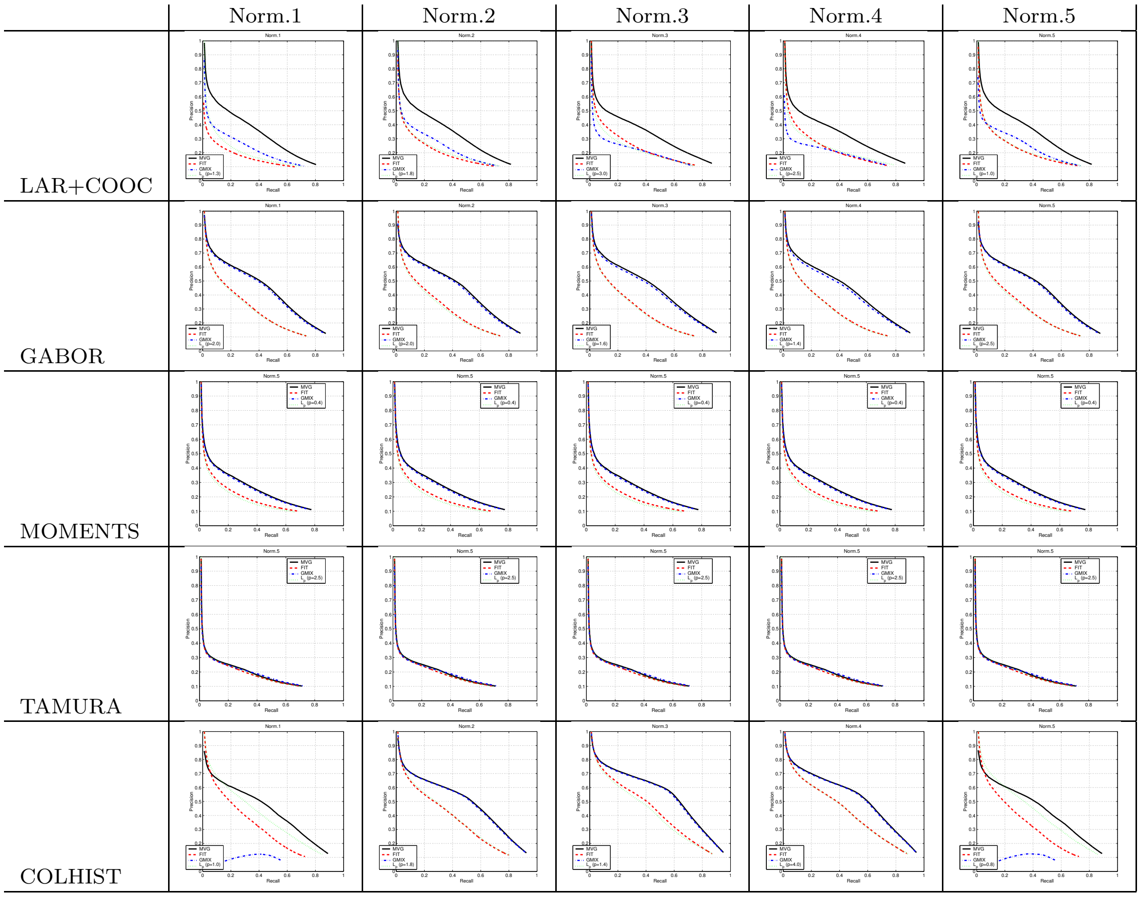


Figure 4.17: Retrieval performance for different similarity models for the COREL Database. Each plot shows average precision (y-axis) vs. recall (x-axis) for a particular feature vector and normalization method. Different curves within the same plot represent the similarity models multivariate Gaussian (black, solid), independently fitted distributions (red, dashed), mixture of Gaussians (blue, dash-dot), and Minkowsky L_p metric (green, dotted).

of all three databases we used. Performance on the ISL Database was also better than the one on the COREL Database. Probabilistic similarity measures always performed better than the geometric measures. The most successful similarity model was the multivariate Gaussian (MVG). It was both easy to compute and simple to train. The best performing feature vector was the Gabor model. Color histograms (COLHIST) were also effective for color images. Line-angle-ratio and co-occurrence (LAR+COOC) feature vectors were the second most successful texture model.

The performances relative to the normalization methods were also consistent with the class-separability results of Section 3.5. Normalization using fitted distributions (Norm.5) was often the most successful with the exception that normalization using the cumulative distribution function (Norm.3) worked the best for color histogram feature vectors. The reason for this may be that the continuous distributions used for fitting as in Section 3.3.5 could not model the histogram-based (discrete) features effectively. Other than that, there were no significant differences between the performances of different normalization methods. Our earlier experiments [8, 10] showed that the performances of geometric similarity measures were quite dependent on the normalization method used when the same p value was used in the Minkowsky L_p metric (e.g. L_1 or L_2) for all normalization methods. However, this problem was not observed when the p value optimal in terms of minimizing the classification error was found for a particular model as described in Section 4.3.2.

Example queries are given in Figures 4.18 - 4.20. The first three rows in the user interface show the best 12 matches and the last row shows the worst 4 matches to the query. (The user interface was described in detail in Section 1.5.4.) These queries illustrate the differences in the effectiveness of retrieval using geometric similarity measures and probabilistic similarity measures with the same feature vector. Due to space limitations we decided to choose most of the examples from the queries where we do not have a perfect retrieval. It can be easily seen from the precision-recall curves that the probabilistic similarity measures achieve a successful retrieval especially for the ISL and VisTex Databases. Both visual examples and precision-recall curves show that probabilistic similarity outperforms the commonly used geometric similarity. Examples using the same query images will be given in Chapter 8 to illustrate the performances of some of the algorithms proposed in the rest of the dissertation.

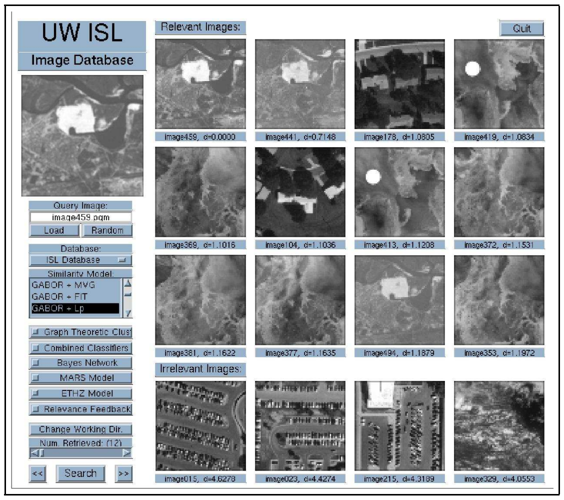
4.6 Summary of Observations

A summary of observations from the experiments presented in this chapter is given below.

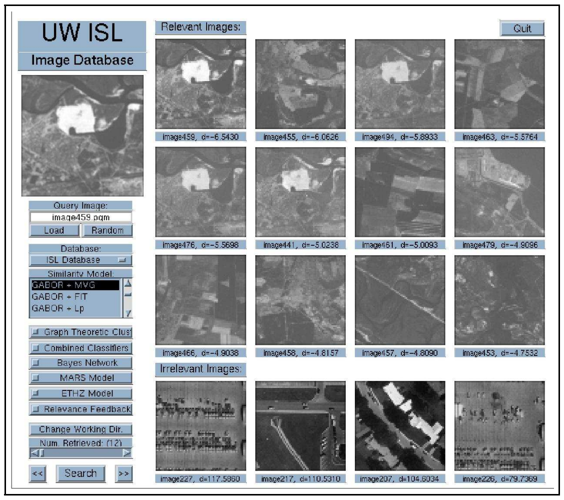
- Classification effectiveness in terms of minimizing the classification error reflects well on the retrieval performance in terms of precision and recall. The methods that gave the smallest classification error also gave the largest precision and recall. Therefore, one can do the design (parameter estimation, model selection, choosing thresholds, etc.) in the classification framework and expect better results in retrieval.

- The normalization methods that performed the best in terms of class separability as shown in Table 3.1 also performed the best in terms of precision and recall in retrieval experiments. We can conclude that studying the distributions of the features and using the results of this study significantly improves the results compared to making only general or arbitrary assumptions.

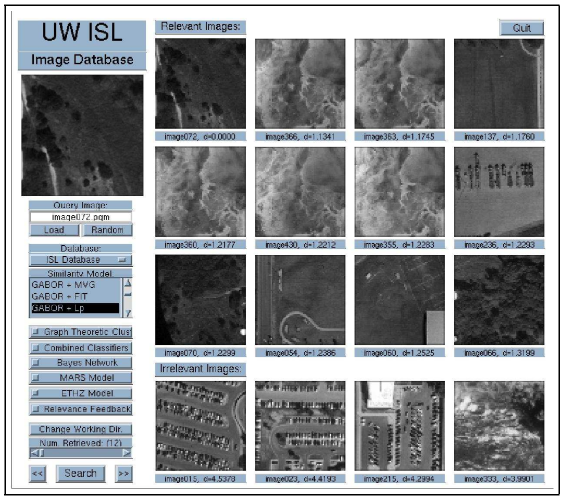
- Normalization using fitted distributions (Norm.5) was often the most successful normalization method. When a fixed p value was used for the Minkowsky L_p metric, normalization using the cumulative distribution function (Norm.3) or rank normalization (Norm.4) gave the best retrieval performance. When the p value was optimized according to the minimum error decision rule in Section 4.3.2, the Minkowsky metric was robust to normalization effects. Furthermore, values of p that gave the smallest classification error also gave the best precision.



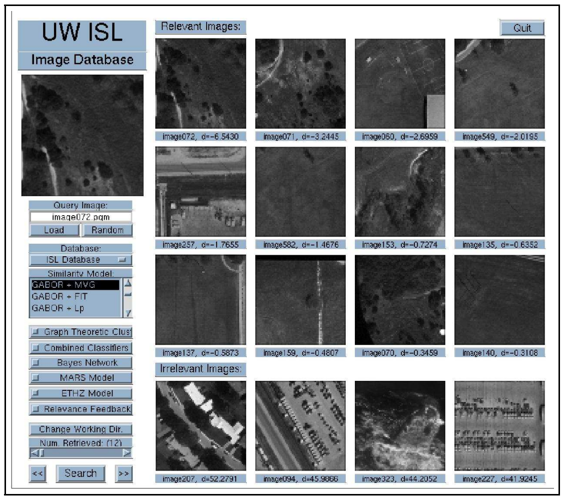
(a) An example query for LANDSAT Chernobyl using Gabor features and L_p metric (3/12)



(b) Same query for LANDSAT Chernobyl using Gabor features and multivariate Gaussian (12/12)

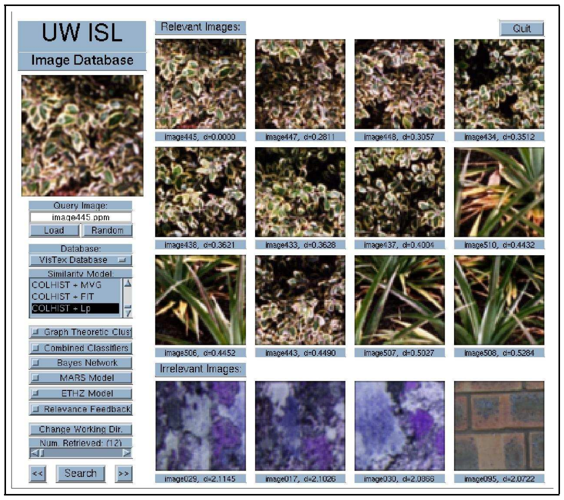


(c) An example query for landscape using Gabor features and L_p metric (6/12)



(d) Same query for landscape using Gabor features and multivariate Gaussian (11/12)

Figure 4.18: Example queries for the ISL Database using a single feature vector with geometric (left) or probabilistic (right) similarity measures. The numbers in parentheses in sub-captions show the number of correct matches for each case.

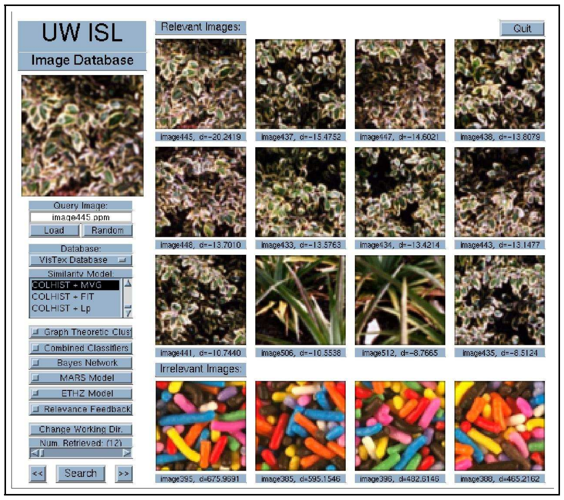


Query Image: image445.ppm
Database: VisTex Database
Similarity Model: COLHIST + MVG, COLHIST + FIT, COLHIST + Lp (selected: COLHIST + Lp)
Num. Retrieved: (12)

Relevant Images:
image445, d=0.0000 | image447, d=0.2811 | image448, d=0.3057 | image434, d=0.3512
image438, d=0.3621 | image433, d=0.3628 | image437, d=0.4004 | image510, d=0.4432
image506, d=0.4452 | image443, d=0.4490 | image507, d=0.5027 | image508, d=0.5284

Irrelevant Images:
image029, d=2.1145 | image017, d=2.1026 | image030, d=2.0866 | image095, d=2.0722

(a) An example query for leaves using color histograms and L_p metric (8/12)

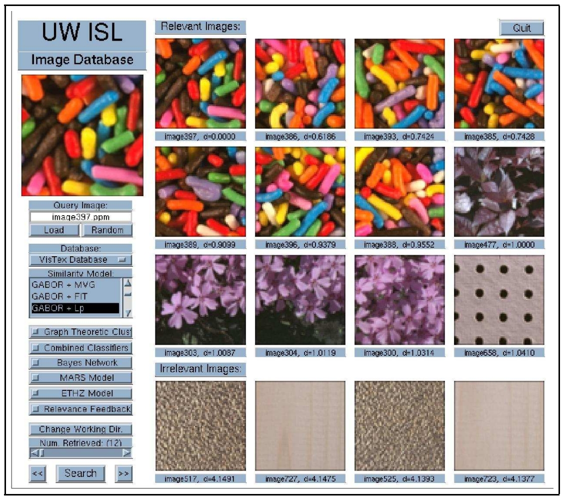


Query Image: image445.ppm
Database: VisTex Database
Similarity Model: COLHIST + MVG, COLHIST + FIT, COLHIST + Lp (selected: COLHIST + MVG)
Num. Retrieved: (12)

Relevant Images:
image445, d=-20.2419 | image437, d=-15.4752 | image447, d=-14.6021 | image438, d=-13.8079
image448, d=-13.7010 | image433, d=-13.5763 | image434, d=-13.4214 | image443, d=-13.1477
image441, d=-10.7440 | image506, d=-10.5538 | image512, d=-8.7665 | image435, d=-8.5124

Irrelevant Images:
image395, d=675.9691 | image385, d=595.1546 | image396, d=482.6146 | image388, d=465.2162

(b) Same query for leaves using color histograms and multivariate Gaussian (10/12)

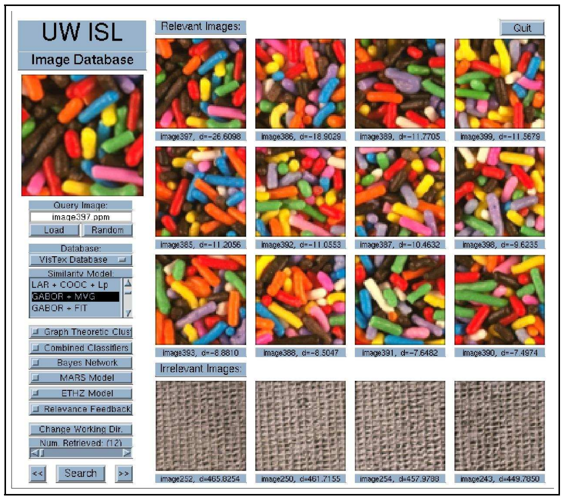


Query Image: image397.ppm
Database: VisTex Database
Similarity Model: GABOR + MVG, GABOR + FIT, GABOR + Lp (selected: GABOR + Lp)
Num. Retrieved: (12)

Relevant Images:
image397, d=0.0000 | image386, d=0.6186 | image393, d=0.7424 | image385, d=0.7428
image389, d=0.9099 | image396, d=0.9379 | image388, d=0.9552 | image477, d=1.0000
image303, d=1.0087 | image304, d=1.0119 | image300, d=1.0314 | image658, d=1.0410

Irrelevant Images:
image517, d=4.1491 | image727, d=4.1475 | image525, d=4.1393 | image723, d=4.1377

(c) An example query for food using Gabor features and L_p metric (7/12)



Query Image: image397.ppm
Database: VisTex Database
Similarity Model: LAR + COOC + Lp, GABOR + MVG, GABOR + FIT (selected: GABOR + MVG)
Num. Retrieved: (12)

Relevant Images:
image397, d=-26.6098 | image386, d=-18.9029 | image389, d=-11.7705 | image399, d=-11.5679
image385, d=-11.2056 | image392, d=-11.0553 | image387, d=-10.4632 | image398, d=-9.6235
image393, d=-8.8810 | image388, d=-8.5047 | image391, d=-7.6482 | image390, d=-7.4974

Irrelevant Images:
image252, d=465.8254 | image250, d=461.7155 | image254, d=457.9788 | image243, d=449.7850

(d) Same query for food using Gabor features and multivariate Gaussian (12/12)

Figure 4.19: Example queries for the VisTex Database using a single feature vector with geometric (left) or probabilistic (right) similarity measures. The numbers in parentheses in sub-captions show the number of correct matches for each case.

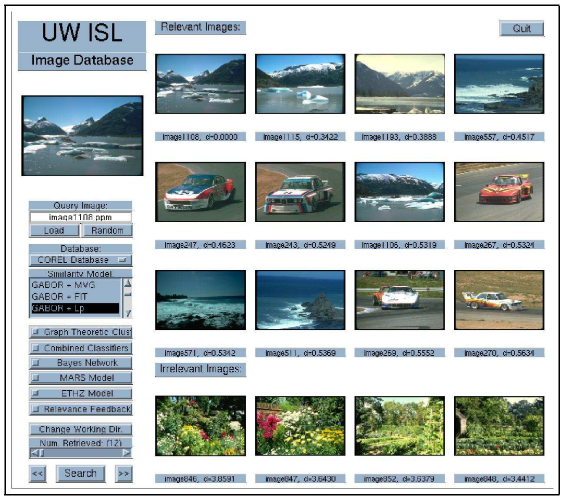


image1108, d=0.0000 · image1115, d=0.3422 · image1193, d=0.3888 · image557, d=0.4517
image247, d=0.4623 · image243, d=0.5249 · image1106, d=0.5319 · image267, d=0.5324
image571, d=0.5342 · image511, d=0.5369 · image269, d=0.5552 · image270, d=0.5634
image846, d=3.8591 · image847, d=3.6430 · image852, d=3.6379 · image848, d=3.4412

(a) An example query for glaciers and mountains using Gabor features and L_p metric (4/12)

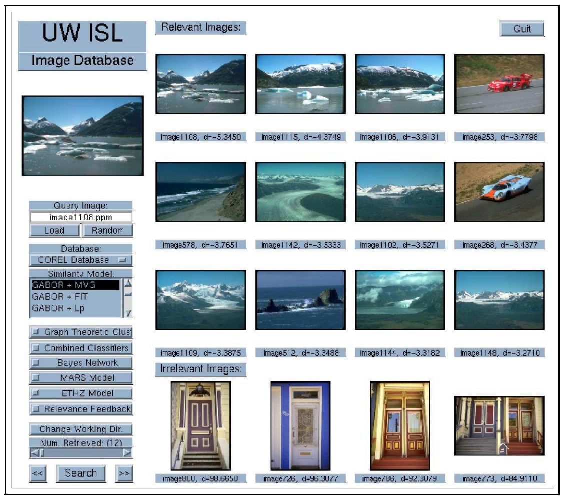


image1108, d=-5.3450 · image1115, d=-4.3749 · image1106, d=-3.9131 · image253, d=-3.7798
image578, d=-3.7651 · image1142, d=-3.5333 · image1102, d=-3.5271 · image268, d=-3.4377
image1109, d=-3.3875 · image512, d=-3.3488 · image1144, d=-3.3182 · image1148, d=-3.2710
image800, d=98.6650 · image726, d=96.3077 · image786, d=92.3079 · image773, d=84.9110

(b) Same query for glaciers and mountains using Gabor features and multivariate Gaussian (8/12)

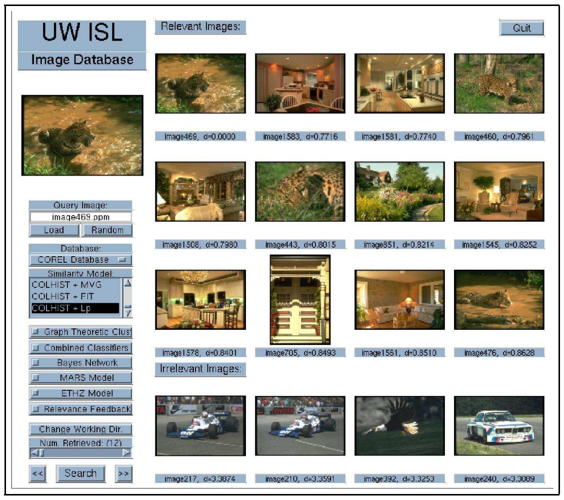


image469, d=0.0000 · image1583, d=0.7716 · image1581, d=0.7740 · image460, d=0.7961
image1508, d=0.7980 · image443, d=0.8015 · image851, d=0.8214 · image1545, d=0.8252
image1578, d=0.8401 · image705, d=0.8493 · image1561, d=0.8510 · image476, d=0.8628
image217, d=3.3874 · image210, d=3.3591 · image392, d=3.3253 · image240, d=3.3089

(c) An example query for cheetahs using color histograms and L_p metric (4/12)

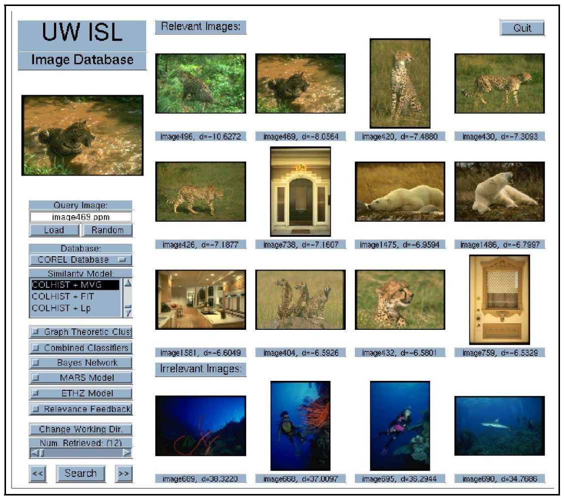
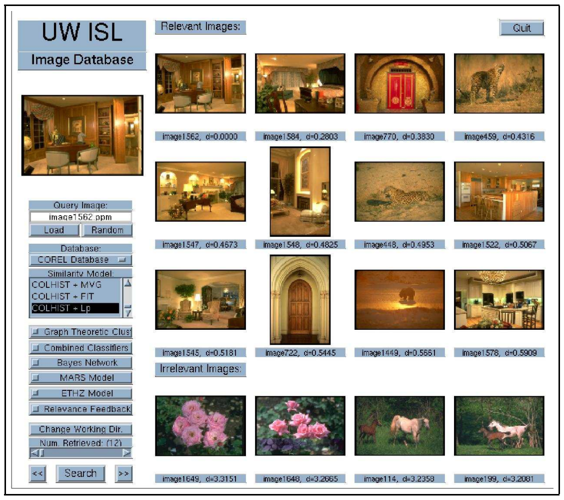


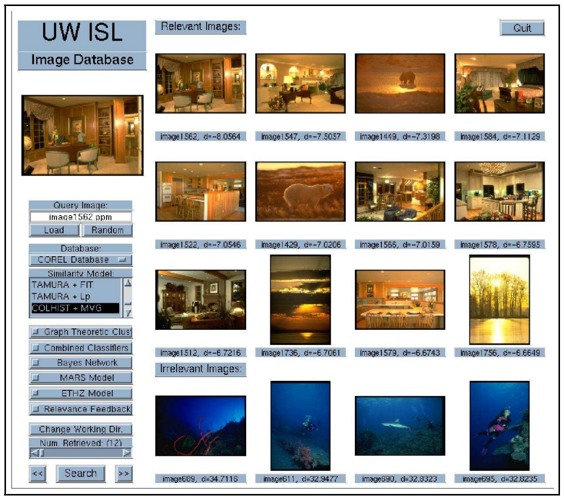
image496, d=-10.6272 · image469, d=-8.0564 · image420, d=-7.4880 · image430, d=-7.3093
image426, d=-7.1877 · image738, d=-7.1607 · image1475, d=-6.9594 · image1486, d=-6.7997
image1581, d=-6.6049 · image404, d=-6.5926 · image432, d=-6.5801 · image759, d=-6.5329
image689, d=38.3220 · image668, d=37.0097 · image695, d=36.2944 · image690, d=34.7686

(d) Same query for cheetahs using color histograms and multivariate Gaussian (7/12)

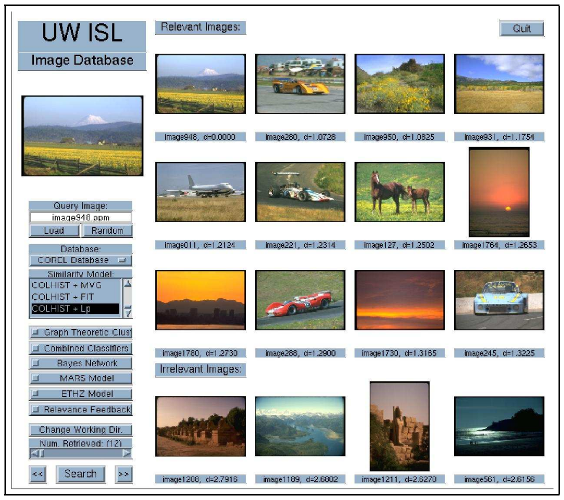
Figure 4.20: Example queries for the COREL Database using a single feature vector with geometric (left) or probabilistic (right) similarity measures. The numbers in parentheses in sub-captions show the number of correct matches for each case.



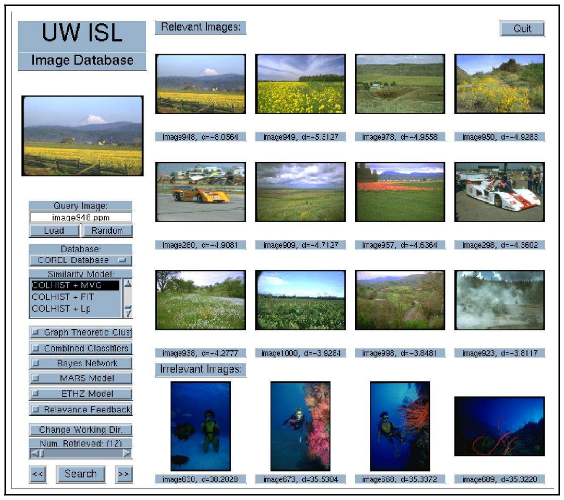
(a) An example query for residential interiors using color histograms and L_p metric (7/12)



(b) Same query for residential interiors using color histograms and multivariate Gaussian (8/12)



(c) An example query for fields using color histograms and L_p metric (3/12)



(d) Same query for fields using color histograms and multivariate Gaussian (9/12)

Figure 4.20: (continued)

- Simple linear classifiers (e.g. Gaussian or Logistic) in the probability space performed better than non-linear classifiers (e.g. decision tree and neural network) in the feature space. This result leads us to do effective classification and retrieval by training only simple linear classifiers in the probability space.

- Although the error rates for most of the classifiers are similar, image pairs misclassified by different classifiers do not necessarily overlap. Combinations of the decisions of multiple classifiers will be studied in Chapter 7.

- The best results were obtained for the VisTex Database. This is an expected result because that database consists of mostly homogeneous texture patches. COREL was the hardest database because of the large variations among its images.

- Gabor and color histogram feature vectors performed better than others where line-angle-ratio and co-occurrence feature vectors performed better than moments and Tamura feature vectors.

- The correspondences between classification performances in the probability space vs. the feature space, and retrieval performances of the probabilistic similarity measures vs. the geometric similarity measures show that our probabilistic framework for retrieval is much more effective than the commonly used geometric framework.

- The multivariate Gaussian model performed better than the other probability models for the estimation of class-conditional distributions in probabilistic similarity measures. (In most of the cases, Gaussians were also the best fits to individual feature components marginally.) This significant performance of the multivariate Gaussian model in addition to the problems of overfitting that we often had with models like Independent Component Analysis and mixtures of

Gaussians shows us that simple models are worth trying before using any of the more complex models because they are often quite effective, do not require extra effort to tune in too many parameters, and do not suffer from the local extrema and convergence problems of more complex models.

- The best performing configuration for all datasets include the Gabor and color histogram feature vectors, normalization using fitted distributions, multivariate Gaussian model, and Logistic linear classifier. However, precision and recall for different groundtruth groups show that some of the groups have worse performances than others and different features perform differently for different images. This gives the motivation for the feature and similarity combination models in the following chapters.

Chapter 5

GRAPH-THEORETIC CLUSTERING FOR IMAGE GROUPING AND RETRIEVAL

5.1 Motivation

Computing feature vectors is one of the most essential steps in image retrieval algorithms like in many computer vision and pattern recognition applications. In the previous chapters we used feature vectors for image representation and then described different similarity measures to find similarities between these representations. These similarity measures, explicitly or implicitly, use the assumption that visually similar images are close to each other in the feature space. The high-dimensionality of the feature vectors make the characterization of this space quite hard and, unfortunately, none of the existing feature extraction algorithms can always map visually similar images to nearby locations in the feature space. A common observation in retrieval results is that sometimes images that are quite irrelevant to the query image are also retrieved simply because they are close to the query image. We believe that an efficient retrieval algorithm should be able to retrieve images that are not only close (similar) to the query image but also close (similar) to each other.

Another important issue is that the feature extraction algorithms may involve many parameters to be adjusted. Most of the times the feature selection process is done by trial and error. One of the main reasons why a smaller but more effective subset of features is not sought is that formulating a statistical feature selection problem is often impossible because the probability distributions of the features may not be known or an optimization problem involving "goodness" of features as an objective function is hard to formulate and expensive to compute.

Clustering the feature space and visually examining the results to check whether visually similar images are actually close to each other is an important step in understanding the behavior of the features and the structure of the feature space. This can help us determine the effectiveness of both the features and the distance measures in establishing similarity between images. In their Blobworld system, Carson *et al.* [37] used an expectation-maximization-based algorithm to cluster the blob space to find representative blobs that can mimic human queries. They noted that their clustering procedure tends to ignore the blobs which have the best chance of distinguishing among categories because the most distinctive blobs in a given category occur much less often than less distinctive blobs.

In this chapter we introduce a graph-theoretic approach for image grouping and retrieval by formulating the database search as a graph clustering problem. The idea that clusters contain visually similar images is similar to the idea in Carson *et al.* [37] but we use the clusters in a post-processing step instead of forming the initial queries. The goal is to have an additional constraint that the retrieved images should be similar to each other as well as being similar to the query image, where similarity is determined by the models described in Chapter 4.

Graph-theoretic approaches have been a popular tool in the computer vision literature, especially in object matching. Recently, graphs were used in image segmentation by treating pixels as nodes and some features as edge weights, and defining criteria like the normalized cut [184] and variations between intensity differences [71] to measure the disassociations between possible partitions of the graph. Graphs did not receive significant attention in image retrieval algorithms mainly due to the computational complexity of graph-related operations. Huet and Hancock [98] used attributed graphs to represent line patterns in images and used these graphs for image matching and retrieval.

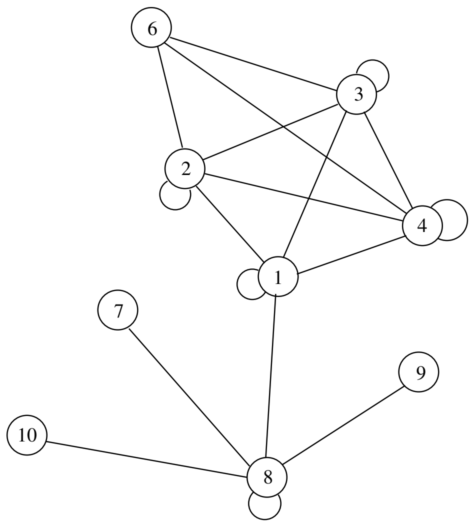
The rest of the chapter is organized as follows. The new image retrieval algorithm is described in Section 5.2 and is followed by the summary of a graph-theoretic clustering algorithm in Section 5.3. A probabilistic model for the graph-theoretic framework is proposed in Section 5.4. An alternative clustering algorithm that uses vector quantization is given in Section 5.5. A measure to evaluate resulting clusters is defined in Section 5.6. Experiments and results are presented in Section 5.7.

5.2 Image Grouping Using Graphs

In most of the retrieval algorithms a distance measure is used to rank the database images in ascending order of their distances to the query image, which is assumed to correspond to a descending order of similarity. Unfortunately, none of the existing feature extraction algorithms can always map visually similar images to nearby locations in the feature space and it is not uncommon to retrieve images that are quite irrelevant to the query image simply because they are close to it. We believe that an efficient retrieval algorithm should be able to retrieve images that are not only similar to the query image but also similar to each other, and propose a new retrieval algorithm as follows. Assume we query the database and get back the best n matches. For each of these n matches we do a query and get back the best n matches again. Define $\mathcal{S}$ as the set containing the original query image and the images that are retrieved as the results of the above queries. $\mathcal{S}$ will contain n images in the best case and n^2+1 images in the worst case. Then, we can construct a graph with the images in $\mathcal{S}$ as the nodes and can draw edges between each query image and each image in the retrieval set of that query image. We call these edges the set $\mathcal{R}$ where $\mathcal{R} = \{(\xi_i, \xi_j) \in \mathcal{S} \times \mathcal{S} \mid \text{image } \xi_j \text{ is in the retrieval set when image } \xi_i \text{ is the query}\}$. An example graph is given in Figure 5.1. The similarity values between images, which correspond to two nodes that an edge connects, can also be assigned as a weight to that edge. We want to find the connected clusters of this graph $(\mathcal{S}, \mathcal{R})$ because they correspond to similar images. The clusters of interest are the ones that include the

Query	Retrieved images				
1	1	2	4	3	8
2	2	1	3	4	6
3	3	2	1	4	6
4	4	1	3	2	6
5	5	7	2	6	10
6	6	2	3	4	5
7	7	5	8	10	9
8	8	7	9	10	1
9	9	8	7	4	10
10	10	7	8	9	5

(e) Results of all possible queries



(f) Constructed graph when image 1 is the original query

Figure 5.1: An example scenario for graph construction for a database of 10 images with $n = 5$.

original query image. The problem now becomes finding $\mathcal{P}$, where $\mathcal{P} \subseteq \mathcal{S}$ such that $\mathcal{P} \times \mathcal{P} \subseteq \mathcal{R}$. This is called a clique of the graph. The clique with the largest number of nodes is called the major or maximal clique. The images that correspond to the nodes in $\mathcal{P}$ can then be retrieved as the final result of the query.

An additional thing to consider is that the graph $(\mathcal{S}, \mathcal{R})$ can have multiple clusters and some of these clusters can overlap. This is a desired property because image content is too complex to be grouped into distinct categories. Hence, an image can be consistent with multiple groups of images.

Additional measures are required to select the cluster that will be returned as the result of the query. In the next section we define the term "compactness" for a set of nodes. The cluster with the largest compactness (or with the largest number of nodes) can be retrieved as the final result. If more than one such cluster exists, we can select the one with the largest number of nodes (or with the largest compactness), or we can compute the sum of the weights of the edges in each of those clusters and select the one with the smallest total weight.

This method increases the chance of retrieving similar images by not only ensuring that the retrieved images are similar to the query image, but also adding another constraint that they should be similar to each other. In the next section we describe a graph-theoretic clustering algorithm which is used to find the clusters.

5.3 Graph-Theoretic Clustering

In the previous section we proposed that cliques of the graph correspond to similar images. Since finding the cliques is computationally too expensive, we use the algorithm by Shapiro and Haralick [181] that finds "near-cliques" as dense regions instead of the maximally connected ones in the graph. To increase the speed further, the best n matches for the images in the database can be found offline so that graph clustering becomes the only overhead for a new query. Therefore, only one n-nearest neighbor search is required for a new query, which is the same amount of computation for the classical search methods.

In the following sections, first, we give some definitions, then, we describe the algorithm for finding dense regions, and finally, we present the algorithm for graph-theoretic clustering. The goal of this algorithm is to find regions in a graph, i.e. sets of nodes, which are not as dense as major cliques but are compact enough within user specified thresholds.

5.3.1 Definitions

- $(\mathcal{S}, \mathcal{R})$ represents a **graph** where $\mathcal{S}$ is the set of nodes and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$, the set of edges, is a symmetric binary relation on $\mathcal{S}$.

- $(X, Y) \in \mathcal{R}$ means node Y is a **neighbor** of node X. The set of all nodes Y

such that Y is a neighbor of X is called the **neighborhood** of X and is denoted by $\text{Neighborhood}(X) = \{Y \mid (X,Y) \in \mathcal{R}\}$.

- **Conditional density** $D(Y|X)$ is the number of nodes in the neighborhood of X which have Y as a neighbor, $D(Y|X) = \#\{Z \in \mathcal{S} \mid (Z,Y) \in \mathcal{R} \text{ and } (X,Z) \in \mathcal{R}\}$. Since $\mathcal{R}$ is symmetric in our case,

$$\begin{aligned} D(X|Y) &= D(Y|X) \\ &= \#\{\text{Neighborhood}(X) \cap \text{Neighborhood}(Y)\}. \end{aligned}$$

- Given an integer k, a **dense region** $\mathcal{V}$ around a node $X \in \mathcal{S}$ is defined as $\mathcal{V}(X,k) = \{Y \in \mathcal{S} \mid D(Y|X) \geq k\}$.

- $\mathcal{V}(X) = \mathcal{V}(X,k')$ is a **dense region candidate** around X where $k' = \max\{k \mid \#\mathcal{V}(X,k) \geq k\}$ because if $\mathcal{M}$ is a major clique of size t, then $X, Y \in \mathcal{M}$ implies that $D(Y|X) \geq t$. Thus $\mathcal{M} \subseteq \mathcal{V}(X,t)$ and $k \leq t \leq \#\mathcal{V}(X,k)$.

- **Association** of a node X to a subset $\mathcal{U}$ of $\mathcal{S}$ is defined as

$$A(X|\mathcal{U}) = \frac{\#\{\text{Neighborhood}(X) \cap \mathcal{U}\}}{\#\mathcal{U}} \tag{5.1}$$

where $0 \leq A(X|\mathcal{U}) \leq 1$.

- **Compactness** of a subset $\mathcal{U}$ of $\mathcal{S}$ is defined as

$$C(\mathcal{U}) = \frac{1}{\#\mathcal{U}} \sum_{X \in \mathcal{U}} A(X|\mathcal{U}) \tag{5.2}$$

where $0 \leq C(\mathcal{U}) \leq 1$.

5.3.2 Algorithm for Finding Dense Regions

A **dense region** $\mathcal{U}$ of the graph $(\mathcal{S},\mathcal{R})$ should satisfy

1. $\mathcal{U} = \{Z \in \mathcal{V}(X) \mid A(Z|\mathcal{V}(X)) \geq \text{MINASSOC}\}$ for some $X \in \mathcal{S}$,

2. $C(\mathcal{U}) \geq$ MINCOMP,

3. $\#\mathcal{U} \geq$ MINSIZE

where MINASSOC, MINCOMP and MINSIZE are thresholds supplied by the user. To determine the dense region around a node X,

1. Compute $D(Y|X)$ for every other node Y in $\mathcal{S}$.

2. Find a dense region candidate $\mathcal{V}(X, k')$ where $k' = \max\{k \mid \#\{Y|D(Y|X) \geq k\} \geq k\}$.

3. Remove the nodes with a low association (determined by the threshold MINASSOC) from the candidate set. Iterate until all of the nodes have high enough association.

4. Check whether the remaining nodes have high enough average association (determined by the threshold MINCOMP).

5. Check whether the candidate set is large enough (determined by the threshold MINSIZE).

When MINASSOC and MINCOMP are both 1, the resulting regions correspond to the cliques of the graph.

5.3.3 Algorithm for Graph-Theoretic Clustering

Given dense regions, to find the clusters of the graph,

1. Define the **dense-region relation** F as

$$\begin{aligned} F = \{(\mathcal{U}_1, \mathcal{U}_2) \mid\ & \mathcal{U}_1, \mathcal{U}_2 \text{ are dense regions of } \mathcal{R}, \\ & \frac{\#(\mathcal{U}_1 \cap \mathcal{U}_2)}{\#\mathcal{U}_1} \geq \text{MINOVERLAP or } \frac{\#(\mathcal{U}_1 \cap \mathcal{U}_2)}{\#\mathcal{U}_2} \geq \text{MINOVERLAP}\} \end{aligned} \quad (5.3)$$

where MINOVERLAP is a threshold supplied by the user. Merge the regions that have enough overlap if all of the nodes in the set resulting after merging have high enough associations.

2. Iterate until no regions can be merged.

The result is a collection of clusters in the graph. Note that a node can be a member of multiple clusters because of the overlap allowed between them.

For the example graph in Figure 5.1, resulting cluster for image 1 is $\{1, 2, 4, 3, 6\}$. Image 6 is retrieved instead of image 8 because it is more consistent with the rest of the retrieved images.

5.4 Probabilistic Model for Graph-Theoretic Retrieval

The previous section described how we can use graph-theoretic clustering for post-processing the query results. In this section, we describe a model to estimate the probability of each image being relevant to the query image under the graph-theoretic framework.

Let $\{\xi_1, \dots, \xi_m\}$ be the set of all m images in the database. Let $\mathcal{U}$ be the list of images retrieved during the iterative retrievals to construct the graph. We want to find the probability $P(\xi_i)$ that image ξ_i is the best match to the query image. Assuming a multinomial distribution parameterized with $\mathbf{\Theta} = (\theta_1, \dots, \theta_m)$ where $p(\xi_i|\mathbf{\Theta}) = \theta_i$, the set $\mathcal{U}$ has the likelihood

$$P(\mathcal{U}|\mathbf{\Theta}) = \theta_1^{N_1} \cdots \theta_m^{N_m} \tag{5.4}$$

where $N_i, i = 1, \dots, m$ is the number of times image ξ_i appears in $\mathcal{U}$. The maximum likelihood estimate of θ_i can be found as

$$\hat{\theta}_i = \frac{N_i}{N} \tag{5.5}$$

where $N = \sum_{i=1}^{m} N_i$. However, this may give extreme results especially when the set $\mathcal{U}$ is small. We can improve this by assigning prior probabilities to $\mathbf{\Theta}$ and compute the Bayes estimate.

Let $\mathcal{V}$ be the set of images (the final near-clique) retrieved as the best matches to the query image as the result of graph-theoretic clustering. Let k be the size of $\mathcal{V}$. Without loss of generality, assume that the first k images $\xi_1, \ldots, \xi_k$ are in $\mathcal{V}$. Under the graph-theoretic retrieval model, we assume that all images in $\mathcal{V}$ are *a priori* twice as likely to be the best match to the query than the other $m-k$ images that are not in $\mathcal{V}$ (heuristic assumption). This gives the Dirichlet prior distribution

$$P(\mathbf{\Theta}|\mathcal{V}) = \text{Dir}(\mathbf{\Theta}|\underbrace{2,\ldots,2}_{k},\underbrace{1,\ldots,1}_{m-k}). \tag{5.6}$$

Then, the posterior distribution of $\mathbf{\Theta}$ can be found as

$$\begin{aligned} P(\mathbf{\Theta}|\mathcal{U},\mathcal{V}) &= \frac{P(\mathcal{U}|\mathbf{\Theta})P(\mathbf{\Theta}|\mathcal{V})}{P(\mathcal{U}|\mathcal{V})} \\ &= \text{Dir}(\mathbf{\Theta}|2+N_i, i \in \mathcal{V},\ 1+N_i, i \notin \mathcal{V}). \end{aligned} \tag{5.7}$$

The Bayes estimate of θ_i becomes

$$\hat{\theta}_i = \begin{cases} \frac{2+N_i}{m+k+N} & \text{if } i \in \mathcal{V} \\ \frac{1+N_i}{m+k+N} & \text{if } i \notin \mathcal{V}. \end{cases} \tag{5.8}$$

We can then select a threshold $\bar{\theta}$ and classify each image into either the relevance or the irrelevance class as

$$\text{assign } \xi_i \text{ to} \quad \begin{cases} \text{class } \mathcal{A} & \text{if } \theta_i > \bar{\theta} \\ \text{class } \mathcal{B} & \text{otherwise.} \end{cases} \tag{5.9}$$

5.5 Clustering Using Vector Quantization

An alternative clustering method is vector quantization which allows clustering in a high dimensional space. We use the Generalized Lloyd Algorithm (GLA) to cluster the feature space that is formed by treating each image as a point which is represented by its feature vector.

5.5.1 Generalized Lloyd Algorithm (GLA)

The Generalized Lloyd Algorithm [132, 81] can be summarized as follows:

1. Given the number of clusters n and the training set $\mathcal{T}$, begin with an initial codebook $\mathcal{C}_1$. Set the iteration number $r = 1$.

2. Given the codebook $\mathcal{C}_r = \{\mathbf{y}_{\boldsymbol{i}},\ i = 1, \ldots, n\}$, where $\mathbf{y}_{\boldsymbol{i}} \in \mathbb{R}^{(q\times 1)}$ is the centroid of the i'th cluster, partition the training set into cluster sets $\mathcal{R}_i$ using the nearest neighbor condition:

$$\mathcal{R}_i = \{\mathbf{x} \mid \mathbf{x} \in \mathcal{T},\ \rho(\mathbf{x}, \mathbf{y}_{\boldsymbol{i}}) \leq \rho(\mathbf{x}, \mathbf{y}_{\boldsymbol{j}}),\ \forall j \neq i\} \tag{5.10}$$

   where $\rho(\mathbf{x}, \mathbf{y})$ is the squared error between the vectors $\mathbf{x}$ and $\mathbf{y}$. When there is a tie, $\mathbf{x}$ is assigned to the cluster with smaller index.

3. Using the centroid condition, compute the centroids for the cluster sets found to obtain the new codebook $\mathcal{C}_{r+1}$ as

$$\mathcal{C}_{r+1} = \left\{ \mathbf{y}_{\boldsymbol{i}} = \frac{1}{\#\mathcal{R}_i} \sum_{\mathbf{x}\in\mathcal{R}_i} \mathbf{x},\ i = 1, \ldots, n \right\}. \tag{5.11}$$

4. Compute the average distortion for $\mathcal{C}_{r+1}$. If it has changed by an amount that is small enough relative to the last iteration, stop. Otherwise, increment r and go to Step 2.

Sets of images that are assigned to each cluster correspond to the vectors that are assigned to those clusters in the last iteration.

5.5.2 Codebook Initialization

The initial codebook for the Generalized Lloyd Algorithm can be determined by:

1. Random initialization, where each codebook vector is randomly drawn from a uniform distribution with a range same as the range of the feature space.

2. Splitting algorithm of Linde *et al.* [132].

3. The Pairwise Nearest Neighbor Algorithm (PNN) [81]:

   (a) Given the number of clusters n and the training set $\mathcal{T}$, begin with creating one cluster for each vector.

   (b) Merge clusters $\mathcal{R}_i$ and $\mathcal{R}_j$ which introduce the smallest error, i.e. minimize

   $$\epsilon'_{ij} - \epsilon_{ij} = \frac{\#\mathcal{R}_i \, \#\mathcal{R}_j}{\#\mathcal{R}_i + \#\mathcal{R}_j} \left(\mathbf{y}_{\boldsymbol{i}} - \mathbf{y}_{\boldsymbol{j}}\right)^2 . \tag{5.12}$$

   If there is a tie in the smallest error, merge the clusters which will result in a cluster with smaller number of elements. The centroid of the new cluster becomes

   $$\mathbf{y}_{\boldsymbol{i \cup j}} = \frac{1}{\#\mathcal{R}_i + \#\mathcal{R}_j} \left(\#\mathcal{R}_i \, \mathbf{y}_{\boldsymbol{i}} + \#\mathcal{R}_j \, \mathbf{y}_{\boldsymbol{j}}\right) . \tag{5.13}$$

   (c) Repeat Step 3b until n clusters remain.

Jain and Dubes [105] divided clustering algorithms into groups according to their properties like exclusive versus non-exclusive, intrinsic versus extrinsic, hierarchical versus partitional, agglomerative versus divisive, serial versus simultaneous, monothetic versus polythetic and graph-theoretic versus algebra-based. Both GLA and PNN are exclusive because no overlap between clusters is allowed, intrinsic because they are unsupervised, polythetic because the objects to be clustered are represented as points in a space, algebra-based because they use algebraic criteria like the mean squared error. On the other hand, PNN is an agglomerative hierarchical algorithm because it involves nested sequence of partitions starting from atomic clusters that are gradually merged into larger clusters, whereas GLA is partitional. PNN is serial because it handles clusters two by two at each iteration, whereas GLA is simultaneous.

5.6 Cluster Evaluation

To evaluate whether a cluster formed by a clustering algorithm is visually consistent or not, we define

$$Consistency = \frac{1}{k}\sum_{j=1}^{k}\frac{\#\{i \mid GT(i) = GT(j), i = 1, \ldots, k\}}{k} \tag{5.14}$$

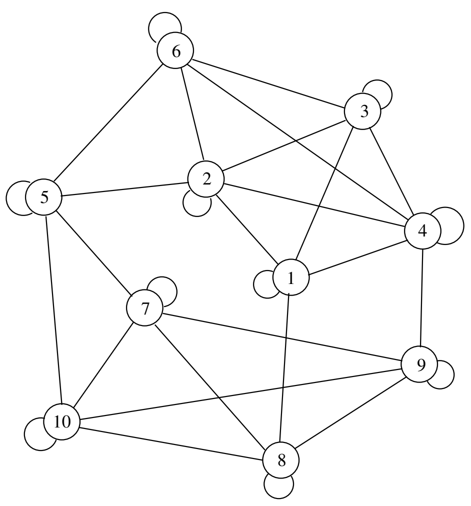
where k is the number of images in the cluster and $GT(i)$ is the groundtruth group to which image i of that cluster belongs. j indexes the images in the cluster. The term inside the summation indicates the percentage of the cluster that image j is correctly associated with. The overall effectiveness of a clustering algorithm for a given number of clusters can be measured using consistency averaged over all clusters.

In the following sections, we will compare graph-theoretic clustering and vector quantization in terms of clustering effectiveness.

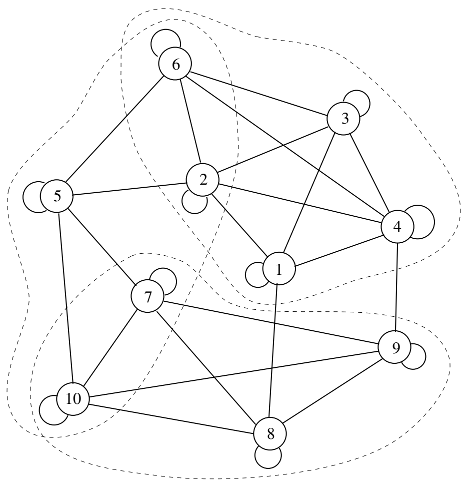
5.7 Experiments

5.7.1 Clustering Experiments

The first step of testing the proposed retrieval algorithm is to check whether the clusters formed by the graph-theoretic clustering algorithm are visually consistent or not. First, each image is used as a query, and for each search, n top-ranked images are retrieved. Then, a graph for the whole database is constructed with all images as nodes and n edges corresponding to the n top-ranked images for each node. Some possible clusters for the example database of Figure 5.1 are given in Figure 5.2. Note that the resulting clusters can overlap. This is a desired property because image



(a) Graph for the whole database



(b) Example clusters are marked by dashed lines

Figure 5.2: Example clusters of the graph when all images in Figure 5.1 are used as queries.

content is too complex to be grouped into distinct categories. Hence, an image can be consistent with multiple groups of images.

MINOVERLAP was fixed to be 0.80 in all experiments. 2256 clustering tests were performed for various values of $n \in [10, 100]$, MINCOMP $\in [0.3, 1.0]$, MINASSOC $\in [0, \text{MINCOMP}]$ and MINSIZE $\in \{8, 12\}$ for different feature vector and similarity model combinations. Parameters that resulted in the largest average consistency values for example cases are given in Table 5.1. Example clusters using these parameters are given in Figure 5.3. Gabor and color histogram feature vectors and the multivariate Gaussian model resulted in larger consistency because the initial retrievals that were used to construct the graphs were more successful for these models. We also obtained larger consistency values than the ones given in Table 5.1 when we allowed some images to remain unclustered. We observed that decreasing n or increasing MINCOMP or MINASSOC increased consistency but also increased the number of

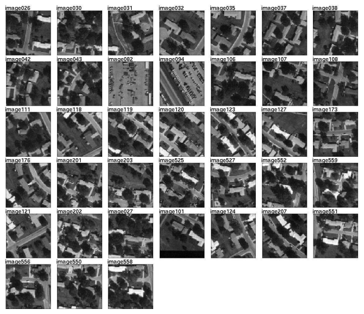
Table 5.1: Parameters of the graph-theoretic clustering algorithm that resulted in the largest average consistency values for example cases. n, MINASSOC, MINCOMP, MINSIZE and MINOVERLAP (0.80) are input, and consistency and number of clusters are output.

Database	Feature	n	MINCOMP	MINASSOC	MINSIZE	Consistency	# Clusters
ISL	LAR+COOC	36	0.7	0.6	12	0.9339	87
ISL	GABOR	40	0.7	0.6	12	0.9739	79
VisTex	GABOR	14	0.7	0.5	8	0.9017	60

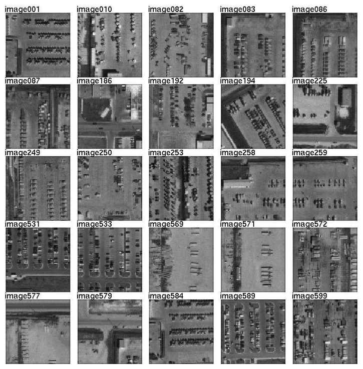
unclustered images.

We also clustered the feature space using the Generalized Lloyd Algorithm (GLA) (also known as the k-means algorithm). The threshold for the percentage decrease in distortion to abort the iterations was selected to be 0.0001. The number of clusters was set to change from 10 to 80 with increments of 5. The initial codebook generated by the Pairwise Nearest Neighbor Algorithm (PNN) resulted in the smallest distortion so only those results are presented here. For all of the algorithms except the GLA using random initialization, distortion decreased with increasing number of clusters. Random initialization sometimes caused an empty cell problem which increased the distortion (the implementation did not take any steps to prevent empty cells because they occurred only after a random initialization).

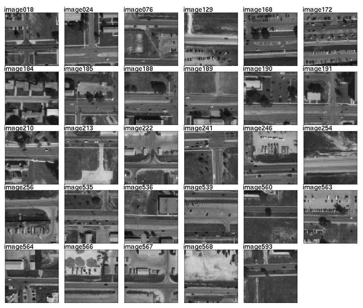
Consistency values for given numbers of clusters for example cases are given in Figure 5.4. Using GLA with PNN had a larger average consistency so we can say that pairwise merging was effective in grouping visually similar images in the feature space. Some example clusters are given in Figure 5.5. Although consistency values for most of the clusters were large, some clusters with small consistency values (around 0.3) decreased the average dramatically. It means that the features could not map some of the visually similar images to nearby locations in the feature space. Another reason may be that the squared error criterion was sometimes reported to have a poor correlation with the human visual system. The largest average consistency



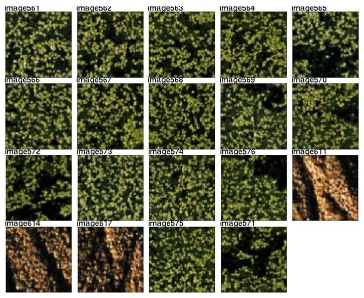
(a) LAR+COOC, Consistency = 0.900277



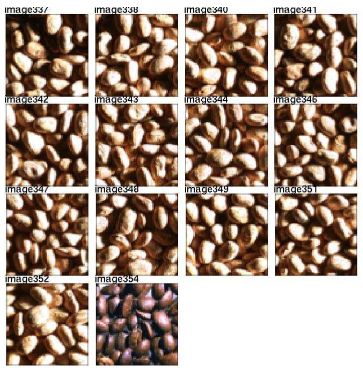
(b) GABOR, Consistency = 0.731200



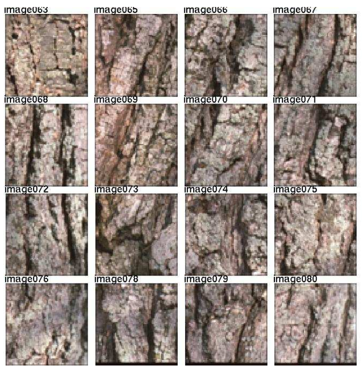
(c) GABOR, Consistency = 0.933413



(d) GABOR, Consistency = 0.734072



(e) GABOR, Consistency = 0.867347



(f) GABOR, Consistency = 0.882812

Figure 5.3: Example clusters obtained with graph-theoretic clustering using the parameters given in Table 5.1. Even though consistency values can be low, clusters contain visually similar images.

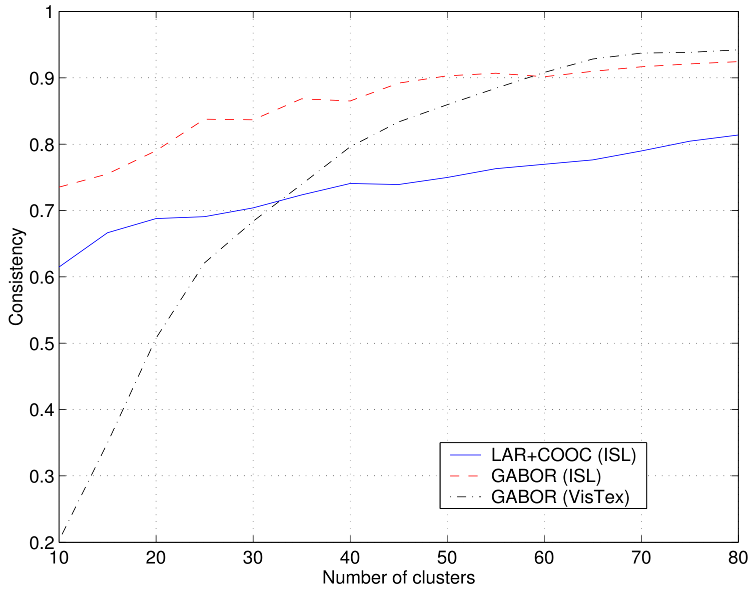
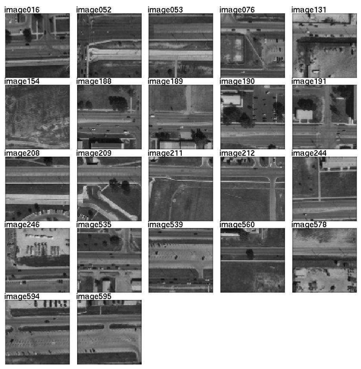


Figure 5.4: Consistency values (y-axis) as a function of number of clusters (x-axis) using the Generalized Lloyd and Pairwise Nearest Neighbor Algorithms for the example feature vectors and databases given in Table 5.1.

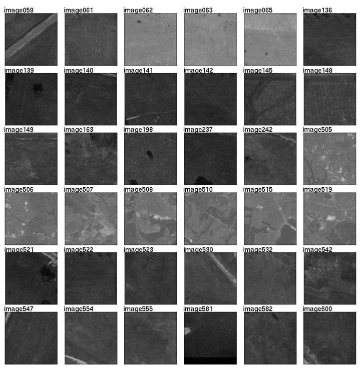
values obtained using the graph-theoretic clustering algorithm were usually greater than the ones obtained using the GLA and PNN algorithms for the same number of clusters. The graph-theoretic clustering algorithm requires the parameters n, MINASSOC, MINCOMP, MINSIZE and MINOVERLAP to be given and then it determines the number of clusters automatically, while the number clusters is the only required parameter in the GLA algorithm.

5.7.2 Retrieval Experiments

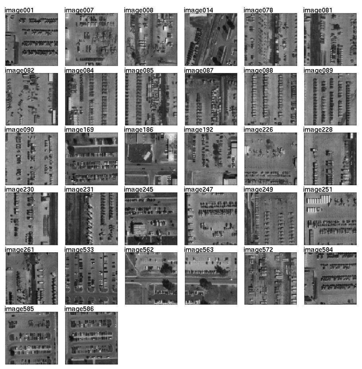
Results of the clustering experiments of the previous section were used to select the best parameter set to be used in retrieval. The cluster with the largest compactness (and with the largest number of nodes if there was a tie) was retrieved as the best match for each query. Then, the model described in Section 5.4 was used to estimate the probability of each image being relevant to the query image under the



(a) LAR+COOC, Consistency = 0.913223

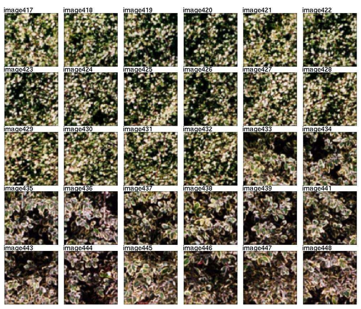


(b) GABOR, Consistency = 0.686728

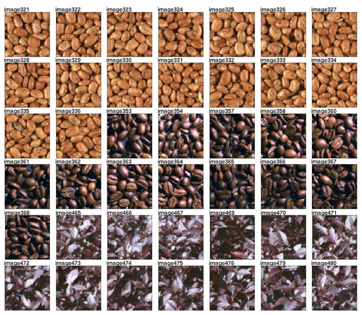


(c) GABOR, Consistency = 0.781250

(d) GABOR, Consistency = 0.500000



(e) GABOR, Consistency = 0.502222



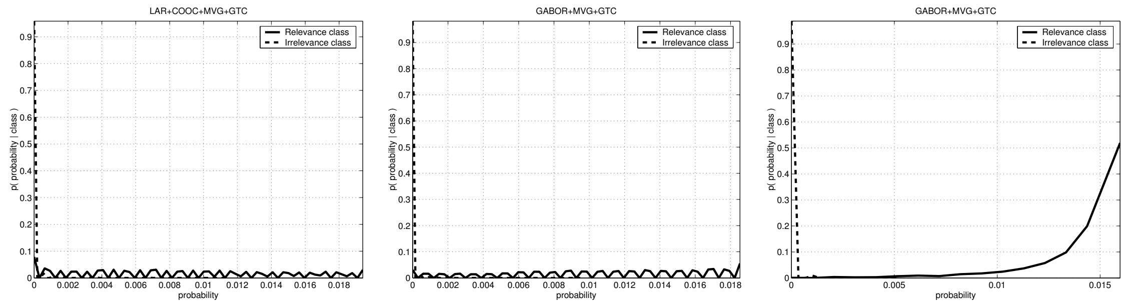
(f) GABOR, Consistency = 0.336735

Figure 5.5: Example clusters obtained with vector quantization. Even though consistency values can be low, clusters contain visually similar images.

graph-theoretic framework, and these estimates were used to rank the images in the database. The results were compared to the ones in Chapter 4 which did not use clustering. When n is set to be equal to the number of images in the database, the graph reduces to a single clique and the graph-theoretic search becomes equivalent to retrieving n top-ranked images.

Figure 5.6 shows histograms of the probabilities for the images in the training sets for the relevance and irrelevance classes. Example queries without and with clustering are given in Figure 5.7. We can see that some images that were visually irrelevant to the query image could be eliminated after graph-theoretic clustering. This is consistent with Figure 5.8 where average precision and recall for the example cases of Table 5.1 are given. The precision-recall curves after clustering were initially above the curves for retrieval without clustering but they intersected when the number of retrieved images was around 90 for the ISL Database and around 50 for the VisTex Database. These numbers are larger than the average number of images in each groundtruth group. Since the model described in Section 5.4 uses data from the clusters and a prior distribution to estimate the probabilities, the prior information becomes dominant when the number of retrieved images go above the average number of images for each groundtruth group, and all the remaining images have the same probability given by the prior. This is not a problem because the images that we are interested in are already retrieved (recall is already high). The average precision when 12 images were retrieved after clustering was 0.9538, 0.9785 and 0.9612 for the ISL Database using LAR+COOC feature vectors and using GABOR feature vectors, and the VisTex Database using GABOR feature vectors respectively. On the other hand, the average precision when only the 12 top-ranked images were retrieved without clustering was 0.9438, 0.9673 and 0.9480 for the same settings.

Since we use the top-ranked images to construct the graph, the initial precision before clustering should be large enough to prevent the graph being dominated by images visually irrelevant to the query image. In our experiments in [6], a significant



(a) ISL, LAR+COOC (b) ISL, GABOR (c) VisTex, GABOR

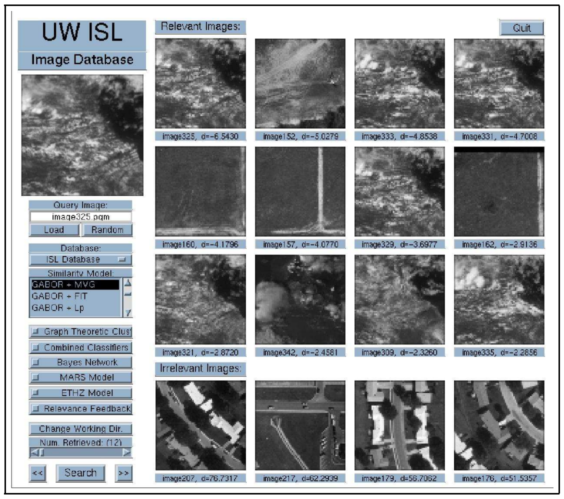
Figure 5.6: Histograms of probabilities for the images in the training sets for the relevance and irrelevance classes under the graph-theoretic clustering model using the parameters given in Table 5.1.

improvement in precision was observed when the initial precision was greater than 0.5 compared to the case when the initial precision was less than 0.5.

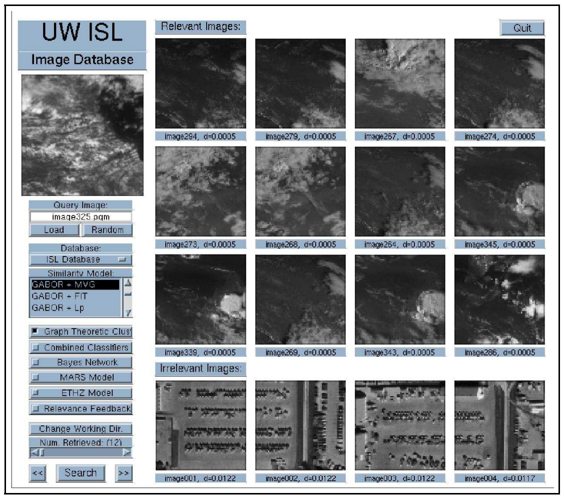
5.7.3 Query Visualization Using Feature Space Projections

In this section we illustrate how clustering modifies the retrieval set iteratively. The data used was a subset of the images in the ISL Database (340 images). Figure 5.9 shows the feature space for the line-angle-ratio and co-occurrence feature vectors. The feature vectors were projected onto their first two principal components to reduce the dimensionality and Sammon's nonlinear mapping algorithm [113] was used to refine the projections.

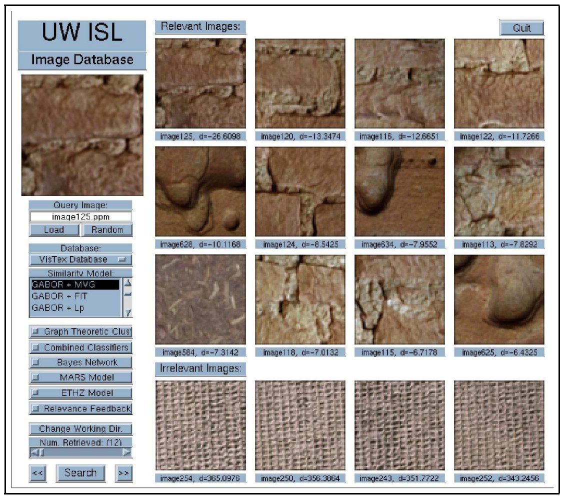
Figures 5.10 - 5.13 show some retrieval examples. Euclidean distance was used for similarity between feature vectors for illustration. As can be seen from these figures, the projection algorithms did not generate a perfect mapping because the nearest neighbors in the original space were not necessarily nearest neighbors when they were projected to 2-D. For images that belonged to the groups that were well separated from other groups in the feature space, the graph-theoretic clustering process often



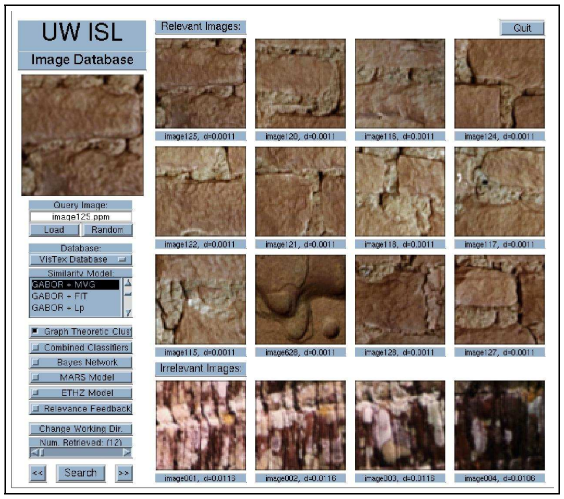
(a) An example query for LANDSAT USA using Gabor features without clustering (8/12)



(b) Same query for LANDSAT USA using Gabor features with clustering (12/12)

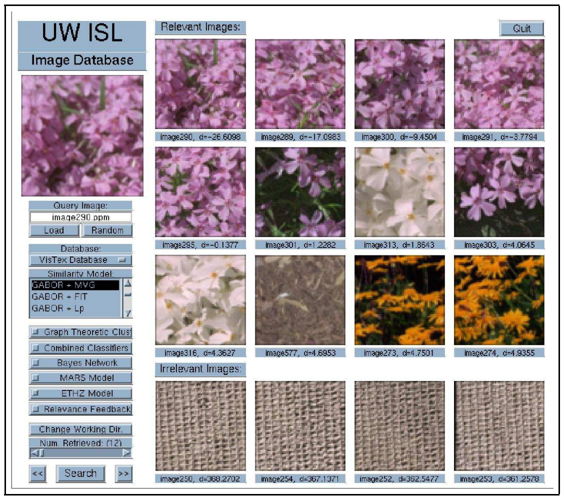


(c) An example query for brick using Gabor features without clustering (8/12)

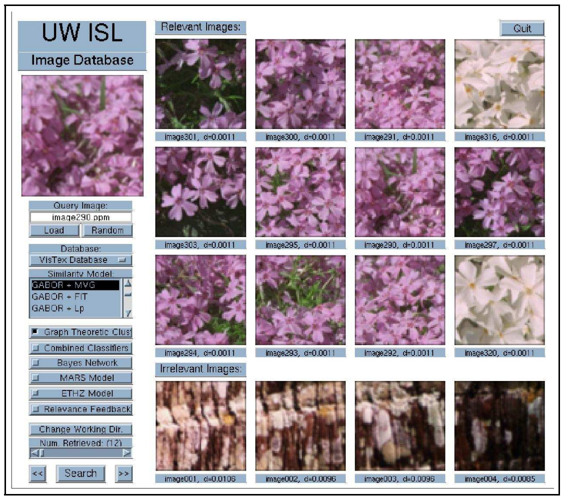


(d) Same query for brick using Gabor features with clustering (11/12)

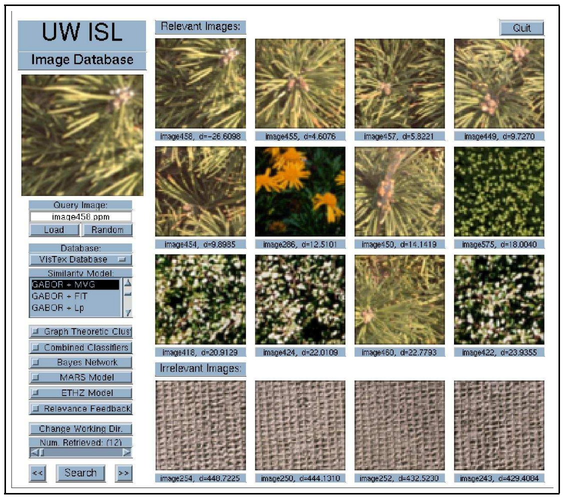
Figure 5.7: Example queries without (left) and with (right) graph-theoretic clustering. The numbers in parentheses in sub-captions show the number of correct matches for each case.



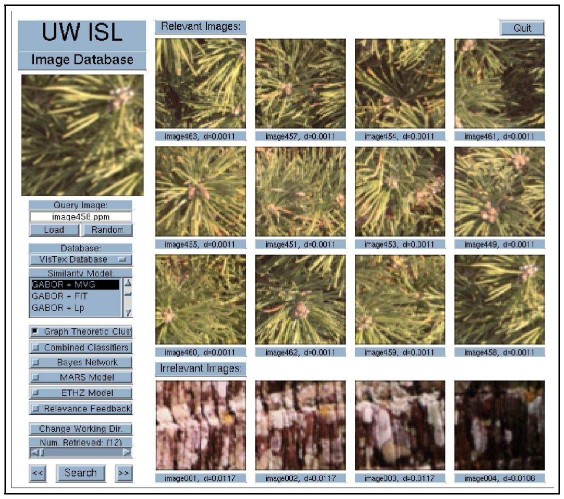
(a) An example query for flowers using Gabor features without clustering (7/12)



(b) Same query for flowers using Gabor features with clustering (10/12)

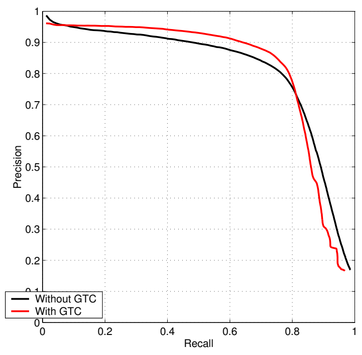


(c) An example query for leaves using Gabor features without clustering (7/12)

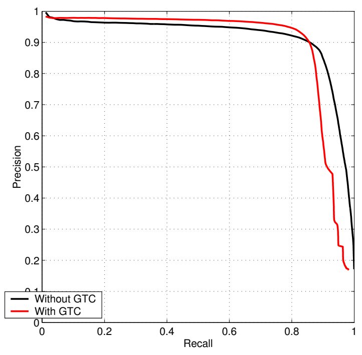


(d) Same query for leaves using Gabor features with clustering (12/12)

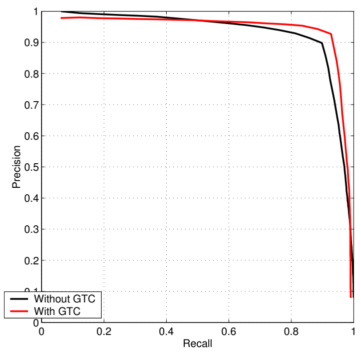
Figure 5.7: (continued)



(e) ISL, LAR+COOC

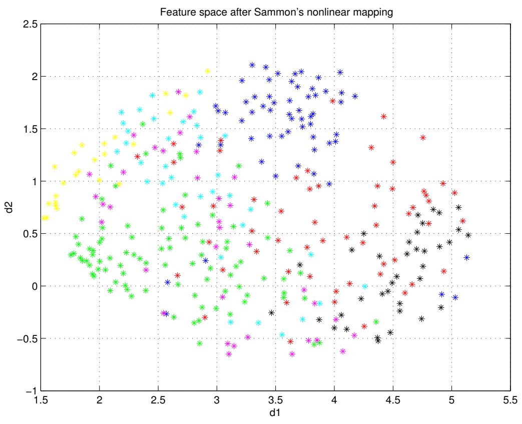


(f) ISL, GABOR

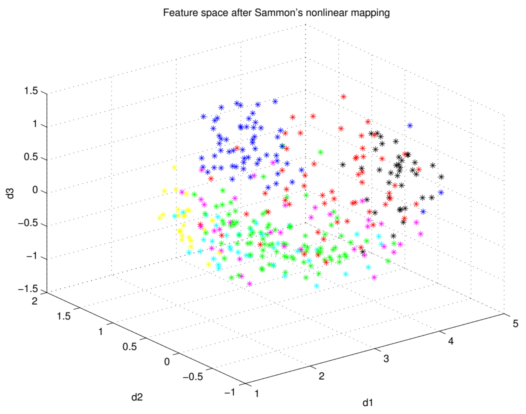


(g) VisTex, GABOR

Figure 5.8: Retrieval performance without (black) and with (red) graph-theoretic clustering using the parameters given in Table 5.1. Each plot shows average precision (y-axis) vs. recall (x-axis) for the example cases.

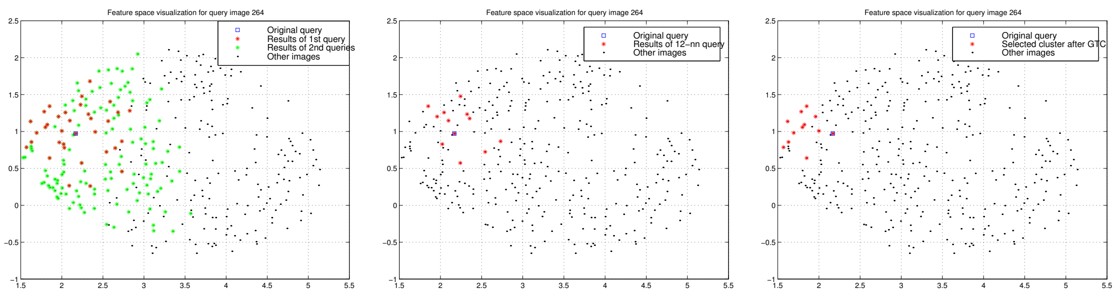


(a) Projections to 2-D using Sammon's algorithm



(b) Projections to 3-D using Sammon's algorithm

Figure 5.9: Feature space projections for 340 images from the ISL Database. The groundtruth groups are: parking lots (black), roads (red), residential areas (blue), landscapes (green), LANDSAT USA (magenta), DMSP North Pole (cyan) and LANDSAT Chernobyl (yellow). These plots show how the feature space was structured compared to the manually generated groundtruth.



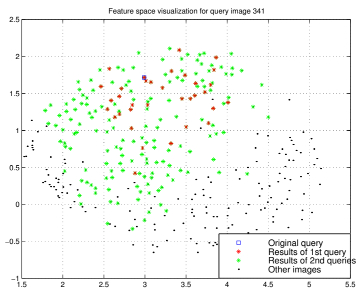
(a) Whole query process

(b) Results without clustering

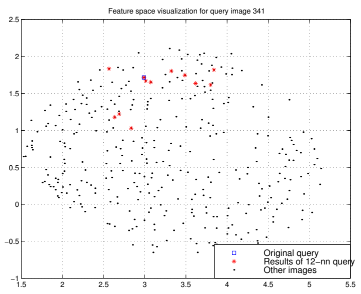
(c) Results with clustering

Figure 5.10: An example for successful clustering where precision increased from 4/12 to 11/12 after clustering. Different colored points and the legends represent the original query image (original query), images that are retrieved as a result of the first level query by the original query image (results of 1st query (or results of 12-nn query)), images that are retrieved as a result of the second level queries when the results of the first level query are used as query images (results of 2nd queries), images in the resulting cluster after graph-theoretic clustering (selected cluster after GTC), and other images in the database (other images).

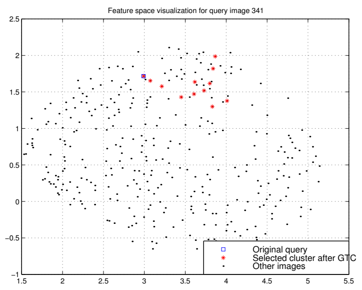
removed the outliers; on the other hand, if the query image appeared at a location where there was not a good separation between groups, one of the groups dominated in clustering and if that group was not the one that the query image actually belonged, a poor precision was obtained. Therefore, improving the features makes graph-theoretic clustering more effective. This is also another example where a nearest neighbor search using a distance measure like the Euclidean distance is too sensitive to the cluster structures in the feature space while the likelihood-based measures make use of the offline training and are more robust.



(a) Whole query process

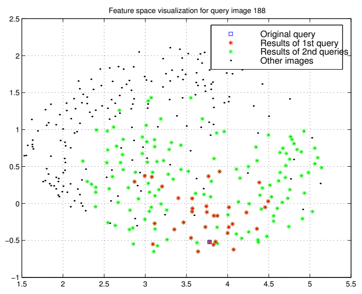


(b) Results without clustering

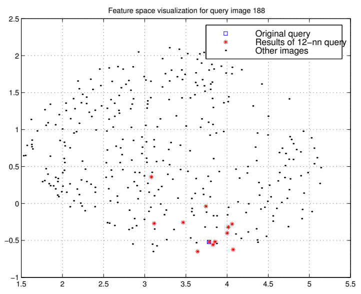


(c) Results with clustering

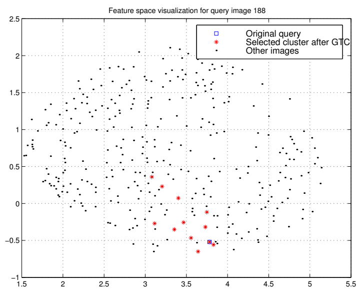
Figure 5.11: An example for successful clustering where precision increased from 8/12 to 12/12 after clustering.



(a) Whole query process

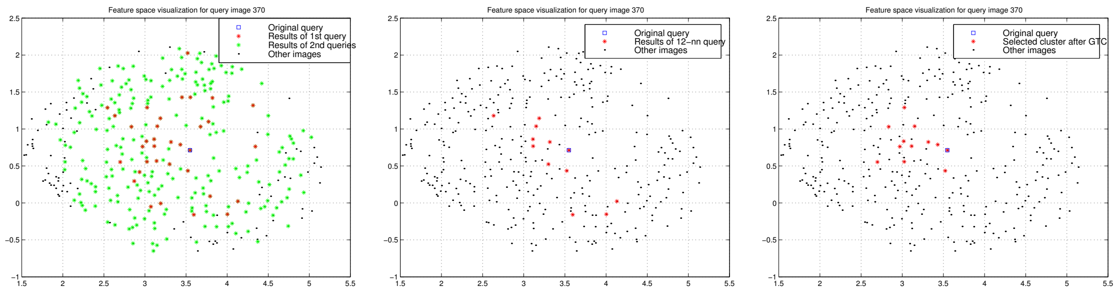


(b) Results without clustering



(c) Results with clustering

Figure 5.12: An example for unsuccessful clustering where precision decreased from 6/12 to 2/12 after clustering.



(a) Whole query process

(b) Results without clustering

(c) Results with clustering

Figure 5.13: An example for unsuccessful clustering where precision decreased from 8/12 to 5/12 after clustering.

Chapter 6

A WEIGHTED DISTANCE APPROACH TO RELEVANCE FEEDBACK

6.1 Introduction

As discussed in Chapters 4 and 5, none of the existing feature extraction algorithms can always map visually similar images to nearby locations in the feature space and post-processing methods can be useful to improve the results. Relevance feedback was shown to be useful in document retrieval [175]. It has also recently been popular in the image retrieval community as reviewed in Chapter 2. In this chapter, we propose a weighted distance approach for relevance feedback.

When the commonly used geometric framework of the nearest neighbor rule is used as the similarity measure, the straightforward way of incorporating feedback is to weight individual feature components according to user's responses. The disadvantages of most of the methods proposed for relevance feedback are that either they require keywords to be manually assigned to images, or they do weight assignment heuristically, or they are based on unrealistic assumptions. However, retrieval algorithms depend on features directly computed from images. We want to use only the information fed back by the user instead of using artificial keywords or unjustified heuristic assumptions. We also cannot assume anything about the user's information need, neither can we assume any rules for the relevancy and irrelevancy he/she is looking for. Therefore, after formulating the weight updating problem in an estimation and regression framework, we compute the optimum weights that can be used for iterative retrieval.

Average precision is computed for the groundtruthed data set for performance evaluation. We also define a new measure called progress to measure the performance. Using experiments on a small subset of the ISL Database, we show that the weighted distance approach can improve precision as much as 19% over the case without feedback. The following section describes the motivation and details of our weighted distance approach.

6.2 The Weighted Distance Approach

First, we present some definitions that will be used in the following sections.

K**:** Number of iterative searches.

$\mathcal{R}^{(k)}=$ {retrieval set after the k'th search}, $k = 0, \ldots, K$, while $\mathcal{R}^{(0)}$ being the whole database.

$\mathcal{R}^{(k)}_{rel}=$ {set of images in $\mathcal{R}^{(k)}$ that are marked as relevant}, $\mathcal{R}^{(k)}_{rel} \subseteq \mathcal{R}^{(k)}$.

$\mathcal{R}^{(k)}_{irrel}=$ {set of images in $\mathcal{R}^{(k)}$ that are marked as irrelevant}, $\mathcal{R}^{(k)}_{irrel} \subseteq \mathcal{R}^{(k)}$, $\mathcal{R}^{(k)}_{rel} \cup \mathcal{R}^{(k)}_{irrel} \subseteq \mathcal{R}^{(k)}$.

$\mathcal{F}^{(k)}_{i}=$ {values of the i'th feature components of the images in $\mathcal{R}^{(k)}$}.

$\mathcal{F}^{(k)}_{rel,i}=$ {values of the i'th feature components of the images in $\mathcal{R}^{(k)}_{rel}$}.

$\mathcal{F}^{(k)}_{irrel,i}=$ {values of the i'th feature components of the images in $\mathcal{R}^{(k)}_{irrel}$}.

In the retrieval scenario of this chapter, similarity between images is measured by computing distances between feature vectors in the feature space. The inputs to the distance measures are two q-dimensional feature vectors $\mathbf{x}$ and $\mathbf{y}$ and the weight vector $\mathbf{w} \in \mathbb{R}^{(q\times 1)}$. We use the weighted Minkowsky L_p distance as defined below.

Definition 3 *Weighted Minkowsky L_p distance*

$$\rho_p(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \left(\sum_{i=1}^{q} |\mathbf{w}_i(\mathbf{x}_i - \mathbf{y}_i)|^p \right)^{1/p} \tag{6.1}$$

where $\mathbf{x}, \mathbf{y}, \mathbf{w} \in \mathbb{R}^{(q\times 1)}$ and $\mathbf{x}_i, \mathbf{y}_i, \mathbf{w}_i$ are the i'th components of the vectors $\mathbf{x}, \mathbf{y}, \mathbf{w}$ respectively.

i) Positivity: $\rho_p(\mathbf{x}, \mathbf{y}; \mathbf{w}) \geq 0$ and if $\mathbf{x} = \mathbf{y}$, then $\rho_p(\mathbf{x}, \mathbf{y}; \mathbf{w}) = 0$.

ii) Strictly positivity: If $\rho_p(\mathbf{x}, \mathbf{y}; \mathbf{w}) = 0$, $|\mathbf{w}_i(\mathbf{x}_i - \mathbf{y}_i)| = 0\ \forall i$. If $\mathbf{w}_i > 0$, this implies that $\mathbf{x}_i = \mathbf{y}_i\ \forall i$. However, $\mathbf{w}_i \geq 0$ in our case so it does not guarantee that $\mathbf{x}_i = \mathbf{y}_i\ \forall i$. Not satisfied.

iii) Symmetry: $\rho_p(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \rho_p(\mathbf{y}, \mathbf{x}; \mathbf{w})$.

iv) Triangle inequality:

$$\begin{aligned}
\rho_p(\mathbf{x}, \mathbf{z}; \mathbf{w}) &= \left(\sum_{i=1}^{q} |\mathbf{w}_i(\mathbf{x}_i - \mathbf{z}_i)|^p \right)^{1/p} \\
&= \left(\sum_{i=1}^{q} |\mathbf{w}_i\mathbf{x}_i - \mathbf{w}_i\mathbf{z}_i|^p \right)^{1/p} \\
&= \left(\sum_{i=1}^{q} |\mathbf{w}_i\mathbf{x}_i - \mathbf{w}_i\mathbf{y}_i + \mathbf{w}_i\mathbf{y}_i - \mathbf{w}_i\mathbf{z}_i|^p \right)^{1/p} \\
&\leq \left(\sum_{i=1}^{q} |\mathbf{w}_i\mathbf{x}_i - \mathbf{w}_i\mathbf{y}_i|^p \right)^{1/p} + \left(\sum_{i=1}^{q} |\mathbf{w}_i\mathbf{y}_i - \mathbf{w}_i\mathbf{z}_i|^p \right)^{1/p} \\
&\qquad \text{(Using Minkowsky inequality [151])} \\
&= \left(\sum_{i=1}^{q} |\mathbf{w}_i(\mathbf{x}_i - \mathbf{y}_i)|^p \right)^{1/p} + \left(\sum_{i=1}^{q} |\mathbf{w}_i(\mathbf{y}_i - \mathbf{z}_i)|^p \right)^{1/p} \\
&= \rho_p(\mathbf{x}, \mathbf{y}; \mathbf{w}) + \rho_p(\mathbf{y}, \mathbf{z}; \mathbf{w})
\end{aligned}$$

Since the strictly positivity requirement is not satisfied, the weighted L_p distance is a pseudo–metric. We describe the methods for weight updating in the following sections.

6.3 Standard Deviation Ratio-based Weight Updating

From the pattern recognition point of view, for a feature to be good,

- its variance among all the images in the database should be large,
- its variance among the relevant images should be small,
- its variance among the irrelevant images should be large.

Any one of these is not enough alone but characterizes a good feature when combined with the others.

Given these observations, we decided to use $\mathbf{w}_i^{(k)} = \sigma_i^{(0)}/\sigma_{rel,i}^{(k)}$, where $\sigma_i^{(0)} = \text{std}(\mathcal{F}_i^{(0)})$ and $\sigma_{rel,i}^{(k)} = \text{std}(\mathcal{F}_{rel,i}^{(k)})$, as the weight for the i'th feature in the $k+1$'st iteration. For a given image, there is a small set of relevant images in the database; on the other hand, the rest of the images can be categorized as irrelevant. We preferred using only the relevant images because the small set of feedback images that are selected by the user for both relevancy and irrelevancy will probably provide a better estimate for the former case.

Depending on $\sigma_i^{(0)}$ and $\sigma_{rel,i}^{(k)}$, four different situations can arise as shown in Table 6.1:

- When $\sigma_i^{(0)}$ is large and $\sigma_{rel,i}^{(k)}$ is small, $\mathbf{w}_i^{(k)}$ becomes large. This means that the feature has a diverse set of values in the database but its values for relevant images are similar. This is a desired situation and shows that this feature is very effective in distinguishing this specific relevant image set so a large weight assigns more importance to this feature.

Table 6.1: Motivation for the weight selection. Moving upwards in the table represents a situation that is closer to ideal.

$\sigma_i^{(0)}$	$\sigma_{rel,i}^{(k)}$	$\mathbf{w}_i^{(k)} = \sigma_i^{(0)} / \sigma_{rel,i}^{(k)}$
large	small	large
large	large	~ 1
small	small	~ 1
small	large	small

- When both $\sigma_i^{(0)}$ and $\sigma_{rel,i}^{(k)}$ are large, $\mathbf{w}_i^{(k)}$ is close to 1. This means that the feature may have good discrimination characteristics in the database but is not effective for this specific relevant image group. The resulting weight does not give any particular importance to this feature.

- When both $\sigma_i^{(0)}$ and $\sigma_{rel,i}^{(k)}$ are small, $\mathbf{w}_i^{(k)}$ is again close to 1. This is a similar but slightly worse situation than the previous one. The feature is not generally effective in the database and is not effective for this relevant set either. No importance is given to this feature.

- When $\sigma_i^{(0)}$ is small and $\sigma_{rel,i}^{(k)}$ is large, $\mathbf{w}_i^{(k)}$ becomes small. This is the worst case among all the possibilities. The feature is not generally effective and even causes the distance between relevant images to increase. A small weight forces the distance measure to ignore the effect of this feature.

All of the resulting weights in these four cases are consistent with the desired situations in an ideal retrieval. Note that at least one image other than the query image should be labeled as relevant by the user to have a non-zero standard deviation. This is a reasonable assumption because the user will create a new query if nothing relevant was retrieved.

6.3.1 Iterative Retrieval

The retrieval algorithm can be described as follows:

1. Initialize all weights uniformly as $\mathbf{w}_i^{(0)} = 1/q$, $i = 1, \ldots, q$. Compute $\sigma_i^{(0)}$, $i = 1, \ldots, q$.

2. For $k = 1, \ldots, K$,

   (a) Search the database using $\mathbf{w}_i^{(k-1)}$ and obtain the retrieval set $\mathcal{R}^{(k)}$.

   (b) Get feedback from the user as $\mathcal{R}_{rel}^{(k)}$.

   (c) Compute $\sigma_{rel,i}^{(k)}, i = 1, \ldots, q$.

   (d) Compute
   $$\mathbf{w}_i^{(k)} = \frac{\sigma_i^{(0)}}{\sigma_{rel,i}^{(k)}}, i = 1, \ldots, q \tag{6.2}$$
   and normalize as $\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k)} / \sum_{i=1}^{q} \mathbf{w}_i^{(k)}$.

3. Do the final search using $\mathbf{w}_i^{(K)}, i = 1, \ldots, q$.

To compute $\sigma_{rel,i}^{(k)}$ in 2c, we use two methods:

- Independent update: Standard deviations are estimated independently in every iteration using only that iteration's retrieval sets, i.e.
  $$(\sigma_{rel,i}^{(k)})^2 = E[(\mathcal{F}_{rel,i}^{(k)})^2] - (E[\mathcal{F}_{rel,i}^{(k)}])^2, i = 1, \ldots, q$$
  where $E[\mathcal{F}_{rel,i}^{(k)}]$ and $E[(\mathcal{F}_{rel,i}^{(k)})^2]$ are the first and second moments of the sample $\mathcal{F}_{rel,i}^{(k)}$ respectively.

- Incremental update: We assume that user's notion of similarity does not change as the iterations progress and he/she is consistent in consecutive iterations.

Therefore, standard deviations are incrementally updated in every iteration, i.e.

$$\begin{aligned}(\sigma_{rel,i}^{(k)})^2 = & \frac{1}{\#\mathcal{R}_{rel}^{(k-1)} + \#\mathcal{R}_{rel}^{(k)}} \left(\#\mathcal{R}_{rel}^{(k-1)}\, E[(\mathcal{F}_{rel,i}^{(k-1)})^2] + \#\mathcal{R}_{rel}^{(k)}\, E[(\mathcal{F}_{rel,i}^{(k)})^2] \right) - \\ & \left(\frac{1}{\#\mathcal{R}_{rel}^{(k-1)} + \#\mathcal{R}_{rel}^{(k)}} \left(\#\mathcal{R}_{rel}^{(k-1)}\, E[\mathcal{F}_{rel,i}^{(k-1)}] + \#\mathcal{R}_{rel}^{(k)}\, E[\mathcal{F}_{rel,i}^{(k)}] \right) \right)^2 ,\end{aligned}$$

i=1,...,q, where the retrieval sets are updated as $\mathcal{R}^{(k)} = \mathcal{R}^{(k)} \cup \mathcal{R}^{(k-1)}$ and $\mathcal{R}_{rel}^{(k)} = \mathcal{R}_{rel}^{(k)} \cup \mathcal{R}_{rel}^{(k-1)}$ after iteration k.

When all the values in $\mathcal{F}_{rel,i}^{(k)}$ are the same, i.e. all images have the same value for that feature, we assign a large constant value to $\mathbf{w}_i^{(k)}$.

6.3.2 Alternative Formulation

A special case of this weighting scheme can alternatively be formulated as follows. Let the feature vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_q)^T \in \mathbb{R}^{(q\times 1)}$ be a multivariate random variable with independent components and the covariance matrix

$$\mathbf{\Sigma}_{\boldsymbol{x}} = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \sigma_q^2 \end{pmatrix}. \tag{6.3}$$

After the whitening transform [78], the random variable $\mathbf{y} = \mathbf{\Sigma}_{\boldsymbol{x}}^{-1/2}\mathbf{x}$ has components with unit variance, i.e. $\mathbf{\Sigma}_{\boldsymbol{y}} = I$. Let $\mathbf{x}_{\boldsymbol{rel,1}}, \ldots, \mathbf{x}_{\boldsymbol{rel,N}}$ be a sample of feature vectors for relevant images. Let $\mathbf{\Sigma}_{\mathbf{x}_{rel}}$ be the sample covariance matrix

$$\mathbf{\Sigma}_{\mathbf{x}_{rel}} = \begin{pmatrix} \sigma_{rel,1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{rel,2}^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \sigma_{rel,q}^2 \end{pmatrix}. \tag{6.4}$$

After the same whitening transform, $\mathbf{y}_{\boldsymbol{rel}} = \mathbf{\Sigma}_{\boldsymbol{x}}^{1/2}\mathbf{x}_{\boldsymbol{rel}}$, the covariance matrix of $\mathbf{y}_{\boldsymbol{rel}}$ becomes

$$\begin{aligned}
\mathbf{\Sigma}_{\mathbf{y}_{rel}} &= \mathbf{\Sigma}_{\boldsymbol{x}}^{-1/2}\mathbf{\Sigma}_{\mathbf{x}_{rel}}(\mathbf{\Sigma}_{\boldsymbol{x}}^{-1/2})^T \\
&= \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \frac{1}{\sigma_q} \end{pmatrix} \begin{pmatrix} \sigma_{rel,1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{rel,2}^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \sigma_{rel,q}^2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \frac{1}{\sigma_q} \end{pmatrix}^T \\
&= \begin{pmatrix} \frac{\sigma_{rel,1}^2}{\sigma_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\sigma_{rel,2}^2}{\sigma_2^2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \frac{\sigma_{rel,q}^2}{\sigma_q^2} \end{pmatrix}.
\end{aligned} \tag{6.5}$$

The Mahalanobis distance under the hypothesis that two feature vectors $\mathbf{u}$ and $\mathbf{v}$ are relevant is

$$\rho(\mathbf{u},\mathbf{v}) = (\mathbf{u}-\mathbf{v})^T\mathbf{\Sigma}_{\mathbf{y}_{rel}}^{-1}(\mathbf{u}-\mathbf{v}) = \sum_{i=1}^{q}\frac{\sigma_i^2}{\sigma_{rel,i}^2}(\mathbf{u}_i-\mathbf{v}_i)^2 = \sum_{i=1}^{q}\mathbf{w}_i^2(\mathbf{u}_i-\mathbf{v}_i)^2. \tag{6.6}$$

This is equivalent to the weighted squared Euclidean distance with the weights $\mathbf{w}_i = \sigma_i/\sigma_{rel,i}$, which is a special case of Definition 3.

6.4 Regression-based Weight Updating

We can also formulate the weight selection process as a regression problem using training data. Given N image pairs with their feature vectors $(\mathbf{x_1},\mathbf{y_1}),\ldots,(\mathbf{x_N},\mathbf{y_N})$ and their labels $c_1,\ldots,c_N$ where

$$c_i = \begin{cases} 0 & \text{if pair } i \text{ is from the relevance class} \\ 1 & \text{if pair } i \text{ is from the irrelevance class,} \end{cases} \tag{6.7}$$

first, the differences $\mathbf{d_1}, \ldots, \mathbf{d_N}$ are computed as

$$\mathbf{d_i} = |\mathbf{x_i} - \mathbf{y_i}|, \quad \mathbf{x_i}, \mathbf{y_i}, \mathbf{d_i} \in \mathbb{R}^{(q \times 1)},\ i = 1, \ldots, N. \tag{6.8}$$

We can then choose the q-dimensional weight vector $\mathbf{w}$ to satisfy

$$\begin{aligned} \underbrace{(\mathbf{d_1}, \cdots, \mathbf{d_N})^T}\ \mathbf{w} &= \underbrace{(c_1, \cdots, c_N)^T}, \\ \mathbf{D} \qquad \mathbf{w} &= \qquad \mathbf{C}. \end{aligned} \tag{6.9}$$

The N image pairs to compute the weights can be formed by using the cross product $\mathcal{R}_{rel} \times \mathcal{R}_{rel}$ for the relevance class and $\mathcal{R}_{rel} \times \mathcal{R}_{irrel}$ for the irrelevance class, where

$$\begin{aligned} \mathcal{R}_{rel} &= \{\text{set of images that are marked as relevant by the user}\}, \\ \mathcal{R}_{irrel} &= \{\text{set of images that are marked as irrelevant by the user}\}. \end{aligned}$$

6.4.1 Linear Least-Squares with Singular Value Decomposition (SVD)

The least-squares solution to Equation (6.9) can be computed as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{D}\mathbf{w} - \mathbf{C}\|^2 \tag{6.10}$$

where $\mathbf{D} \in \mathbb{R}^{(N \times q)}$ is the matrix formed by feature difference vectors, $\mathbf{w} \in \mathbb{R}^{(q \times 1)}$ is the weight vector, and $\mathbf{C} \in \mathbb{R}^{(N \times 1)}$ is the class label vector. A straightforward solution for this regression problem is

$$\mathbf{w}^* = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{C}. \tag{6.11}$$

To avoid the problems that can be caused by the singularities in $\mathbf{D}^T\mathbf{D}$, a more stable solution can be found by replacing $\mathbf{D}$ by its Singular Value Decomposition [161] $\mathbf{D} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T\mathbf{w} - \mathbf{C}\|^2 \tag{6.12}$$

where $\mathbf{U} \in \mathbb{R}^{(N\times N)}$ and $\mathbf{V} \in \mathbb{R}^{(q\times q)}$ are orthonormal matrices, and $\mathbf{\Lambda} \in \mathbb{R}^{(N\times q)}$ is the matrix of singular values. Using the fact that $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$, the problem can be restated as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{\Lambda}\mathbf{V}^T\mathbf{w} - \mathbf{U}^T\mathbf{C}\|^2 \tag{6.13}$$

and the solution becomes

$$\mathbf{w}^* = \mathbf{V}\mathbf{\Lambda}^*\mathbf{U}^T\mathbf{C} \tag{6.14}$$

where $\mathbf{\Lambda}^*$ is the transpose of $\mathbf{\Lambda}$ with all singular values whose ratio to the largest singular value is greater than N times the machine precision (10^{-6} in our case) replaced by their reciprocals.

One problem that can be observed in the solution of this unconstrained least-squares problem is that some of the weight components can be negative. We make them nonnegative by either shifting the weight vector by its smallest component (represented as "SVD+min" in the experiments), or by setting the negative components to zero (represented as "SVD+set 0" in the experiments).

6.4.2 Nonnegative Least-Squares (NNLS)

Another way of obtaining nonnegative weights is to solve the problem

$$\text{find } \mathbf{w} \text{ to minimize } \|\mathbf{D}\mathbf{w} - \mathbf{C}\|^2 \text{ subject to } \mathbf{w} \geq 0. \tag{6.15}$$

A nonnegative least-squares algorithm is given in [130] as follows:

1. Set $\mathcal{P} = \{\}$, $\mathcal{Z} = \{1, 2, \ldots, q\}$ and $\mathbf{w} = 0$.

2. Compute $\mathbf{u} = \mathbf{D}^T(\mathbf{C} - \mathbf{D}\mathbf{w})$.

3. If the set $\mathcal{Z}$ is empty or if $\mathbf{u}_i \leq 0 \ \forall i \in \mathcal{Z}$, terminate the algorithm.

4. Find an index $t \in \mathcal{Z}$ such that $\mathbf{u}_t = \max\{\mathbf{u}_i | i \in \mathcal{Z}\}$.

5. Move the index t from set $\mathcal{Z}$ to set $\mathcal{P}$.

6. Let $\mathbf{D}_{\boldsymbol{\mathcal{P}}} \in \mathbb{R}^{(N\times q)}$ denote the matrix defined by

$$\text{column } i \text{ of } \mathbf{D}_{\boldsymbol{\mathcal{P}}} = \begin{cases} \text{column } i \text{ of } \mathbf{D} & \text{if } i \in \mathcal{P} \\ \mathbf{0} & \text{if } i \in \mathcal{Z}. \end{cases} \tag{6.16}$$

   Compute $\mathbf{z} \in \mathbb{R}^{(q\times 1)}$ as a solution of the least-squares problem $\mathbf{D}_{\boldsymbol{\mathcal{P}}}\mathbf{z} = \mathbf{C}$. Note that only the components $\mathbf{z}_i, i \in \mathcal{P}$ are determined by this problem. Define $\mathbf{z}_i = 0$ for $i \in \mathcal{Z}$.

7. If $\mathbf{z}_i > 0\ \forall i \in \mathcal{P}$, set $\mathbf{w} = \mathbf{z}$ and go to step 2.

8. Find an index $p \in \mathcal{P}$ such that $\frac{\mathbf{w}_p}{\mathbf{w}_p - \mathbf{z}_p} = \min\{\frac{\mathbf{w}_i}{\mathbf{w}_i - \mathbf{z}_i} | \mathbf{z}_i \leq 0, i \in \mathcal{P}\}$.

9. Set $\alpha = \frac{\mathbf{w}_p}{\mathbf{w}_p - \mathbf{z}_p}$.

10. Set $\mathbf{w} = \mathbf{w} + \alpha(\mathbf{z} - \mathbf{w})$.

11. Move from set $\mathcal{P}$ to set $\mathcal{Z}$ all indices $i \in \mathcal{P}$ for which $\mathbf{w}_i = 0$. Go to step 6.

On termination, the vector $\mathbf{w}$ satisfies

$$\mathbf{w} = \begin{cases} \mathbf{w}_i > 0, & i \in \mathcal{P} \\ \mathbf{w}_i = 0, & i \in \mathcal{Z} \end{cases} \tag{6.17}$$

and becomes a solution to the least-squares problem.

6.4.3 Ridge Regression

Another way of overcoming the problem of singular matrices is ridge regression where the range of functions is restricted. This can also be regarded as reducing the degrees of freedom. Ridge regression introduces a term that penalizes large weights in addition to the squared error. The cost function to be minimized is

$$\epsilon^2 = \|\mathbf{D}\mathbf{w} - \mathbf{C}\|^2 + \lambda \mathbf{w}^T\mathbf{w}. \tag{6.18}$$

The regularization parameter $\lambda > 0$ controls the balance between the residual error and the penalty for large weights.

6.4.4 Constrained Non-linear Least-Squares

We can also constrain the weight vector to have unit length. Then, the problem becomes to

$$\text{find } \mathbf{w} \text{ to minimize } \|\mathbf{D}\mathbf{w} - \mathbf{C}\|^2 \text{ subject to } \|\mathbf{w}\| = 1. \tag{6.19}$$

After introducing a Lagrange multiplier, the cost function to be minimized becomes

$$\epsilon^2 = (\mathbf{D}\mathbf{w} - \mathbf{C})^T(\mathbf{D}\mathbf{w} - \mathbf{C}) + \lambda(\mathbf{w}^T\mathbf{w} - 1). \tag{6.20}$$

The optimum weight vector is the one that minimizes ϵ^2 as

$$\frac{\partial \epsilon^2}{\partial \mathbf{w}} = 2\mathbf{D}^T(\mathbf{D}\mathbf{w} - \mathbf{C}) + 2\lambda\mathbf{w} = 0 \quad \Rightarrow \quad \mathbf{w} = (\mathbf{D}^T\mathbf{D} + \lambda\mathbf{I})^{-1}\mathbf{D}^T\mathbf{C}. \tag{6.21}$$

A search algorithm can be used to find λ as the solution of

$$\frac{\partial \epsilon^2}{\partial \lambda} = \mathbf{w}^T\mathbf{w} - 1 = 0 \quad \Rightarrow \quad \mathbf{C}^T\mathbf{D}(\mathbf{D}^T\mathbf{D} + \lambda\mathbf{I})^{-2}\mathbf{D}^T\mathbf{C} = 1. \tag{6.22}$$

Then this λ can be substituted into Equation (6.21) to find the optimum weight.

6.4.5 Alternative Formulation

Given N data pairs $(\mathbf{d}_{\boldsymbol{i}}, c_i), i = 1, \ldots, N$, we try to find a least-squares fit to the model

$$\underbrace{(\mathbf{d}_{\mathbf{1}}, \cdots, \mathbf{d}_{\boldsymbol{N}})^T}_{\mathbf{D}} \ \mathbf{w} = \underbrace{(c_1, \cdots, c_N)^T}_{\mathbf{C}}, \qquad \mathbf{w} = \tag{6.23}$$

as

$$\text{find } \mathbf{w} \text{ to minimize } \|\mathbf{Dw} - \mathbf{C}\|^2 = \sum_{i=1}^{N} (\mathbf{d}_{\boldsymbol{i}}^T \mathbf{w} - c_i)^2. \tag{6.24}$$

Suppose each $\mathbf{d}_{\boldsymbol{i}}^T \mathbf{w}$ have a Gaussian distribution around the true value c_i. Also suppose that the standard deviations are the same, σ, for all data points. The distribution model can be defined as

$$p(\mathbf{d}) \propto e^{-(\mathbf{d}^T \mathbf{w} - c)^2 / 2\sigma^2}. \tag{6.25}$$

The likelihood function for the parameter $\mathbf{w}$ is

$$L(\mathbf{w} | \mathbf{d_1}, \ldots, \mathbf{d}_{\boldsymbol{N}}) \propto \prod_{i=1}^{N} e^{-(\mathbf{d}_{\boldsymbol{i}}^T \mathbf{w} - c_i)^2 / 2\sigma^2}. \tag{6.26}$$

The maximum likelihood estimate of $\mathbf{w}$ can be found as

$$\begin{aligned} \text{find } \mathbf{w} \text{ to maximize } L(\mathbf{w} | \mathbf{d_1}, \ldots, \mathbf{d}_{\boldsymbol{N}}) &\equiv \text{maximize } \log L(\mathbf{w} | \mathbf{d_1}, \ldots, \mathbf{d}_{\boldsymbol{N}}) \\ &\equiv \text{maximize } -\sum_{i=1}^{N} (\mathbf{d}_{\boldsymbol{i}}^T \mathbf{w} - c_i)^2 / 2\sigma^2 \\ &\equiv \text{minimize } \sum_{i=1}^{N} (\mathbf{d}_{\boldsymbol{i}}^T \mathbf{w} - c_i)^2 \end{aligned} \tag{6.27}$$

which is equivalent to the least-squares estimation problem in Equation (6.24).

The retrieval performances of the weight updating algorithms are evaluated in the next section.

6.5 Experiments

The feedback algorithms of this chapter were tested only on a smaller subset (340 images) of the ISL Database using only the line-angle-ratio and co-occurrence feature vectors. Retrieval results in terms of precision averaged over the groundtruth images are given in Figures 6.1 and 6.2. The search engine performs a new search in the database and retrieves 12 images in every iteration. In these experiments, we used only the weighted L_1 and L_2 distances with independent updating. The results are

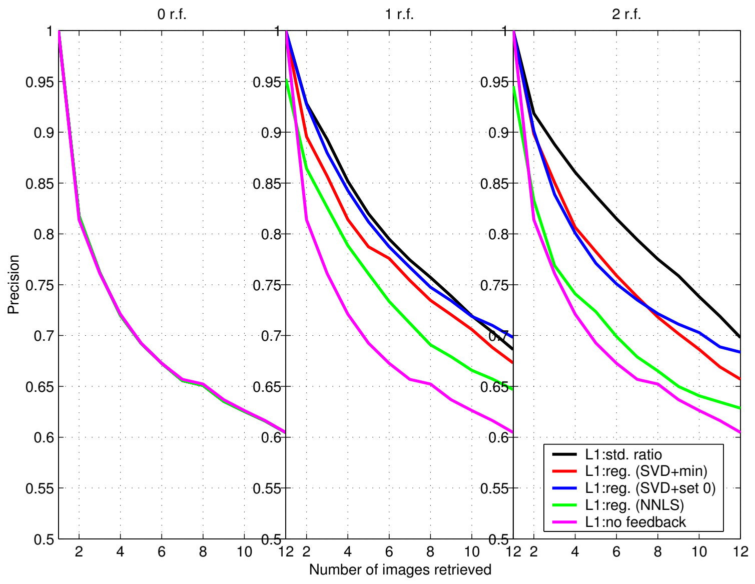


Figure 6.1: Average precision using independent weight updating for the L_1 distance with different weight estimation methods for the first two iterations. "n r.f." represents the n'th feedback iteration.

summarized in Table 6.2. All the feedback methods showed an improvement over the case without feedback. We can see that slightly better results could be obtained on the average when the L_2 distance was used. We tried up to five iterations and the largest average improvement was obtained as 19% after the first iteration using the standard deviation ratio-based updating. The largest average improvement among regression-based methods with the weighted L_1 distance was obtained as 15.48% after the first iteration when SVD was used to update the weights and the negative weight components were set to zero ("SVD+set 0"). The constrained nonlinear least-squares algorithm performed worse than all the other weight updating methods.

When the whole database is searched in every iteration, the improvement is usually a few additional relevant images and this can also be achieved by showing a new set of images from the retrieval set of the original query instead of waiting for the computation of getting feedback and doing one more search. Another way of inves-

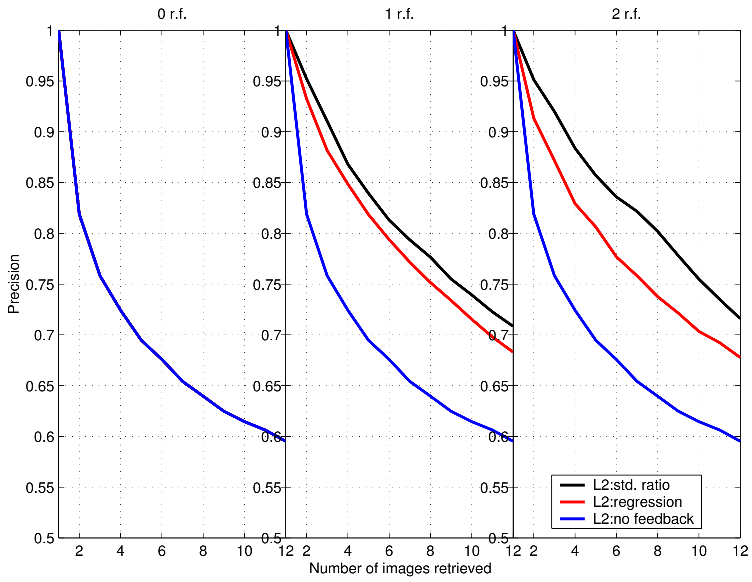


Figure 6.2: Average precision using independent weight updating for the L_2 distance with different weight estimation methods for the first two iterations. "n r.f." represents the n'th feedback iteration.

Table 6.2: Average precision when 12 images are retrieved using independent weight updating for the L_1 and L_2 distances with different weight estimation methods. Improvements for each iteration over the previous iteration are given in parentheses. "n r.f." represents the n'th feedback iteration.

Distance	0 r.f.	1 r.f.	2 r.f.
L_1 (std. ratio)	0.60	0.69 (13.53%)	0.70 (1.71%)
L_1 (SVD+min)	0.60	0.67 (11.33%)	0.66 (-2.38%)
L_1 (SVD+set 0)	0.60	0.70 (15.48%)	0.68 (-2.07%)
L_1 (NNLS)	0.60	0.65 (7.00%)	0.63 (-2.84%)
L_2 (std. ratio)	0.60	0.71 (19.03%)	0.72 (1.06%)
L_2 (SVD+min)	0.60	0.68 (14.76%)	0.68 (-0.78%)

tigating how well the relevance feedback performs is to compare the performance of iterative retrieval with that of the original search in terms of the *progress* made towards retrieving a specific number of images. After obtaining the feedback, the search engine performs a new search in the database but ignores all the images that were retrieved in previous iterations. Therefore, a new set of 12 images are retrieved in every iteration. This performance is compared with the case where the next set of 12 images from the retrieval set of the original search are presented to the user (showing the next page in the user interface). Given n as a specific number of images retrieved, we define *progress* as the ratio of two precisions,

$$Progress = \frac{\#\{relevant\ images\ among\ n\ after\ \lceil n/12 \rceil\ iterations\}}{\#\{relevant\ images\ among\ n\ retrieved\ without\ feedback\}}. \tag{6.28}$$

When progress is greater than 1, it means the feedback algorithm is effective and converges faster. Average progress is given in Figures 6.3 and 6.4. The results are summarized in Table 6.3. We can see that incremental updating performed better than independent updating. The largest improvement for the standard deviation ratio-based updating was obtained as 5.3% greater progress over the no feedback case. The largest average improvement among regression-based methods with the weighted L_1 distance was obtained as 7.4% greater progress over the case without feedback after the second iteration with "SVD+set 0".

We also investigated the residual errors in the regression solutions. By considering only the first iteration, the weight vector that was computed using singular value decomposition resulted in the smallest residual error. But, when we set the negative weight components to zero, the residual error increased significantly. On the other hand, the nonnegative least-squares algorithm gave residual errors that were larger than but comparable to the ones with singular value decomposition with full weight vectors. The nonnegative least-squares algorithm also resulted in many zero weight components (7 out of 28 weight components were positive on the average). On the other hand, singular value decomposition resulted in more positive weight components

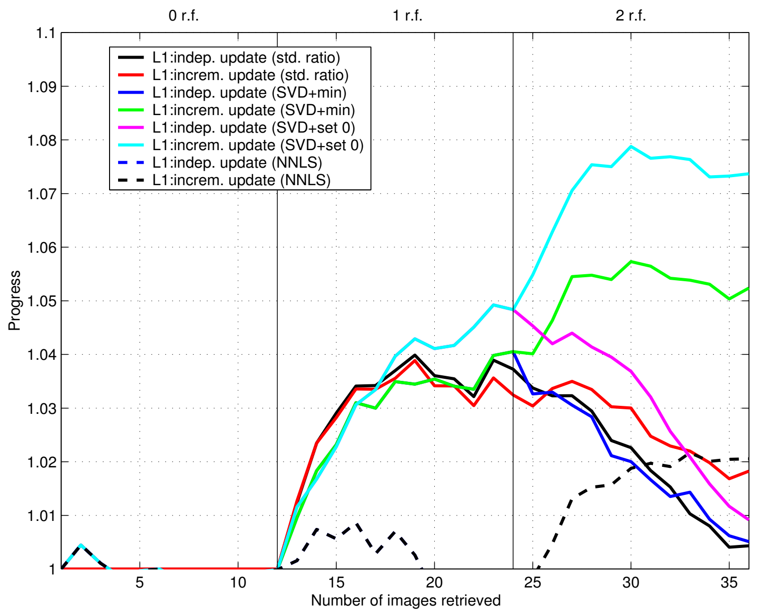


Figure 6.3: Average progress using independent and incremental weight updating for the L_1 distance with different weight estimation methods for the first two iterations. "n r.f." represents the n'th feedback iteration.

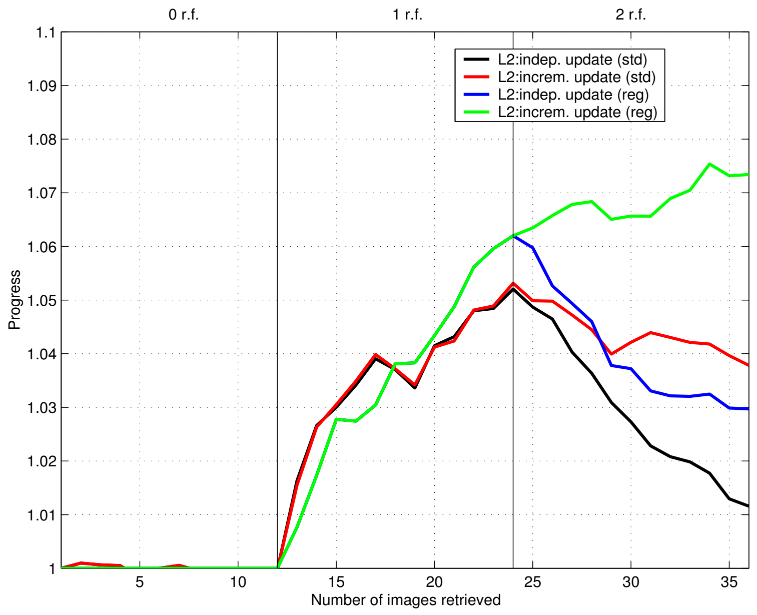


Figure 6.4: Average progress using independent and incremental weight updating for the L_2 distance with different weight estimation methods for the first two iterations. "n r.f." represents the n'th feedback iteration.

Table 6.3: Average progress using independent and incremental weight updating for the L_1 and L_2 distances with different weight estimation methods when 12, 24 and 36 images are retrieved. "n r.f." represents the n'th feedback iteration.

Distance	0 r.f.	1 r.f.	2 r.f.
L_1 (std. ratio):Indep. update	1	1.037	1.004
L_1 (std. ratio):Increm. update	1	1.033	1.018
L_1 (SVD+min):Indep. update	1	1.041	1.005
L_1 (SVD+min):Increm. update	1	1.041	1.052
L_1 (SVD+set 0):Indep. update	1	1.048	1.009
L_1 (SVD+set 0):Increm. update	1	1.048	1.074
L_1 (NNLS):Indep. update	1	0.991	0.935
L_1 (NNLS):Increm. update	1	0.991	1.021
L_2 (std. ratio):Indep. update	1	1.052	1.012
L_2 (std. ratio):Increm. update	1	1.053	1.038
L_2 (SVD+min):Indep. update	1	1.062	1.030
L_2 (SVD+min):Increm. update	1	1.062	1.073

(15 positive weight components on the average).

The experiments showed that all of the feedback methods gave an improvement over the case without feedback. However, their effectiveness is limited by the problems of the geometric similarity framework as discussed in Chapter 4. Furthermore, the weights are defined only for the components of a single feature vector and they do not support multiple feature vectors in the current setting. One way to overcome this problem is to formulate a hierarchical weighting scheme [172], which may, unfortunately, still be effected by the limitations of the geometric framework.

In the following chapter, we will propose a Bayesian framework that results in a powerful way of both combining different feature vectors and incorporating relevance feedback with a performance better than two competing algorithms from the recent content-based image retrieval literature.

Chapter 7

A UNIFIED FRAMEWORK FOR FEATURE AND SIMILARITY COMBINATION

7.1 Introduction

Numerous feature extraction methods and similarity measures that were designed for particular applications have been proposed in the literature. However, there seems to be no general method or a formal approach which is useful in a broad range of images. Results of the previous chapters also showed that low-level features cannot always map visually similar images to nearby locations in the feature space and distance-based similarity measures often retrieve images that are quite irrelevant to the query image. Therefore, we discussed two post-processing algorithms; graph-theoretic clustering and a relevance feedback approach with weighted distances, and showed that improvements in both image grouping and retrieval can be obtained using these methods. However, it is still hard to generalize these methods to mimic the high-level notion of similarity in humans.

An important observation in Chapter 4 was that different features and different similarity measures performed differently for different types of images. Therefore, developing a framework to combine feature vectors and similarity measures looks promising to improve the overall performance. One simple method to combine multiple features is to append different feature vectors and treat the result as a big global feature vector [131]. Another method is to compute distances using each feature vector separately and then use a linear or Boolean combination of them as the final distance measure [24].

In the database retrieval literature, systems such as QBIC [74] and Virage [15] offered the user the ability to take weighted combinations of color, texture, shape, and position measures in combination with keyword search. Vailaya and Jain [203] used the weighted sum of distances based on edge direction histograms and moment invariants. They also used the weighted sum of distances based on edge direction histograms and color histograms [200]. However, in all of these approaches, the weights have to be assigned by the user. This is not generally applicable because the ordinary user does not usually have a detailed understanding of the features designed by an expert. Furthermore, all of these methods still suffer from the geometric retrieval problem.

To make use of different features for different types of images, Vailaya *et al.* [200] used a hierarchical classifier to first classify images into the city or landscape classes using edge direction coherence vectors with a nearest neighbor classifier, then to classify landscape images into forests, mountains and sunset/sunrise classes using color coherence vectors. In [199], they used a Bayesian classifier for the same classification task. The class-conditional densities were estimated using a non-parametric vector quantization approach. They also used the same type of classifier for automatic image orientation detection [202]. The classifier was used to assign an image into one of the four orientations: 0°, 90°, 180°, 270°. However, this is applicable only for specific groups of images that can be categorized in binary or a small number of groups.

The biggest drawback of the above system is that the defined classes (city, landscape, forest, mountain, sunset, etc.) are assumed to form a partition of the database and each image in the database is required to be a member of one and only one of these groups. When the database gets larger and more complex, the number of classes that needs to be defined increases, the limit being a class for each image. Vasconcelos and Lippman [206] proposed such an approach where each image in the database was assumed to be a class and the goal was to assign the query image to one of these classes. They used Discrete Cosine Transform coefficients for small non-overlapping neighborhoods in an image as features, and mixtures of Gaussians for class-conditional distributions. This approach is applicable to other types of feature vectors that can be computed for each pixel or for small neighborhoods in an image so that a sample of the feature vectors can be used to train the distributions for that image. However, it cannot be generalized for feature vectors that require larger neighborhoods or when each image is represented by a single feature vector because there will not be enough data to estimate the parameters of any distribution.

Dy *et al.* [68] used a similar approach by two-level hierarchical classifiers. They tried to classify a query using features that best differentiate the major classes and then specialized the query to that class by using the features that best distinguish the images within the chosen major class. The motivation came from the observation that the features that are the most effective in discriminating among images from different classes may not be the most effective for retrieval of visually similar images within a class. They used a nearest neighbor classifier in the first level, then used a k-nearest neighbor classifier in each class. This method does not have the binary categorization restriction but the nearest neighbor classifier still computes similarity in the geometric setting.

In another approach to combine different methods in the distance level, Berman and Shapiro [22] proposed the following set of operations to enable more expressive queries: Addition ($\rho = \rho_1 + \rho_2$), weighting ($\rho = \alpha\rho_1, \alpha \geq 0$), maximum ($\rho = \max(\rho_1, \rho_2)$), and minimum ($\rho = \min(\rho_1, \rho_2)$), where ρ_1 and ρ_2 are two distance measures. They showed that these operators satisfy the triangle inequality and applied triangle inequality-based pruning algorithms to distance measures that were combined using these operations. They also experimented with polynomial combinations of distance measures to create new distance measures and extended the triangle inequality to compute lower bounds for these new measures to prune the database [21]. However, they decided that the functionality was too unintuitive with little apparent gain in utility.

Recently, relevance feedback has been used as a tool to improve retrieval performance as discussed in Chapter 2. It was also used to combine different feature vectors or similarity measures. Benitez *et al.* [20] used relevance feedback to rank search engines to select a search engine that gave the best results for a given query. All the search engines were initially given equal weights and then these weights were heuristically increased or decreased based on the images they retrieved and the feedback given by the user. A heuristic ordering method was used where results from search engines with better ranks were displayed before the results from others. No detailed performance of this approach is available. Rui *et al.* [172] also used a similar weighting approach but in the feature and distance levels. User's feedback was used to update the weights that were used to compute a weighted linear combination of distance values for different feature vectors. However, the weights were assigned heuristically and similarity was based on geometric distances as in most of the other approaches described in Chapter 2.

Neural networks have also been used as a tool for feature combination. Haering *et al.* [87] used a neural network with features like color, roughness, directionality, co-occurrence features, Fourier features, Gabor features and fractal features as input and trained it to detect deciduous trees. Haering and de Vitoria Lobo [86] again used 51 features like Fourier features, Gabor features, steerable bar and step edge filters, co-occurrence features, fractal dimension measures, HSV color features and entropy measures for locating deciduous trees in images. They compared the performance of a back-propagation neural network against those of convolutional neural networks, Fisher's linear classifiers, Gaussian quadratic classifiers, eigenanalysis, and minimally correlated features and concluded that the back-propagation neural network performed the best. Oh *et al.* [152, 153] also used neural network classifiers with two methods to combine different features in a handwriting recognition application. They first used a class-common approach where the features were chosen according to their class separation performances when all classes were combined, then used a single final classifier. The second approach was a class-dependent approach where features were chosen specifically for each class and sub-classifiers were used for individual cases.

There has been a lot of research on feature and decision combination in the handwriting and speech recognition areas. Most of these methods try to combine the outputs of multiple classifiers to arrive at a final decision. Possible methods for combinations in the classifier level are:

- A weighted majority vote (each classifier makes a binary decision (vote) about each class and the final decision is made in favor of the class with the largest number of votes) [211, 126, 127, 120],

- Sum, product, maximum, minimum and median of the *a posteriori* probabilities computed by individual classifiers [119, 120],

- Softmin and softmax combination rules for the outputs of individual classifiers that also allow joint training [118],

- Class ranking (each class receives m ranks from m classifiers, the highest (minimum) of these ranks is the final score for that class) [94],

- Borda count (sum of the number of classes ranked below a class by each classifier) [94],

- Discrete optimization of the overall probability of correct classification using ranked-based classifiers [176],

- Weighted combination of classifiers [121],

- Hierarchical multiple classifiers (component classifiers are built using clusters of training data and a super-classifier is built using the outputs of component classifiers and clustering information) [45].

However, as discussed above, image retrieval problem is usually set as ranking images according to their distances to the query image in the feature space. Unfortunately, probabilities are not usually obtained in this geometric retrieval setting and the classifier-based combination methods cannot be directly applicable in the commonly used framework. Hence, it is apparent that combining feature vectors and similarity measures is not promising to be effective enough in the geometric retrieval framework.

In this dissertation, we pose the retrieval problem in a probabilistic framework. We define a mapping from the high-dimensional feature space to the two-dimensional probability space and use class-conditional probabilities $p(\mathbf{d}|\mathcal{A})$ and $p(\mathbf{d}|\mathcal{B})$ for retrieval. One important advantage of this framework is that it provides a natural way to combine multiple measurements on images. Since each possible combination of feature vectors, similarity models and classifiers gives a set of these class-conditional probabilities, the classifier combination methods listed above can be directly used to arrive at a final decision about the similarity between images. The classifier combination methods that we use will be given in Section 7.2.

There have been many classifiers, like linear classifiers, quadratic classifiers, decision tree classifiers, ML and MAP classifiers, neural network classifiers, and rule-based classifiers that were proposed and applied to specific classification tasks; however, the naive Bayesian classifier [63] is still competitive with the state-of-the-art classifiers [75, 61, 62]. The classification is done by applying the Bayes rule to compute the probability of a class given a particular instance of the pattern's attribute values by making an independence assumption: all the attributes are conditionally independent given the value of the class. Even though this assumption looks like a very strong one, Domingos and Pazzani [61, 62] showed that the naive Bayesian classifier can still be optimal when this assumption is violated by a wide margin. Their conclusion was that correct classification can be achieved even when the probability estimates contain large errors. The classifier combination methods listed above are also based on this conditional independence assumption.

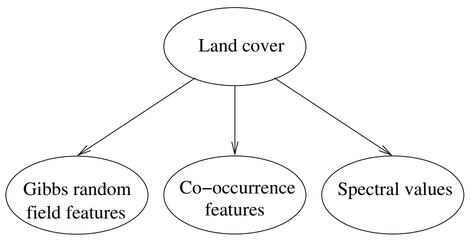
The naive Bayesian classifier can be studied in the framework of Bayesian networks. Even though we showed that the class-conditional probabilities were successful in both classification and retrieval, an important fact is that they also include uncertainty because they are estimated from noisy data, the density models are not perfect and there are additional factors like quantization and high-dimensionality. An additional modeling of probability can be explicitly incorporated as "probability of probability" in a Bayesian network framework. Bayesian networks have become a popular representation for encoding uncertain knowledge in expert systems in the last decade. Researchers have developed methods for learning Bayesian networks from a combination of expert knowledge and data. Some example applications [108, 14, 93] include medical diagnosis systems, speech recognition, space missions, weather forecasting, technical support troubleshooting and help wizards in Microsoft products, document retrieval, sensor fusion and junk e-mail filtering.

A recent successful application of Bayesian networks for image retrieval was described by Schroder *et al.* [177] where a naive Bayesian classifier was used to link user interests and signal models to iteratively learn user-specific land cover types in a remote sensing image archive. They first performed feature extraction from image data. Then an unsupervised classification algorithm was used on these feature vectors to obtain class labels for each feature type for each pixel. The feature vectors they used were computed from Gibbs random fields, co-occurrence matrices and spectral values. The next level of information contained the land cover types. The network they used is given in Figure 7.1(a). They iteratively computed the posterior probability of a pixel belonging to that cover type given its feature classes. Since the class label was a binary variable and there were a fixed number of feature classes (specified in the unsupervised clustering algorithm), they used relative frequency tables to represent the conditional probabilities in the naive Bayesian network. They started with uniform priors and used user's feedback to iteratively update the conditional probabilities using relative frequencies. User feedback was in the form of clicks on pixels as positive and negative examples for the cover type. Then, this trained classifier was used to perform a search in the database to rank images according to their posterior probabilities. They showed retrieval examples and also used a small groundtruth to compare the classified pixels with manually labeled regions using a confusion matrix. The average correct detection rate was given to be 90%.

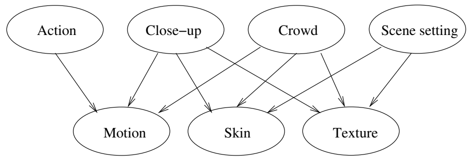
Kumar and Desai [124] also used a discrete variable Bayesian network to identify types of the segments of an image. They estimated the conditional probabilities from histograms of feature values. They showed an aerial image as an example of the application of their method.

Another application of Bayesian networks for retrieval was presented by Vasconcelos and Lippman [204]. They used the fact that movie production usually has specific conventions and structure, and used a Bayesian framework to incorporate this structure in video summarization and classification. They used sensors (algorithms) trained to detect relevant visual features and used a Bayesian network to link semantic content descriptors with these sensors. The sensors they used were "action", "skin" and "texture", and the semantic attributes they used were "action", "scene setting" (man-made or nature), "close-up" and "crowd". The network they used is given in Figure 7.1(b). They showed examples on 100 video clips from a movie and obtained 88% classification accuracy. However, all the variables in the network were binary and both the structure and the conditional probabilities were set manually.

Recently, Tieu and Viola [196] used boosting for image retrieval. They first extracted a large number of features from oriented edges and bar filters. The total number of features were 46,875 for RGB in three resolutions. Their motivation was that highly selective features would respond to only a small percentage of images in the database. Then, given positive and negative query images, they trained 20 classifiers that selected individually highly selective features along which positive examples were more distinct from the negative examples. Boosting was used to combine these



(a) Network used by Schroder *et al.* [177]



(b) Network used by Vasconcelos and Lippman [204]

Figure 7.1: Example Bayesian networks used for content-based image or video classification.

classifiers with a final strong classifier. Precision-recall results on 500 COREL images showed that boosting was effective and also allowed relevance feedback.

In the rest of the chapter we first summarize classifier combination in Section 7.2. Then, in Section 7.3, we give a brief introduction to Bayesian networks, describe how to construct a Bayesian network with its associated probability distributions (i.e. offline training), and present the methodology to update these distributions when new data is available (i.e. online updating). Finally, we describe the details of our approach that allows us to extend the feature representations and similarity models that were described in previous chapters into a combined framework. Furthermore, this framework offers a relevance feedback architecture to incorporate the user's feedback to further improve the performance. Performance evaluation of the framework proposed in this chapter is done using extensive classification and retrieval experiments in Chapter 8.

7.2 Classifier Combination

We have used different classifiers, e.g. linear classifiers, quadratic classifiers, nearest neighbor classifiers, decision tree classifiers, neural network classifiers in the two-class classification problem in Section 4.4. An empirical comparison of different classifiers like in Section 4.5.1 lets us choose one of them as the best classifier for the problem at hand. However, although most of the classifiers may have similar error rates, sets of image pairs misclassified by different classifiers do not necessarily overlap. Classifier combination is motivated by the goal of further improving the classification performance by not relying on a single decision but rather by combining the decisions made by the individual classifiers.

Kittler *et al.* [119, 120] and Duin and Tax [67] observed that combinations of different classifiers that used different feature vectors and different training sets were the most useful. In this scenario, the classifiers operate in different measurement spaces and each classifier uses its own representation of the input pattern.

In this section, we discuss classifier combination in a Bayesian framework. In the general problem of assigning the pattern ξ to one of the m classes $\mathcal{C}_1, \ldots, \mathcal{C}_m$ with prior probabilities $p(\mathcal{C}_1), \ldots, p(\mathcal{C}_m)$ using n classifiers with measurement vectors $\mathbf{x_1}, \ldots, \mathbf{x}_{\boldsymbol{n}}$, the Bayesian classifier makes the decision using *a posteriori* probabilities as

$$\begin{aligned} \text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k &= \arg\max_{j=1}^{m} p(\mathcal{C}_j | \mathbf{x_1}, \ldots, \mathbf{x}_{\boldsymbol{n}}) \\ &= \arg\max_{j=1}^{m} p(\mathbf{x_1}, \ldots, \mathbf{x}_{\boldsymbol{n}} | \mathcal{C}_j) p(\mathcal{C}_j). \end{aligned} \tag{7.1}$$

In a practical situation where we have limited training data, computing the joint probability density $p(\mathbf{x_1}, \ldots, \mathbf{x}_{\boldsymbol{n}} | \mathcal{C}_j)$ for each class will be difficult. Therefore, we need to make some assumptions to simplify the decision rule. The following sections describe some possible and common assumptions in the literature. We follow the framework of Kittler *et al.* [120].

7.2.1 Product Rule

Assuming that the measurements $\mathbf{x_1}, \ldots, \mathbf{x}_{\boldsymbol{n}}$ are conditionally statistically independent given the class, the joint class-conditional probability in Equation (7.1) can be written as

$$p(\mathbf{x_1}, \ldots, \mathbf{x_n} | \mathcal{C}_j) = \prod_{i=1}^{n} p(\mathbf{x_i} | \mathcal{C}_j) \tag{7.2}$$

where $p(\mathbf{x_i}|\mathcal{C}_j)$ is the class-conditional model for the i'th classifier under class $\mathcal{C}_j$. Then, the decision rule becomes

$$\begin{aligned} \text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \ k &= \arg\max_{j=1}^{m} \left[p(\mathcal{C}_j) \prod_{i=1}^{n} p(\mathbf{x_i}|\mathcal{C}_j) \right] \\ &= \arg\max_{j=1}^{m} \left[(p(\mathcal{C}_j))^{-(n-1)} \prod_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x_i}) \right] \end{aligned} \tag{7.3}$$

where $p(\mathcal{C}_j|\mathbf{x_i})$ are the posterior probabilities for each classifier under class $\mathcal{C}_j$. Under the assumption of equal priors, Equation (7.3) becomes

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \ k = \arg\max_{j=1}^{m} \prod_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x_i}). \tag{7.4}$$

The conditional independence assumption may not always hold but it gives a practical approximation and the errors caused by this assumption will not be too severe if we use different feature vectors and different classifiers in the combination [67].

7.2.2 Sum Rule

If we assume that the posterior probabilities from individual classifiers will not deviate dramatically from the corresponding prior probabilities, they can be rewritten as

$$p(\mathcal{C}_j|\mathbf{x_i}) = p(\mathcal{C}_j)(1 + \epsilon_{ji}) \tag{7.5}$$

where $\epsilon_{ji} \ll 1$. Substituting this approximation in Equation (7.3) and neglecting any terms of second and higher order of ϵ_{ji} gives the decision rule

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \ k = \arg\max_{j=1}^{m} \left[(1-n)p(\mathcal{C}_j) + \sum_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x_i}) \right]. \tag{7.6}$$

Under the assumption of equal priors, Equation (7.6) becomes

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \sum_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}). \tag{7.7}$$

The assumption that the posterior probabilities from individual classifiers do not deviate dramatically from the priors will be unrealistic in some cases. However, this approximation will have a low sensitivity to estimation errors [120].

7.2.3 Max Rule

Approximating the sum in Equation (7.6) by the maximum of the posterior probabilities[1] gives the decision rule

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \left[(1-n)p(\mathcal{C}_j) + n \max_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}) \right] \tag{7.8}$$

which under the assumption of equal priors becomes

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \max_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}). \tag{7.9}$$

7.2.4 Min Rule

Approximating the product in Equation (7.3) by the minimum of the posterior probabilities[2] gives the decision rule

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \left[(p(\mathcal{C}_j))^{-(n-1)} \min_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}) \right] \tag{7.10}$$

which under the assumption of equal priors becomes

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \min_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}). \tag{7.11}$$

---

[1] $\frac{1}{n}\sum_{i=1}^{n} a_i \leq \max_{i=1}^{n} a_i$ for $a_i \in \mathbb{R}$.

[2] $\prod_{i=1}^{n} a_i \leq \min_{i=1}^{n} a_i$ for $0 \leq a_i \leq 1$.

7.2.5 Median Rule

Using the fact that median is a robust estimate of the mean, approximating the sum in Equation (7.6) by the median of the posterior probabilities gives the decision rule

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \operatorname*{median}_{i=1}^{n} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}) \tag{7.12}$$

under the assumption of equal priors.

7.2.6 Harmonic Mean Rule

Another measure of central tendency is the harmonic mean. Therefore, the sum in Equation (7.6) can be replaced by the harmonic mean and the decision rule becomes

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \left[\sum_{i=1}^{n} (p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}))^{-1} \right]^{-1} \tag{7.13}$$

under the assumption of equal priors. Note that this rule is valid when the posterior probabilities $p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}})$ are positive.

7.2.7 Majority Vote Rule

If we set each classifier to make a binary decision

$$\delta_{ki} = \begin{cases} 1 & \text{if } k = \arg \max_{j=1}^{m} p(\mathcal{C}_j|\mathbf{x}_{\boldsymbol{i}}) \\ 0 & \text{otherwise,} \end{cases} \tag{7.14}$$

approximating the sum in Equation (7.6) by the individual binary decision outcomes gives the decision rule

$$\text{assign } \xi \text{ to } \mathcal{C}_k \text{ if } \; k = \arg \max_{j=1}^{m} \sum_{i=1}^{n} \delta_{ji} \tag{7.15}$$

under the assumption of equal priors. The sum in Equation (7.15) counts the votes received by each class from individual classifiers and the final decision is made in favor of the class with the largest number of votes.

All of these combination methods are based on the conditional independence assumption. Furthermore, some additional conditions need to be satisfied for the classifier combination methods to improve the classification performance. The individual classifiers should not be strongly correlated in their misclassification, i.e. they should not agree with each other when they misclassify a sample, or at least they should not assign the same incorrect class to a particular sample. This requirement can be satisfied to a certain extent by using different feature vectors and different classifiers.

A numerical measure of dependency between classifiers is the Q statistic [125]. Given a labeled sample of n data points, the correct decisions made by the i'th classifier can be collected in a vector $\mathbf{T}_{\boldsymbol{i}} = (t_{i1}, \ldots, t_{ik}, \ldots t_{in})^T \in \mathbb{R}^{(n\times 1)}$ where t_{ik} is 1 if the i'th classifier correctly classifies the k'th data point and 0 otherwise. For the pair of classifiers i and j with the decision vectors $\mathbf{T}_{\boldsymbol{i}}$ and $\mathbf{T}_{\boldsymbol{j}}$ respectively, the statistic Q_{ij} is given as

$$Q_{ij} = \frac{N_{11}N_{00} - N_{01}N_{10}}{N_{11}N_{00} + N_{01}N_{10}} \tag{7.16}$$

where $N_{ab} = \#\{k | t_{ik} = a \wedge t_{jk} = b, 1 \leq k \leq n\}$ for $a \in \{0,1\}$ and $b \in \{0,1\}$. Q varies between -1 and 1, and will be 0 for statistically independent classifiers. Classifiers that tend to classify the same objects similarly will have positive values of Q and those which make errors on different objects will have negative Q values. For more than two classifiers, the average of the pairwise Q values can be used as the overall measure.

Extensions of this framework to support multiple feature vectors and similarity measures as well as relevance feedback will be described in Section 7.4.1. Experiments and details of our system are described in Chapter 8.

7.3 Bayesian Networks

Bayesian networks [157] are directed acyclic graphs that allow effective representation of the joint probability distribution of a set of random variables. Therefore, they provide a tool to deal with two problems: uncertainty and complexity. There are two components of a Bayesian network model $\mathcal{M} = \{\mathcal{G}, \mathbf{\Theta}\}$: the graph $\mathcal{G}$ that describes the variables and their structural relationships, and the set $\mathbf{\Theta}$ of parameters for probability distributions associated with each variable. Each node in the graph represents a random variable and edges represent conditional independence relationships between the variables.

Let $\mathcal{M}$ be a Bayesian network over the set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$. Then, the joint probability distribution is given as

$$\begin{aligned} p(\mathcal{X}) &= p(x_1, \ldots, x_n) \\ &= \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}) \end{aligned} \tag{7.17}$$

using the chain rule of probability. The conditional independence relationships encoded in the Bayesian network state that a node x_i is conditionally independent of its ancestors given its parents $\boldsymbol{\pi_i}$. Therefore,

$$p(\mathcal{X}) = \prod_{i=1}^{n} p(x_i | \boldsymbol{\pi_i}). \tag{7.18}$$

Once we know the joint probability distribution encoded in the network, we can answer all possible inference questions about the variables using marginalization.

There are two groups of problems involved in Bayesian network design: learning and inference. Learning includes estimating the network structure and the parameters of the probability distributions from data and prior information (e.g. expert knowledge, causal relationships) if available. The simplest situation is the one where the network structure is completely known (either specified by an expert or designed using the casual relationships between the variables in the problem domain) and there are no unobserved variables in the training data. Other situations with increasing complexity are known structure but unobserved variables, unknown structure with observed variables and unknown structure with unobserved variables. It is possible to learn both structure and parameters from data. However, learning structure is much harder than learning parameters. Inference involves computing information about some of the variables given the information about others. It is not always possible to make exact inferences because of the complexity of the network structure but approximation algorithms exist. Books [157, 109, 128], detailed tutorials [92, 34, 122, 110] and survey papers [35] have been written to present different aspects of Bayesian networks.

Even though knowledge interpretation in Bayesian networks is similar to that in neural networks, Bayesian networks have two advantages. First, they can incorporate expert knowledge to increase the efficiency and accuracy of the knowledge extracted from data. Second, they present the relationships between variables in a more causal way and therefore enhance the understandability of the knowledge in the representation.

One needs to specify two things to fully characterize a Bayesian network: the structure and the parameters. Here we assume that the structure is known *a priori* and all the variables are always observable (complete data). In the next section, we describe methods to estimate the parameters of the conditional distributions $p(x_i|\boldsymbol{\pi_i})$.

7.3.1 Learning Parameters

Let the joint probability distribution of the variables in the network with parameter set $\boldsymbol{\Theta}$ be

$$\begin{aligned} p(\mathcal{X}|\boldsymbol{\Theta}) &= p(x_1, \ldots, x_n|\boldsymbol{\Theta}) \\ &= \prod_{i=1}^{n} p(x_i|\boldsymbol{\pi_i}, \boldsymbol{\theta_i}) \end{aligned} \tag{7.19}$$

where $\boldsymbol{\theta_i}$ is the vector of parameters for the conditional distribution of x_i and $\boldsymbol{\Theta} = (\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_n})$. Given training data $\mathcal{D} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ as a random sample from this joint distribution, we can define the goal of learning as finding the parameters of each conditional probability distribution that maximizes the likelihood of the training data

$$\begin{aligned} L(\boldsymbol{\Theta}|\mathcal{D}) &= \prod_{l=1}^{m} p(\mathcal{X}_l|\boldsymbol{\Theta}) \\ &= \prod_{l=1}^{m}\prod_{i=1}^{n} p(x_{li}|\boldsymbol{\pi_i}, \boldsymbol{\theta_i}) \end{aligned} \tag{7.20}$$

where x_{li} is the i'th variable in the training set $\mathcal{X}_l$. This is equivalent to maximizing the log-likelihood

$$\log L(\boldsymbol{\Theta}|\mathcal{D}) = \sum_{l=1}^{m}\sum_{i=1}^{n} \log p(x_{li}|\boldsymbol{\pi_i}, \boldsymbol{\theta_i}). \tag{7.21}$$

We see that the likelihood decomposes according to the structure of the network; therefore, we can maximize the contribution of each node independently assuming the parameters for each node are independent of the parameters of the other nodes (global parameter independence) [92]. This procedure gives the maximum likelihood estimates (MLE).

Another way of estimating the parameters is to assign a prior probability density function $p(\boldsymbol{\theta_i})$ to each $\boldsymbol{\theta_i}$ and use the training data $\mathcal{D}$ to compute the posterior distribution $p(\boldsymbol{\theta_i}|\mathcal{D})$ and Bayes estimate $E_{p(\boldsymbol{\theta_i}|\mathcal{D})}[\boldsymbol{\theta_i}]$. Next, we present learning methods for the conditional probability distributions for discrete and continuous variables.

Discrete Variables With Discrete Parents

The most commonly used types of variables in Bayesian networks are discrete variables. Let each discrete variable x_i have r_i possible values (states) with probabilities

$$p(x_i = k|\boldsymbol{\pi_i} = j, \boldsymbol{\theta_i}) = \boldsymbol{\theta_{ijk}} > 0 \tag{7.22}$$

where $k \in \{1, \ldots, r_i\}$, j is the state of x_i's parents and $\boldsymbol{\theta_i} = \{\{\boldsymbol{\theta_{ijk}}\}_{k=2}^{r_i}\}_{j=1}^{q_i}$ with $q_i = \prod_{x_l \in \boldsymbol{\pi_i}} r_l$ (the parameter $\boldsymbol{\theta_{i1}}$ can be calculated as $\boldsymbol{\theta_{i1}} = 1 - \sum_{k=2}^{r_i} \boldsymbol{\theta_{ijk}}$), i.e. having a multinomial distribution for every combination of $\boldsymbol{\pi_i}$. Given the training data $\mathcal{D}$, the MLE of $\boldsymbol{\theta_{ijk}}$ can be found as

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \frac{N_{ijk}}{N_{ij}} \tag{7.23}$$

where N_{ijk} is the number of cases in $\mathcal{D}$ in which $x_i = k$ and $\boldsymbol{\pi_i} = j$, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

However, MLEs are known to be sensitive to sparse data. When the sample is small and the number of parameters is large, we can have unreliable estimates. Another way is to use the Bayes estimate. First, we need to assign a prior distribution $p(\boldsymbol{\Theta})$ for $\boldsymbol{\Theta}$. Under the parameter independence assumptions, the parameters remain independent given the random sample [92], i.e.

$$p(\boldsymbol{\Theta}|\mathcal{D}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\boldsymbol{\vartheta_{ij}}|\mathcal{D}) \tag{7.24}$$

where $\boldsymbol{\vartheta_{ij}} = \{\boldsymbol{\theta_{i}}_{j2}, \ldots, \boldsymbol{\theta_{i}}_{jr_i}\}$. The global independence assumption states that the parameters for each variable are independent of the parameters of the other variables. The local independence assumption states that parameters for the states of a variable conditioned on each configuration of its parents are also independent. Thus, we can update each $\boldsymbol{\vartheta_{ij}}$ independently.

We can choose any prior for $\boldsymbol{\vartheta_{ij}}$ but there is a big advantage to use conjugate priors. A conjugate prior is one which, when multiplied with the direct probability, gives a posterior probability having the same functional form as the prior, thus allowing the posterior to be used as a prior in further computations [30].

The conjugate prior for the multinomial distribution is the Dirichlet distribution [56]. Geiger and Heckerman [80] showed that if all allowed states of the variables are possible (i.e. $\boldsymbol{\theta_{i}}_{jk} > 0$), then parameter independence[3] imply that the physical probabilities for complete network structures must have a Dirichlet distribution specified as

$$\begin{aligned} p(\boldsymbol{\vartheta_{ij}}) &= \mathrm{Dir}(\boldsymbol{\vartheta_{ij}}|\alpha_{ij1}, \ldots, \alpha_{ijr_i}) \\ &= \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} (\boldsymbol{\theta_{i}}_{jk})^{\alpha_{ijk}-1} \end{aligned} \tag{7.25}$$

[3]There are also other assumptions that need to be made about the structure of the network but we do not mention them here because we assume that the structure is fixed.

where $\sum_{k=1}^{r_i} \boldsymbol{\theta_{ijk}} = 1$ and α_{ijk} are positive constants.

To obtain the Bayes estimate, we first compute the likelihood of the sample $\mathcal{D}$ as

$$p(\mathcal{D}|\boldsymbol{\vartheta_{ij}}) = \prod_{k=1}^{r_i} (\boldsymbol{\theta_{ijk}})^{N_{ijk}}. \tag{7.26}$$

Then, the posterior distribution of $\boldsymbol{\vartheta_{ij}}$ can be computed using the Bayes rule as

$$\begin{aligned} p(\boldsymbol{\vartheta_{ij}}|\mathcal{D}) &= \frac{p(\mathcal{D}|\boldsymbol{\vartheta_{ij}})p(\boldsymbol{\vartheta_{ij}})}{p(\mathcal{D})} \\ &\propto \prod_{k=1}^{r_i} (\boldsymbol{\theta_{ijk}})^{N_{ijk}} \prod_{k=1}^{r_i} (\boldsymbol{\theta_{ijk}})^{\alpha_{ijk}-1} \\ &\propto \prod_{k=1}^{r_i} (\boldsymbol{\theta_{ijk}})^{\alpha_{ijk}+N_{ijk}-1} \\ &= \mathrm{Dir}(\boldsymbol{\vartheta_{ij}}|\alpha_{ij1} + N_{ij1}, \ldots, \alpha_{ijr_i} + N_{ijr_i}). \end{aligned} \tag{7.27}$$

The Bayes estimate for $\boldsymbol{\theta_{ijk}}$ is found by taking the conditional expected value

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{\boldsymbol{ijk}} &= E_{p(\boldsymbol{\vartheta_{ij}}|\mathcal{D})}[\boldsymbol{\theta_{ijk}}] \\ &= \int \boldsymbol{\theta_{ijk}} p(\boldsymbol{\vartheta_{ij}}|\mathcal{D}) d\boldsymbol{\vartheta_{ij}} \\ &= \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}} \end{aligned} \tag{7.28}$$

where $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ as before. α_{ij} is sometimes called the equivalent sample size for the Dirichlet distribution because it is equal to the number of observations we would have to make starting from complete ignorance in order to arrive at that distribution [91]. It represents user's confidence in the prior values of $\boldsymbol{\vartheta_{ij}}$, i.e. the larger α_{ij} is, the more certain the user is about the values. We will discuss different choices for $\alpha_{ij1}, \ldots, \alpha_{ijr_i}$ in Section 7.3.3.

Continuous Variables With Discrete Parents

Continuous variables have not been used as widely as discrete variables and a common way to handle them has been to quantize the values and use the estimation methods for discrete variables. The approaches that include continuous variables used either multivariate Gaussians [79, 129], mixtures of Gaussians [55] or Gaussian kernel estimation [111].

We presented maximum likelihood estimates (MLEs) for the parameters of some distributions in the exponential family in Section 3.3.5. One of the main advantages of using distributions from the exponential family is that the computational requirements for learning are guaranteed to be linear in the sample size [34]. After the sufficient statistics are computed, learning proceeds independently of the sample size.

As in the case of discrete variables, Bayes estimates for the parameters of continous variables can be derived using data and prior information. We consider continuous variables with only Gaussian or Gamma distributions with discrete parents in the rest of the section. Exact inference for the case of continous variables with continuous parents is also possible [79, 34, 129, 55] but exact inference for discrete variables with continous parents is not always tractable [147].

Let x_i be a continuous variable having a Gaussian distribution

$$p(x_i|\boldsymbol{\pi_i} = j, \boldsymbol{\theta_i}) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} e^{(x_i-\mu_{ij})^2/2\sigma_{ij}^2}, \; -\infty < x_i < \infty, \; -\infty < \mu_{ij} < \infty, \; \sigma_{ij} > 0 \tag{7.29}$$

for each possible state j of its parents $\boldsymbol{\pi_i}$ where $j \in \{1, \ldots, q_i\}$ and $\boldsymbol{\theta_i} = \{\mu_{ij}, \sigma_{ij}^2\}_{j=1}^{q_i}$ with the q_i possible states as described in the previous section. To find the MLEs of the parameters, we first partition the training data $\mathcal{D}$ into $\mathcal{D}_1, \ldots, \mathcal{D}_{q_i}$ according to the states of $\boldsymbol{\pi_i}$. Then, the MLEs are computed as

$$\hat{\mu}_{ij} = \frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i \qquad \text{and} \qquad \hat{\sigma}_{ij}^2 = \frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} (x_i - \hat{\mu}_{ij})^2. \tag{7.30}$$

Multivariate case is computed similarly.

However, we can have a small sample problem as in the discrete variable case. Using prior information helps avoiding overfitting and can also ensure that the covariance matrix is positive definite in the multivariate case. Conjugate priors for the parameters of a Gaussian(μ, σ^2) distribution with mean μ and variance σ^2 are given in [56, p. 169] as follows: The prior conditional distribution of μ for fixed σ is a Gaussian distribution

$$\mu|\sigma \sim \text{Gaussian}(\mu_0, \sigma^2/\tau_0),\ -\infty < \mu < \infty,\ -\infty < \mu_0 < \infty,\ \tau_0 > 0, \tag{7.31}$$

the prior marginal distribution of σ^{-2} is a Gamma distribution

$$\sigma^{-2} \sim \text{Gamma}(\alpha_0, \beta_0),\ \sigma, \alpha_0, \beta_0 > 0 \tag{7.32}$$

and the prior marginal distribution of μ is a Student-t distribution

$$\mu \sim \text{Student-}t(2\alpha_0, \mu_0, \alpha_0\beta_0\tau_0),\ -\infty < \mu < \infty,\ \alpha_0, \beta_0, \tau_0 > 0,\ -\infty < \mu_0 < \infty \tag{7.33}$$

where the Student-t distribution is defined as

$$\text{Student-}t(x|\alpha, \mu, \tau) = \frac{\tau^{1/2}\Gamma[(\alpha+1)/2]}{(\alpha\pi)^{1/2}\Gamma(\alpha/2)}\left[1 + \frac{\tau}{\alpha}(x-\mu)^2\right]^{-(\alpha+1)/2} \tag{7.34}$$

with α being the degrees of freedom, μ being the location parameter and τ being the precision parameter.

Then, given a sample $y_1, \ldots, y_n$ and the sample mean $\bar{y} = \sum_{i=1}^n y_i$, the posterior joint distribution of μ and σ^{-2} is computed as follows: The posterior conditional distribution of μ given σ is a Gaussian distribution

$$\mu|\sigma \sim \text{Gaussian}\left(\frac{\tau_0\mu_0 + n\bar{y}}{\tau_0 + n}, \frac{\sigma^2}{\tau_0 + n}\right), \tag{7.35}$$

the posterior marginal distribution of σ^{-2} is a Gamma distribution

$$\sigma^{-2} \sim \text{Gamma}\left(\alpha_0 + \frac{n}{2}, \left[\frac{1}{\beta_0} + \frac{1}{2}\sum_{i=1}^{n}(y_i - \bar{y})^2 + \frac{\tau_0 n(\bar{y} - \mu_0)^2}{2(\tau_0 + n)}\right]^{-1}\right) \tag{7.36}$$

and the posterior marginal distribution of μ is a Student-t distribution

$$\mu \sim \text{Student-}t\left(2\alpha_0 + n, \frac{\tau_0\mu_0 + n\bar{y}}{\tau_0 + n}, \frac{(\alpha_0 + \frac{n}{2})\tau_0}{\frac{1}{\beta_0} + \frac{1}{2}\sum_{i=1}^{n}(y_i - \bar{y})^2 + \frac{\tau_0 n(\bar{y} - \mu_0)^2}{2(\tau_0 + n)}}\right). \tag{7.37}$$

In the multivariate case, the conjugate conditional distribution of $\boldsymbol{\mu}$ given $\boldsymbol{\Sigma}$ is a Gaussian distribution, the conjugate marginal distribution of $\boldsymbol{\Sigma}^{-1}$ is a Wishart distribution and the conjugate marginal distribution of $\boldsymbol{\mu}$ is a multivariate Student-t distribution [56, p. 178].

Given the training data $\mathcal{D}$ partitioned into $\mathcal{D}_1, \ldots, \mathcal{D}_{q_i}$, the posterior distributions for the parameters of a continuous variable x_i can be computed using the formulas above with the corresponding subsets of the training data for each possible state of its parents $\boldsymbol{\pi_i}$. Then, the Bayes estimates of μ_{ij} and σ_{ij}^2 can be computed as the expected values of the posterior distributions (conditional expected values given data), i.e.

$$\hat{\mu}_{ij} = \frac{\tau_{ij0}\mu_{ij0} + \#\mathcal{D}_j \, \overline{x_{i|j}}}{\tau_{ij0} + \#\mathcal{D}_j} \tag{7.38}$$

and

$$\hat{\sigma}_{ij}^2 = \frac{\frac{2}{\beta_{ij0}} + \sum_{x_i \in \mathcal{D}_j} (x_i - \overline{x_{i|j}})^2 + \frac{\tau_{ij0}\,\#\mathcal{D}_j\,(\overline{x_{i|j}} - \mu_{ij0})^2}{\tau_{ij0} + \#\mathcal{D}_j}}{2\alpha_{ij0} + \#\mathcal{D}_j} \tag{7.39}$$

where $\overline{x_{i|j}} = \frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i$.

When x_i is a continuous variable having a Gamma distribution,

$$p(x_i|\boldsymbol{\pi_i} = j, \boldsymbol{\theta_i}) = \frac{1}{\Gamma(\alpha_{ij})\beta_{ij}^{\alpha_{ij}}} x_i^{\alpha_{ij}-1} e^{-x_i/\beta_{ij}} \; x_i \geq 0, \; \alpha_{ij}, \beta_{ij} > 0 \tag{7.40}$$

for each possible state j of its parents $\boldsymbol{\pi_i}$ where $j \in \{1, \ldots, q_i\}$ and $\boldsymbol{\theta_i} = \{\alpha_{ij}, \beta_{ij}\}_{j=1}^{q_i}$ with the q_i possible states as described in the previous section. After partitioning the training data as in the previous case, the method of moments estimators (MOMs) are computed as

$$\hat{\alpha}_{ij} = \frac{\left(\frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i\right)^2}{\left(\frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i^2\right) - \left(\frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i\right)^2} \tag{7.41}$$

and

$$\hat{\beta}_{ij} = \frac{\left(\frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i^2\right) - \left(\frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i\right)^2}{\left(\frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i\right)}. \tag{7.42}$$

The conjugate prior conditional distribution for β for a fixed α in a Gamma(α, β) distribution is also a Gamma distribution [34]

$$\beta^{-1}|\alpha \sim \text{Gamma}(\alpha_0, \beta_0),\ \alpha_0, \beta_0 > 0 \tag{7.43}$$

which results in the posterior conditional distribution which is also a Gamma

$$\beta^{-1}|\alpha \sim \text{Gamma}\left(\alpha n + \alpha_0, \left[\frac{1}{\beta_0} + \sum_{i=1}^{n} y_i\right]^{-1}\right) \tag{7.44}$$

where $y_1, \ldots, y_n$ is the data sample.

Given the training data $\mathcal{D}$ partitioned into $\mathcal{D}_1, \ldots, \mathcal{D}_{q_i}$, the posterior conditional distributions for the β_{ij} parameters can be computed as above and their Bayes estimates can be found as

$$\hat{\beta}_{ij}|\alpha_{ij} = \frac{\alpha_{ij}\ \#\mathcal{D}_j + \alpha_{ij0}}{\frac{1}{\beta_{ij0}} + \sum_{x_i \in \mathcal{D}_j} x_i}. \tag{7.45}$$

We will discuss how to choose the prior distribution parameters in Section 7.3.3.

7.3.2 Making Inferences

Once we know the joint probability distribution encoded in the network, we can answer all possible inference questions about the variables using marginalization. For example, the joint probability distribution for the network in Figure 7.1(a) is

$$p(\mathcal{C}, x_1, x_2, x_3) = p(c)p(x_1|\mathcal{C})p(x_2|\mathcal{C})p(x_3|\mathcal{C}) \tag{7.46}$$

where $\mathcal{C}$ = land cover class, x_1 = Gibbs features class label, x_2 = co-occurrence features class label and x_3 = spectral value class label. A useful application is to estimate the probability of a pixel belonging to the land cover class $\mathcal{C}$ given the values of the feature class labels x_1, x_2 and x_3 for that pixel. Using the Bayes rule,

$$\begin{aligned} p(\mathcal{C}|x_1, x_2, x_3) &= \frac{p(\mathcal{C}, x_1, x_2, x_3)}{p(x_1, x_2, x_3)} \\ &= \frac{p(x_1, x_2, x_3|\mathcal{C})p(\mathcal{C})}{p(x_1, x_2, x_3)} \\ &= \frac{p(x_1|\mathcal{C})p(x_2|\mathcal{C})p(x_3|\mathcal{C})p(\mathcal{C})}{p(x_1|\mathcal{C})p(x_2|\mathcal{C})p(x_3|\mathcal{C})p(\mathcal{C}) + p(x_1|\bar{\mathcal{C}})p(x_2|\bar{\mathcal{C}})p(x_3|\bar{\mathcal{C}})p(\bar{\mathcal{C}})} \end{aligned} \tag{7.47}$$

where $\bar{\mathcal{C}}$ means the pixel does not belong to the land cover class. This is called "bottom-up" inference because it goes from effects to causes.

Similarly, we can compute the joint probability distribution for the network in Figure 7.1(b) as

$$p(a,u,c,s,m,k,t) = p(a)p(u)p(c)p(s)p(m|a,u,c)p(k|u,c,s)p(t|u,c,s) \tag{7.48}$$

where the binary variables represent a = action, u = close-up, c = crowd, s = scene setting, m = motion, k = skin and t = texture. A possible application is to compute the probability of a video clip shot in a particular scene setting s given its measurements m for motion, k for skin and t for texture. Using the Bayes rule, we obtain

$$\begin{aligned} p(s|m,k,t) &= \frac{p(s,m,k,t)}{p(m,k,t)} \\ &= \frac{\sum_{a,u,c} p(a,u,c,s,m,k,t)}{\sum_{a,u,c,s} p(a,u,c,s,m,k,t)}. \end{aligned} \tag{7.49}$$

Another interesting application is to find video clips which have a high probability of having a crowd in a specific setting but not having any action given the motion, skin and texture measurements, i.e. $p(\bar{a},c,s|m,k,t)$.

7.3.3 Updating Parameters

This section discusses how to set the prior distributions and then how to update the parameters of a network using new data given its current state.

Discrete Variables With Discrete Parents

The maximum likelihood estimate in Equation (7.23) is also called the relative frequency estimate and involves only the counts of the states of discrete variables in the training data. However, as mentioned in Section 7.3.1, maximum likelihood estimates are known to be sensitive to sparse data. If we have only a few training examples, the relative frequency estimate can give extreme values and becomes unreliable. The

Bayes estimate in Equation (7.28) deals with this problem by making use of the prior information. When there are a few training examples, the prior information has more effect. When the number of training examples increases, the effect of training data starts to increase.

An intuitive choice for the hyperparameters $\alpha_{ij1}, \ldots, \alpha_{ijr_i}$ for the Dirichlet prior is to assume all r_i states to be equally probable and set $\alpha_{ijk} = 1, \forall k \in \{1, \ldots, r_i\}$. This corresponds to the prior distribution $p(\boldsymbol{\vartheta_{ij}}) = \Gamma(r_i)$ which is equivalent to the number of combinatorial subsets of the variables in $\boldsymbol{\vartheta_{ij}}$. This special case is also called the Laplace's law of succession [123] where

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \frac{1 + N_{ijk}}{r_i + N_{ij}}. \tag{7.50}$$

It is called a law of succession because it represents the conditional expectation that $\boldsymbol{\theta}_{\boldsymbol{i}jk}$ will have a particular value given its values in previous cases, i.e. training examples.

This can also be explained using the form of the Bayes estimate

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \frac{\alpha\beta + N_{ijk}}{\alpha + N_{ij}} \tag{7.51}$$

where β is the prior estimate for α_{ijk} and α is the weight given to the prior (i.e. equivalent sample size, the number of training examples required for the significance of their estimate to be the same as the significance of the prior) [144]. It can also be explained as a linear interpolation between the maximum likelihood estimate N_{ijk}/N_{ij} and the prior β as

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \left(\frac{\alpha}{\alpha + N_{ij}}\right)\beta + \left(\frac{N_{ij}}{\alpha + N_{ij}}\right)\frac{N_{ijk}}{N_{ij}}. \tag{7.52}$$

Laplace's law of succession is a special case of a Bayesian estimate starting from the uniform prior of $\beta = 1/r_i$ on $\{\alpha_{ijk}\}_{k=1}^{r_i}$ and using $\alpha = r_i$.

Another choice is the Jeffreys-Perks' law of succession

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \frac{0.5 + N_{ijk}}{0.5r_i + N_{ij}} \tag{7.53}$$

which is the special case with $\alpha = 0.5r_i$ [123]. The Laplace and Jeffreys-Perks estimates have been criticized for assigning either too much or too little probability to specific events [165] but were also decided to be safe choices when the distribution of the source is unknown and the number of possible states r_i is fixed and known [123]. Witten and Bell [210] developed a Poisson process model and got small improvements of the coding efficiency for text compression over the case where the Laplace's law of succession was used.

Yet another choice is Ristad's law of succession [165] which can be explained as follows. We consider each state to be a symbol in an alphabet. Then, a training session is a string that is a subset of symbols in the alphabet. In the first interpretation, all nonempty subsets of the alphabet are equally likely (uniform subsets prior)

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \begin{cases} \frac{1}{r_i} & \text{if} \quad t_{ij} = 0 \\ \frac{(N_{ijk}+1)(N_{ij}+1-t_{ij})}{(N_{ij}+t_{ij})(N_{ij}+1-t_{ij})+t_{ij}(r_i-t_{ij})} & \text{if} \quad N_{ijk} > 0 \\ \frac{t_{ij}}{(N_{ij}+t_{ij})(N_{ij}+1-t_{ij})+t_{ij}(r_i-t_{ij})} & \text{otherwise} \end{cases} \tag{7.54}$$

where $t_{ij} = \#\{k : N_{ijk} > 0\}$. In the second interpretation, all nonzero subset cardinalities are equally likely (uniform cardinality prior)

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{i}jk} = \begin{cases} \frac{1+N_{ijk}}{r_i+N_{ij}} & \text{if} \quad t_{ij} = 0 \quad \text{or} \quad t_{ij} = r_i \\ \frac{(N_{ijk}+1)(N_{ij}+1-t_{ij})}{N_{ij}^2+N_{ij}+2t_{ij}} & \text{if} \quad t_{ij} < r_i \quad \text{and} \quad N_{ijk} > 0 \cdot \\ \frac{t_{ij}(t_{ij}+1)}{(r_i-t_{ij})(N_{ij}^2+N_{ij}+2t_{ij})} & \text{otherwise} \end{cases} \tag{7.55}$$

According to the uniform subsets prior, both very large and very small subsets are relatively improbable. On the other hand, the uniform cardinality prior assigns more probability to small and large subsets of the alphabet and less to subsets of moderate cardinality. Both laws reduce to Laplace's law of succession when all the states are attested, i.e. $t_{ij} = r_i$ in our case.

Laplace's law of succession and Jeffreys-Perks' law of succession still have the influence of the priors a lot if we do not have examples for some of the states. Ristad's law of succession with uniform cardinality prior allocates more probability to unseen states than the method with uniform subsets prior. When estimating the probability of a specific state, Ristad's laws of succession with uniform subsets and uniform cardinality priors take into account the examples for other states of the same variable as well. However, Laplace's law of succession and Jeffreys-Perks' law of succession just count the examples for that particular state.

Given the current state of the network that was trained using the prior information and the sample $\mathcal{D}$, we can easily update the parameters when new data $\mathcal{D}'$ is available. The new posterior distribution for $\boldsymbol{\vartheta_{ij}}$ becomes

$$p(\boldsymbol{\vartheta_{ij}}|\mathcal{D},\mathcal{D}') = \frac{p(\mathcal{D}'|\boldsymbol{\vartheta_{ij}})p(\boldsymbol{\vartheta_{ij}}|\mathcal{D})}{p(\mathcal{D}'|\mathcal{D})}. \tag{7.56}$$

With the Dirichlet priors and the posterior distribution for $p(\boldsymbol{\vartheta_{ij}}|\mathcal{D})$ given in Equation (7.27), the updated posterior distribution becomes

$$p(\boldsymbol{\vartheta_{ij}}|\mathcal{D},\mathcal{D}') = \text{Dir}(\boldsymbol{\vartheta_{ij}}|\alpha_{ij1} + N_{ij1} + N'_{ij1}, \ldots, \alpha_{ijr_i} + N_{ijr_i} + N'_{ijr_i}) \tag{7.57}$$

where N'_{ijk} is the number of cases in $\mathcal{D}'$ in which $x_i = k$ and $\boldsymbol{\pi_i} = j$. This is equivalent to defining a new prior distribution $\text{Dir}(\boldsymbol{\vartheta_{ij}}|\alpha'_{ij1}, \ldots, \alpha'_{ijr_i})$ with $\alpha'_{ijk} = \alpha_{ijk} + N_{ijk}, k = 1, \ldots, r_i$, and using this new prior to compute the updated posterior distribution from the new data using Equation (7.27). Hence, updating the network parameters involves only updating the counts in the estimates for $\hat{\boldsymbol{\theta}}_{\boldsymbol{ijk}}$.

Continuous Variables With Discrete Parents

A straightforward way of choosing the parameters for the prior distributions for continuous variables is to specify prior means and variances. When the continuous variable x_i is assumed to have a Gaussian$(\mu_{ij}, \sigma^2_{ij})$ distribution for each possible state j of its parents, we start by assuming prior means and variances to its parameters as $E[\mu_{ij}]$, $\text{var}(\mu_{ij})$, $E[\sigma^{-2}_{ij}]$ and $\text{var}(\sigma^{-2}_{ij})$. Then, the parameters for the prior distributions for μ_{ij} and σ_{ij}^{-2} can be computed as

$$\mu_{ij0} = E[\mu_{ij}], \tag{7.58}$$

$$\alpha_{ij0} = \frac{E^2[\sigma_{ij}^{-2}]}{\mathrm{var}(\sigma_{ij}^{-2})}, \tag{7.59}$$

$$\beta_{ij0} = \frac{\mathrm{var}(\sigma_{ij}^{-2})}{E[\sigma_{ij}^{-2}]}, \tag{7.60}$$

$$\tau_{ij0} = \frac{1}{(\alpha_{ij0} - 1)\beta_{ij0}\mathrm{var}(\mu_{ij})}. \tag{7.61}$$

After being given the data sample, the posterior distributions and Bayes estimates for μ_{ij} and σ_{ij}^2 can be computed using the formulas in Section 7.3.1. The Bayes estimate for μ_{ij} can be written as a linear combination of the prior mean and the maximum likelihood estimate as

$$\hat{\mu}_{ij} = \left(\frac{\tau_{ij0}}{\tau_{ij0} + \#\mathcal{D}_j}\right)\mu_{ij0} + \left(\frac{\#\mathcal{D}_j}{\tau_{ij0} + \#\mathcal{D}_j}\right)\overline{x_{i|j}} \tag{7.62}$$

where τ_{ij0} acts as the equivalent sample size. When there are a few training examples, the prior information has more effect. When the number of training examples increases, the effect of training data starts to increase.

Given the current state of the network that was trained using the prior information and the sample $\mathcal{D}$, the parameters can be easily updated using new data $\mathcal{D}'$ as

$$\hat{\mu}_{ij} = \frac{\tau_{ij0}\mu_{ij0} + \#(\mathcal{D}_j \cup \mathcal{D}'_j)\,\overline{x_{i|j}}}{\tau_{ij0} + \#(\mathcal{D}_j \cup \mathcal{D}'_j)} \tag{7.63}$$

and

$$\hat{\sigma}_{ij}^2 = \frac{\frac{2}{\beta_{ij0}} + \sum_{x_i \in \mathcal{D}_j \cup \mathcal{D}'_j}(x_i - \overline{x_{i|j}})^2 + \frac{\tau_{ij0}\,\#(\mathcal{D}_j \cup \mathcal{D}'_j)\,(\overline{x_{i|j}} - \mu_{ij0})^2}{\tau_{ij0} + \#(\mathcal{D}_j \cup \mathcal{D}'_j)}}{2\alpha_{ij0} + \#(\mathcal{D}_j \cup \mathcal{D}'_j)} \tag{7.64}$$

where $\overline{x_{i|j}} = \frac{1}{\#(\mathcal{D}_j \cup \mathcal{D}'_j)}\sum_{x_i \in \mathcal{D}_j \cup \mathcal{D}'_j} x_i$. A more efficient method is to first update the parameters μ_{ij0}, α_{ij0}, β_{ij0} and τ_{ij0} as

$$\mu'_{ij0} = \frac{\tau_{ij0}\mu_{ij0} + \#\mathcal{D}_j \, \overline{x_{i|j}}}{\tau_{ij0} + \#\mathcal{D}_j}, \tag{7.65}$$

$$\alpha'_{ij0} = \alpha_{ij0} + \frac{\#\mathcal{D}_j}{2}, \tag{7.66}$$

$$\beta'_{ij0} = \left[\frac{1}{\beta_{ij0}} + \frac{1}{2} \sum_{x_i \in \mathcal{D}_j} (x_i - \overline{x_{i|j}})^2 + \frac{\tau_{ij0} \, \#\mathcal{D}_j \, (\overline{x_{i|j}} - \mu_{ij0})^2}{2(\tau_{ij0} + \#\mathcal{D}_j)} \right]^{-1}, \tag{7.67}$$

$$\tau'_{ij0} = \tau_{ij0} + \#\mathcal{D}_j \tag{7.68}$$

after each observation set $\mathcal{D}$ where $\overline{x_{i|j}} = \frac{1}{\#\mathcal{D}_j} \sum_{x_i \in \mathcal{D}_j} x_i$, and then use the new values as priors for the following case where $\mathcal{D}'$ is observed. Then, the Bayes estimates for μ_{ij0} and β_{ij0} can be computed using the formulas in Equations (7.38) and (7.39) respectively.

7.4 Combining Features and Similarity Measures

7.4.1 Combination in Combined Classifiers Framework

We described nine classifiers in Section 4.4 and described six methods to combine the decisions made by individual classifiers in Section 7.2. As well as assigning its input to either the relevance or the irrelevance class, each classifier also outputs the strength of its decision in terms of posterior probabilities computed from its measurement vector as shown in Figure 4.2. Using the notation and definitions in Section 4.4, the straightforward extension of Figure 4.2 in the combined classifiers framework becomes:

1. Combine $\mathcal{Z}_1 = \{\mathbf{x}_{\boldsymbol{ijk}}, 1 \leq k \leq K\}$, the outputs of all classifiers for a particular feature vector i and similarity model j,

2. Combine $\mathcal{Z}_2 = \{\mathbf{x}_{\boldsymbol{ijk}}, 1 \leq i \leq I, 1 \leq j \leq J\}$, the outputs of classifier k for all feature vectors and similarity models,

3. Combine $\mathcal{Z}_3 = \{\mathbf{x}_{\boldsymbol{ijk}}, 1 \leq i \leq I, 1 \leq j \leq J, 1 \leq k \leq K\}$, the outputs of all classifiers for all feature vectors and similarity models.

Given the posterior probabilities computed by a combined classifier model, the final similarity is measured using the posterior ratio

$$\Delta(\mathcal{Z}_l) = \frac{p(\mathcal{A}|\mathcal{Z}_l)}{p(\mathcal{B}|\mathcal{Z}_l)}, \quad 1 \leq l \leq 3. \tag{7.69}$$

Under the assumption of equal priors, the posterior probability for the relevance class $\mathcal{A}$ in Equation (7.69) is computed for each classifier combination rule in Section 7.2 as follows:

1. Product rule:
$$p(\mathcal{A}|\mathcal{Z}_l) = \alpha \prod_{\mathbf{x} \in \mathcal{Z}_l} p(\mathcal{A}|\mathbf{x}), \tag{7.70}$$

2. Sum rule:
$$p(\mathcal{A}|\mathcal{Z}_l) = \alpha \sum_{\mathbf{x} \in \mathcal{Z}_l} p(\mathcal{A}|\mathbf{x}), \tag{7.71}$$

3. Max rule:
$$p(\mathcal{A}|\mathcal{Z}_l) = \alpha \max_{\mathbf{x} \in \mathcal{Z}_l} p(\mathcal{A}|\mathbf{x}), \tag{7.72}$$

4. Min rule:
$$p(\mathcal{A}|\mathcal{Z}_l) = \alpha \min_{\mathbf{x} \in \mathcal{Z}_l} p(\mathcal{A}|\mathbf{x}), \tag{7.73}$$

5. Median rule:
$$p(\mathcal{A}|\mathcal{Z}_l) = \alpha \operatorname*{median}_{\mathbf{x} \in \mathcal{Z}_l} p(\mathcal{A}|\mathbf{x}) \tag{7.74}$$

   where α is the normalization factor, and

6. Majority vote rule:
$$p(\mathcal{A}|\mathcal{Z}_l) = \frac{\sum_{\mathbf{x} \in \mathcal{Z}_l} \delta_{\mathcal{A}\mathbf{x}} + 1}{\#\mathcal{Z}_l + 2} \tag{7.75}$$

i.e. a Bayes estimate using Laplace's prior (see Section 7.3.3) and the votes from all classifiers where $\delta_{\mathcal{A}\mathbf{x}}$ is the vote for class $\mathcal{A}$ from the classifier with the measurement vector $\mathbf{x}$.

The posterior probability for the irrelevance class $\mathcal{B}$ is computed analogously. Classification and retrieval performances for the combined classifiers framework are given in Sections 8.2 and 8.3.

7.4.2 Combination in Bayesian Network Framework

We showed that the class-conditional and posterior probabilities are very effective in retrieval. However, the probabilistic models are estimated from noisy data so their outputs are also noisy. This uncertainty is already incorporated as "probability of probability" in the two-level modeling in the classification framework. Furthermore, a Bayesian network can be considered as another classifier that can handle complex relationships of its inputs.

Assume that the system is using a combination of n of the feature vectors, similarity models and classifiers described in the previous chapters. Each model measures the relevancy of a database image with respect to the query image. Denote the final measurements by these models as $x_1, \ldots, x_n$, which can be likelihood ratio values, distance values or probabilities. Each of these measurements will also have an associated probability distribution. The joint posterior probability for the relevance class is $p(\mathcal{A}|x_1, \ldots, x_n)$ and the joint posterior probability for the irrelevance class is $p(\mathcal{B}|x_1, \ldots, x_n)$. Using the conditional independence assumption for the model outputs, the probability that a database image is relevant to the query image becomes

$$\begin{aligned} p(\mathcal{A}|x_1, \ldots, x_n) &= \frac{p(x_1, \ldots, x_n|\mathcal{A})p(\mathcal{A})}{p(x_1, \ldots, x_n)} \\ &= \frac{p(\mathcal{A}) \prod_{i=1}^n p(x_i|\mathcal{A})}{p(\mathcal{A}) \prod_{i=1}^n p(x_i|\mathcal{A}) + p(\mathcal{B}) \prod_{i=1}^n p(x_i|\mathcal{B})} \end{aligned} \tag{7.76}$$

and the probability that they are irrelevant becomes

$$\begin{aligned} p(\mathcal{B}|x_1,\ldots,x_n) &= \frac{p(x_1,\ldots,x_n|\mathcal{B})p(\mathcal{B})}{p(x_1,\ldots,x_n)} \\ &= \frac{p(\mathcal{B})\prod_{i=1}^n p(x_i|\mathcal{B})}{p(\mathcal{A})\prod_{i=1}^n p(x_i|\mathcal{A}) + p(\mathcal{B})\prod_{i=1}^n p(x_i|\mathcal{B})}. \end{aligned} \tag{7.77}$$

The conditional independence assumption reduces the Bayesian network into a product rule like in Section 7.2. Here, the input values $x_1,\ldots,x_n$ are the scalar random variables for the likelihood ratio values output by the probabilistic similarity models and the distance values output by the geometric models. The final measure for similarity between images is the combined posterior ratio

$$\Delta = \frac{p(\mathcal{A}|x_1,\ldots,x_n)}{p(\mathcal{B}|x_1,\ldots,x_n)}. \tag{7.78}$$

The resulting network is given in Figure 7.2. The root node c is a binary variable representing whether two images belong to the relevance class or not, i.e. $c = \{1,0\} \equiv \{\mathcal{A},\mathcal{B}\}$. Leaf nodes represent the model outputs. The class-conditional probabilities for the measurements $x_1,\ldots,x_n$ and the marginal probabilities for the classes $\mathcal{A}$ and $\mathcal{B}$ can be estimated using the models described in Section 7.3.1. Experiments are presented in Section 8.3.

7.5 Relevance Feedback

All of the similarity measures up to this point were based on the original query image. The Bayesian framework can also be extended to the case when feedback from the user is available. Given the original query feature vector and feature vectors for the images in the database, initial search is done by computing the feature difference vectors between the query image and all images in the database. Then, each image in the database can be ranked according to the posterior ratios

$$\Delta = \frac{p(\mathcal{A}|\xi^{(0)})}{p(\mathcal{B}|\xi^{(0)})} \tag{7.79}$$

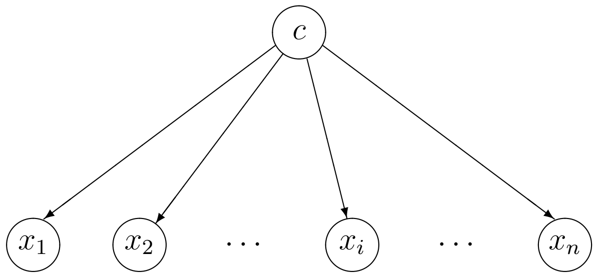


Figure 7.2: The naive Bayesian network structure to combine multiple similarity measures. The root node c is a binary variable representing whether two images belong to the relevance class or not, and the leaf nodes are the outputs of the models that measure the relevancy of database images with respect to the query image. Details of the network are described in the text.

where $\xi^{(0)}$ represents the measurements based on the initial query image and can be computed using any of the models described earlier.

1. Positive feedback:

   When the user labels an image as relevant, new feature difference vectors between the labeled image and all images in the database are computed. Then, using the class-conditional probabilities that are based on these feature difference vectors, each similarity model outputs the posterior probabilities and the images are ranked according to the updated posterior ratios

$$\Delta = \frac{p(\mathcal{A}|\xi^{(0)}, \xi_+^{(1)})}{p(\mathcal{B}|\xi^{(0)}, \xi_+^{(1)})} = \frac{p(\xi_+^{(1)}|\mathcal{A})p(\mathcal{A}|\xi^{(0)})}{p(\xi_+^{(1)}|\mathcal{B})p(\mathcal{B}|\xi^{(0)})} \tag{7.80}$$

   where $\xi_+^{(1)}$ represents the new measurements based on the first positive feedback image. Given a sequence of n images labeled as relevant, the updated posteriors are incrementally computed using the conditional independence assumption as

$$p(\mathcal{A}|\xi^{(0)}, \xi_+^{(1)}, \ldots, \xi_+^{(n)}) \propto p(\xi_+^{(n)}|\mathcal{A})p(\mathcal{A}|\xi^{(0)}, \xi_+^{(1)}, \ldots, \xi_+^{(n-1)}) \tag{7.81}$$

   and

$$p(\mathcal{B}|\xi^{(0)}, \xi_+^{(1)}, \ldots, \xi_+^{(n)}) \propto p(\xi_+^{(n)}|\mathcal{B})p(\mathcal{B}|\xi^{(0)}, \xi_+^{(1)}, \ldots, \xi_+^{(n-1)}) \tag{7.82}$$

where $\xi_+^{(n)}$ represents the measurements based on the n'th positive feedback image.

2. Negative feedback:

   When the user labels an image as irrelevant, search proceeds by computing new feature difference vectors as above but the posteriors are updated differently. The strength of the evidence of two images being relevant is a negative evidence that they are irrelevant. Therefore, the likelihood of an image being relevant to a negative example also represents its likelihood of being irrelevant to the user's desired image. Given the first image labeled as irrelevant by the user, the posteriors are updated as

$$p(\mathcal{A}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)},\xi_-^{(1)}) \propto p(\xi_-^{(1)}|\mathcal{B})p(\mathcal{A}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)}) \tag{7.83}$$

   and

$$p(\mathcal{B}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)},\xi_-^{(1)}) \propto p(\xi_-^{(1)}|\mathcal{A})p(\mathcal{B}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)}) \tag{7.84}$$

   where $\xi_-^{(1)}$ represents the measurements based on the first negative feedback image. Given a sequence of m images labeled as irrelevant, the updated posteriors are incrementally computed using the conditional independence assumption as

$$p(\mathcal{A}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)},\xi_-^{(1)},\ldots,\xi_-^{(m)}) \propto \\ p(\xi_-^{(m)}|\mathcal{B})p(\mathcal{A}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)},\xi_-^{(1)},\ldots,\xi_-^{(m-1)}) \tag{7.85}$$

   and

$$p(\mathcal{B}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)},\xi_-^{(1)},\ldots,\xi_-^{(m)}) \propto \\ p(\xi_-^{(m)}|\mathcal{A})p(\mathcal{B}|\xi^{(0)},\xi_+^{(1)},\ldots,\xi_+^{(n)},\xi_-^{(1)},\ldots,\xi_-^{(m-1)}). \tag{7.86}$$

Experiments are presented in Section 8.5. Vasconcelos and Lippman [205] and Cox *et al.* [49] proposed similar feedback algorithms with the conditional independence assumptions but the former paper used only one kind of feature vector and the latter paper used only positive feedback with feature vector combination performed as a weighted sum of L_1 distances. Furthermore, the former paper used as many classes as there are images in the database and having too many classes caused estimation problems for the likelihood based on negative examples.

Chapter 8

EXPERIMENTS

Experiments for testing individual algorithms were already presented in their corresponding chapters. In this chapter, we first give a short summary of our system components. Then, the rest of the chapter presents extensive experiments for both classification and retrieval algorithms proposed in Chapter 7. Comparative tests using two algorithms from the literature are also presented. In all of the experiments, we use independent training (approximately one-third of the whole data) and testing (approximately two-thirds of the whole data) sets as described in Section 1.5.2. Results show that the Bayesian framework gives significant performance improvements and performs more robustly and much more powerful than the competing algorithms.

8.1 System Components

1. Databases: Details of three groundtruth databases were given in Section 1.5.1. These databases are:

    (a) ISL Database: 600 aerial and satellite images divided into 7 categories.

    (b) VisTex Database: 736 texture images divided into 46 categories.

    (c) COREL Database: 1575 stock photo images including nature, animals, buildings, cars and airplanes divided into 18 categories.

2. Feature level: Descriptions of the features were given in Section 3.2. Each image is associated with the following q-dimensional feature vectors:

    (a) Line-angle-ratio statistics (LAR) ($q = 20$)

   (b) Co-occurrence variances (COOC) ($q = 20$)

   (c) Gabor features (GABOR) ($q = 60$)

   (d) Moments features (MOMENTS) ($q = 36$)

   (e) Tamura features (TAMURA) ($q = 4$)

   (f) Color histograms (COLHIST) ($q = 27$ for the VisTex Database and 64 for the COREL Database)

3. Similarity level: Given the feature vectors for a pair of images, similarity measures compute a value that can be used to rank each image in the database according to its similarity to the query image. These values have a distance-like interpretation, i.e. a smaller value means that that image is more similar to the query image than another image with a larger value. Our system includes both probabilistic and geometric similarity measures and also supports combinations of multiple measurements for similarity:

   (a) Likelihood ratio with multivariate Gaussian (MVG)
   This measure uses likelihood ratios that are derived from a Bayesian classifier that measure the relevancy of two images, one being the query image and one being a database image, so that image pairs which have a high likelihood value are classified as "relevant" and the ones which have a lower likelihood value are classified as "irrelevant". A multivariate Gaussian for each class is used to estimate the class-conditional probabilities in the likelihood ratio.

   (b) Likelihood ratio with independently fitted distributions (FIT)
   This measure uses the same idea as the previous one but uses independently fitted distributions for each component in the class-conditional densities.

   (c) Likelihood ratio with mixtures of Gaussians (GMIX)
   This measure uses the same idea as the previous ones but uses mixtures of

Gaussians for the class-conditional densities.

(d) Minkowsky L_p metric (Lp)

This is the classical L_p metric. We use a classification-based approach with a minimum error decision rule to select the best performing p for the L_p metric for our datasets.

(e) Combined classifiers

Outputs of different classifiers trained on different combinations of feature vectors and class-conditional models are combined to compute the similarities between images based on multiple measurements.

(f) Bayesian network

A naive Bayesian network is used to have one more level of fusion of the probabilities of likelihood ratio values, distance values or class-conditional probabilities.

4. Post-processing level: Post-processing algorithms address the problem of retrieving images that are quite irrelevant to the query image simply because they are close to it in the feature space, and also support relevance feedback. Our system includes the following post-processing algorithms:

   (a) Graph-theoretic clustering (GTC)

   This graph-theoretic approach formulates the database search as a problem of finding the cliques of a graph that is constructed from the top-ranked results of successive queries. The "best" clique (according to the criteria described in Chapter 5) is returned as the final set of relevant images to the query. Each image in the database is also assigned a probability of its being similar to the query image.

   (b) Relevance feedback with weighted distances

   This post-processing method uses a weighted distance approach with two ways of weight updating. In the first one, weights are the ratios of standard deviations of the features both for the whole database and also among the images selected as relevant by the user. In the second one, weight updating problem is formulated as a regression problem where the optimal weights are found using different least-squares methods. The distance values are then used as similarity measures to rank images in the database. An iterative approach is used to update the weights to find new sets of relevant images.

(c) Bayesian relevance feedback
Posterior probabilities in the probabilistic similarity measures are updated using the positive and negative relevance feedback information according to the Bayes' formula.

8.2 Classification Performance

Classifier combination methods of Section 7.2 were used to perform classification experiments in the framework described in Section 7.4.1. The following tables present classification errors for training and testing datasets using a combination of n classifiers for different databases. The smallest classification errors for each case are marked with boxes.

Table 8.1: Outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean) for a particular feature vector and similarity model were combined for the ISL Database ($n = 5$).

Table 8.2: Outputs of a particular classifier for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA) and all similarity models (MVG, FIT) were combined for the ISL Database ($n = 8$).

Table 8.3: Outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's

linear, Logistic linear, scaled nearest mean, neural network) for a particular feature vector and similarity model were combined for the VisTex Database ($n = 6$).

Table 8.4: Outputs of a particular classifier for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA, COLHIST) and all similarity models (MVG, FIT) were combined for the VisTex Database ($n = 10$).

Table 8.5: Outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean) for a particular feature vector and similarity model were combined for the COREL Database ($n = 5$).

Table 8.6: Outputs of a particular classifier for all feature vectors (LAR+COOC, GABOR, MOMENTS, COLHIST) and all similarity models (MVG, FIT) were combined for the COREL Database ($n = 8$).

Table 8.7: Outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean) for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA) and all similarity models (MVG, FIT) were combined for the ISL Database ($n = 40$); outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean, neural network) for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA, COLHIST) and all similarity models (MVG, FIT) were combined for the VisTex Database ($n = 60$); and outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean) for all feature vectors (LAR+COOC, GABOR, MOMENTS, COLHIST) and all similarity models (MVG, FIT) were combined for the COREL Database ($n = 40$).

Table 8.1: Classification performance in terms of classification error for training and testing datasets by combining the outputs of all classifiers for a particular feature vector and similarity model for the ISL Database. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

		Classification error in probability space					
		MVG		FIT		GMIX	
Feature	Classifier	Training	Testing	Training	Testing	Training	Testing
LAR+COOC	Product rule	0.084762	0.187910	0.319048	0.313506	0.242302	0.326281
	Sum rule	0.085397	0.188571	0.318968	0.313459	0.242222	0.325691
	Max rule	$\boxed{0.077857}$	$\boxed{0.181015}$	0.322937	0.314569	0.247857	0.329067
	Min rule	$\boxed{0.077857}$	$\boxed{0.181015}$	0.322937	0.314569	0.247857	0.329067
	Median rule	0.098810	0.197851	$\boxed{0.314841}$	$\boxed{0.312893}$	$\boxed{0.238333}$	$\boxed{0.324557}$
	Maj. vote rule	0.098810	0.197851	$\boxed{0.314841}$	$\boxed{0.312893}$	$\boxed{0.238333}$	$\boxed{0.324557}$
GABOR	Product rule	0.019286	$\boxed{0.086068}$	0.250635	0.251570	0.025079	$\boxed{0.091712}$
	Sum rule	0.025079	0.088170	0.250794	0.251523	0.029365	0.092822
	Max rule	$\boxed{0.016746}$	0.092774	0.250476	0.254309	$\boxed{0.022143}$	0.096128
	Min rule	$\boxed{0.016746}$	0.092774	0.250476	0.254309	$\boxed{0.022143}$	0.096128
	Median rule	0.054921	0.108571	$\boxed{0.245952}$	$\boxed{0.247981}$	0.056905	0.111854
	Maj. vote rule	0.054921	0.108571	$\boxed{0.245952}$	$\boxed{0.247981}$	0.056905	0.111854
MOMENTS	Product rule	0.140159	$\boxed{0.219504}$	$\boxed{0.250238}$	0.257450	0.175317	0.271830
	Sum rule	0.140556	0.219811	$\boxed{0.250238}$	0.256104	$\boxed{0.149841}$	$\boxed{0.235372}$
	Max rule	$\boxed{0.135317}$	0.220897	0.253175	0.261346	0.181032	0.276151
	Min rule	$\boxed{0.135317}$	0.220897	0.253175	0.261346	0.181032	0.276151
	Median rule	0.154524	0.224935	0.251746	$\boxed{0.255679}$	0.173810	0.270697
	Maj. vote rule	0.154524	0.224935	0.251746	$\boxed{0.255679}$	0.173810	0.270697
TAMURA	Product rule	0.270159	0.287438	$\boxed{0.272302}$	$\boxed{0.289728}$	0.275397	0.292727
	Sum rule	0.269762	0.287934	0.274444	0.292515	0.277302	0.294404
	Max rule	0.269048	$\boxed{0.283943}$	0.276825	0.291523	$\boxed{0.268968}$	$\boxed{0.287532}$
	Min rule	0.269048	$\boxed{0.283943}$	0.276825	0.291523	$\boxed{0.268968}$	$\boxed{0.287532}$
	Median rule	$\boxed{0.268730}$	0.288028	0.282063	0.295348	0.290635	0.310484
	Maj. vote rule	$\boxed{0.268730}$	0.288028	0.282063	0.295348	0.290635	0.310484

Table 8.2: Classification performance in terms of classification error for training and testing datasets by combining the outputs of a particular classifier for all feature vectors and similarity models for the ISL Database. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

	Classification error in probability space					
	Product rule		Sum rule		Max rule	
Database	Training	Testing	Training	Testing	Training	Testing
Gaussian linear classifier	0.060159	$\boxed{0.106942}$	0.082302	0.122527	$\boxed{0.041349}$	0.108028
Gaussian quadratic classifier	$\boxed{0.023254}$	$\boxed{0.088052}$	0.040873	0.095159	0.032222	0.101960
Fisher's linear classifier	0.060159	$\boxed{0.106942}$	0.082302	0.122527	$\boxed{0.041349}$	0.108028
Logistic linear classifier	0.020794	$\boxed{0.079764}$	0.047460	0.098347	$\boxed{0.014048}$	0.085148
Scaled nearest mean classifier	$\boxed{0.198333}$	0.277875	0.203413	0.256954	0.199444	0.303400
	Min rule		Median rule		Maj. vote rule	
	Training	Testing	Training	Testing	Training	Testing
Gaussian linear classifier	$\boxed{0.041349}$	0.108028	0.118810	0.154758	0.115397	0.145502
Gaussian quadratic classifier	0.032222	0.101960	0.117460	0.158772	0.108254	0.150035
Fisher's linear classifier	$\boxed{0.041349}$	0.108028	0.118810	0.154758	0.115397	0.145502
Logistic linear classifier	$\boxed{0.014048}$	0.085148	0.093730	0.133908	0.095159	0.136080
Scaled nearest mean classifier	0.199444	0.303400	0.230794	$\boxed{0.246328}$	0.226349	0.248335

Using combined classifiers usually did not improve classification performance when classifiers for a particular feature vector and similarity model were used (Tables 8.1, 8.3, 8.5). This is consistent with other results [119, 120, 67] that using different classifiers for the same feature vector often violates the conditional independence assumptions in the derivations of the combination rules. On the other hand, combining the outputs of a particular classifier for different feature vectors and similarity models highly improved the results of the experiments without combination (Tables 8.2, 8.4, 8.6), which is the actual goal of our combination framework.

The most successful combination rule was the product rule when multiple feature vectors were used. The sum, max and min rules were also successful. The most successful classifiers were again the Logistic linear and Gaussian quadratic classifiers. Combining the outputs of all classifiers for all feature vectors and all similarity models did not give much improvement and is not worth the heavy computation (Table 8.7).

Q statistics values for all feature vectors, similarity models and classifiers (as described above for different databases) were computed using Equation (7.16) and

Table 8.3: Classification performance in terms of classification error for training and testing datasets by combining the outputs of all classifiers for a particular feature vector and similarity model for the VisTex Database. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

		Classification error in probability space					
		MVG		FIT		GMIX	
Feature	Classifier	Training	Testing	Training	Testing	Training	Testing
LAR+COOC	Product rule	0.129529	0.168043	0.186896	0.197609	0.151268	0.203478
	Sum rule	0.130435	0.171630	0.187198	0.197174	0.150060	0.200761
	Max rule	0.128623	$\boxed{0.162935}$	$\boxed{0.184481}$	$\boxed{0.197065}$	0.226751	0.278804
	Min rule	0.128623	$\boxed{0.162935}$	$\boxed{0.184481}$	$\boxed{0.197065}$	0.226751	0.278804
	Median rule	0.153986	0.191957	0.195048	0.200109	0.146739	$\boxed{0.197174}$
	Maj. vote rule	$\boxed{0.126510}$	0.177826	0.189312	0.201196	$\boxed{0.146135}$	0.199891
GABOR	Product rule	0.010568	$\boxed{0.060761}$	0.081824	0.090217	0.015097	$\boxed{0.064239}$
	Sum rule	0.013587	0.061304	0.082729	0.091739	0.020531	0.064457
	Max rule	0.009360	0.075000	0.075483	0.081848	0.012681	0.064565
	Min rule	0.009360	0.075000	0.075483	0.081848	0.012681	0.064565
	Median rule	0.022947	0.061087	0.097524	0.102826	0.030495	0.064674
	Maj. vote rule	$\boxed{0.008454}$	0.075978	$\boxed{0.072464}$	$\boxed{0.081087}$	$\boxed{0.011171}$	0.071957
MOMENTS	Product rule	0.131643	0.161087	0.186896	0.183370	0.138889	0.175543
	Sum rule	0.131944	0.163478	0.187198	0.183587	0.140097	0.178913
	Max rule	$\boxed{0.122283}$	$\boxed{0.157391}$	$\boxed{0.179952}$	$\boxed{0.180000}$	$\boxed{0.121377}$	$\boxed{0.165870}$
	Min rule	$\boxed{0.122283}$	$\boxed{0.157391}$	$\boxed{0.179952}$	$\boxed{0.180000}$	$\boxed{0.121377}$	$\boxed{0.165870}$
	Median rule	0.140700	0.170435	0.194143	0.194457	0.146437	0.187174
	Maj. vote rule	0.131341	0.165543	0.183273	0.182500	0.137077	0.181739
TAMURA	Product rule	0.182065	0.182283	0.174215	0.172283	$\boxed{0.175423}$	$\boxed{0.172174}$
	Sum rule	0.182367	0.182500	0.174215	0.173370	$\boxed{0.175423}$	$\boxed{0.172174}$
	Max rule	$\boxed{0.177234}$	0.178804	0.174215	0.172065	0.175725	0.172609
	Min rule	$\boxed{0.177234}$	0.178804	0.174215	0.172065	0.175725	0.172609
	Median rule	0.190217	0.189130	0.176932	0.173696	0.177234	0.175109
	Maj. vote rule	0.180857	$\boxed{0.176522}$	$\boxed{0.169988}$	$\boxed{0.170435}$	0.176027	0.177065
COLHIST	Product rule	0.054952	0.066413	0.118659	0.121957	0.069746	0.075870
	Sum rule	0.054348	0.065326	0.118961	0.121413	0.071558	0.078804
	Max rule	0.049517	0.065870	0.113829	0.117283	$\boxed{0.060688}$	$\boxed{0.070435}$
	Min rule	0.049517	0.065870	0.113829	0.117283	$\boxed{0.060688}$	$\boxed{0.070435}$
	Median rule	0.057669	0.067283	0.125000	0.125000	0.088768	0.094022
	Maj. vote rule	$\boxed{0.047705}$	$\boxed{0.061413}$	$\boxed{0.110809}$	$\boxed{0.113370}$	0.086353	0.090435

Table 8.4: Classification performance in terms of classification error for training and testing datasets by combining the outputs of a particular classifier for all feature vectors and similarity models for the VisTex Database. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

Database	Classification error in probability space					
	Product rule		Sum rule		Max rule	
	Training	Testing	Training	Testing	Training	Testing
Gaussian linear classifier	0.060688	0.056630	0.092995	0.085761	$\boxed{0.041063}$	$\boxed{0.055000}$
Gaussian quadratic classifier	$\boxed{0.009662}$	0.052500	0.022947	$\boxed{0.034130}$	0.034420	0.101087
Fisher's linear classifier	0.060688	0.056630	0.092995	0.085761	$\boxed{0.041063}$	$\boxed{0.055000}$
Logistic linear classifier	$\boxed{0.007246}$	$\boxed{0.031087}$	0.022041	0.034022	0.008756	0.056087
Scaled nearest mean classifier	$\boxed{0.078200}$	$\boxed{0.086196}$	0.091787	0.088043	0.084239	0.118587
Nearest neighbor classifier	$\boxed{0.000000}$	$\boxed{0.059783}$	$\boxed{0.000000}$	0.066848	$\boxed{0.000000}$	0.066739
Parzen classifier	0.001812	0.077283	0.008454	$\boxed{0.043370}$	$\boxed{0.001510}$	0.092609
Binary decision tree classifier	$\boxed{0.000000}$	$\boxed{0.059674}$	$\boxed{0.000000}$	0.063261	$\boxed{0.000000}$	0.090543
Neural network classifier	$\boxed{0.015097}$	$\boxed{0.033152}$	0.016908	0.034348	0.015399	0.049239
	Min rule		Median rule		Maj. vote rule	
	Training	Testing	Training	Testing	Training	Testing
Gaussian linear classifier	$\boxed{0.041063}$	$\boxed{0.055000}$	0.119565	0.115652	0.104469	0.097283
Gaussian quadratic classifier	0.034420	0.101087	0.050423	0.053804	0.045894	0.047826
Fisher's linear classifier	$\boxed{0.041063}$	$\boxed{0.055000}$	0.119565	0.115652	0.104469	0.097283
Logistic linear classifier	0.008756	0.056087	0.037440	0.043261	0.035628	0.045870
Scaled nearest mean classifier	0.084239	0.118587	0.130133	0.121413	0.112319	0.109130
Nearest neighbor classifier	$\boxed{0.000000}$	0.066739	$\boxed{0.000000}$	0.074239	$\boxed{0.000000}$	0.091957
Parzen classifier	$\boxed{0.001510}$	0.089239	0.017814	0.048261	0.019324	0.054239
Binary decision tree classifier	$\boxed{0.000000}$	0.090543	$\boxed{0.000000}$	0.070000	$\boxed{0.000000}$	0.093804
Neural network classifier	0.015399	0.049239	0.033213	0.045217	0.033213	0.049022

Table 8.5: Classification performance in terms of classification error for training and testing datasets by combining the outputs of all classifiers for a particular feature vector and similarity model for the COREL Database. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

		Classification error in probability space					
		MVG		FIT		GMIX	
Feature	Classifier	Training	Testing	Training	Testing	Training	Testing
LAR+COOC	Product rule	0.276852	0.321225	0.287901	0.290671	0.313889	0.363065
	Sum rule	0.276852	0.321481	0.287840	0.290671	0.313920	0.363650
	Max rule	$\boxed{0.266914}$	$\boxed{0.314103}$	0.289012	0.291347	0.313333	$\boxed{0.362225}$
	Min rule	$\boxed{0.266914}$	$\boxed{0.314103}$	0.289012	0.291347	0.313333	$\boxed{0.362225}$
	Median rule	0.279784	0.325444	$\boxed{0.286667}$	$\boxed{0.290032}$	$\boxed{0.312469}$	0.364745
	Maj. vote rule	0.279784	0.325444	$\boxed{0.286667}$	$\boxed{0.290032}$	$\boxed{0.312469}$	0.364745
GABOR	Product rule	0.213117	0.258711	0.326636	$\boxed{0.331836}$	0.212160	0.259369
	Sum rule	0.213333	0.258657	0.326728	0.331927	0.212160	0.259551
	Max rule	$\boxed{0.200494}$	$\boxed{0.255022}$	0.333056	0.335525	$\boxed{0.199444}$	$\boxed{0.254876}$
	Min rule	$\boxed{0.200494}$	$\boxed{0.255022}$	0.333056	0.335525	$\boxed{0.199444}$	$\boxed{0.254876}$
	Median rule	0.220494	0.262127	$\boxed{0.322623}$	0.335653	0.219012	0.262747
	Maj. vote rule	0.220494	0.262127	$\boxed{0.322623}$	0.335653	0.219012	0.262747
MOMENTS	Product rule	0.286790	0.315198	0.343179	0.340237	0.287284	0.328001
	Sum rule	0.286728	0.315070	0.343272	0.340164	0.287562	$\boxed{0.314797}$
	Max rule	$\boxed{0.276420}$	$\boxed{0.310194}$	0.348704	0.343670	$\boxed{0.278735}$	0.338903
	Min rule	$\boxed{0.276420}$	$\boxed{0.310194}$	0.348704	0.343670	$\boxed{0.278735}$	0.338903
	Median rule	0.291667	0.318559	$\boxed{0.340031}$	$\boxed{0.339944}$	0.292870	0.332968
	Maj. vote rule	0.291667	0.318559	$\boxed{0.340031}$	$\boxed{0.339944}$	0.292870	0.332968
TAMURA	Product rule	0.337191	0.353422	0.335463	$\boxed{0.349441}$	0.366451	0.370571
	Sum rule	0.337284	0.353642	$\boxed{0.335432}$	0.349496	0.366389	0.370590
	Max rule	$\boxed{0.333642}$	$\boxed{0.347304}$	0.339136	0.350482	$\boxed{0.358395}$	$\boxed{0.364654}$
	Min rule	$\boxed{0.333642}$	$\boxed{0.347304}$	0.339136	0.350482	$\boxed{0.358395}$	$\boxed{0.364654}$
	Median rule	0.338272	0.360381	0.335617	0.350720	0.369938	0.373420
	Maj. vote rule	0.338272	0.360381	0.335617	0.350720	0.369938	0.373420
COLHIST	Product rule	0.127253	$\boxed{0.177917}$	0.250556	0.263624	0.123642	0.173205
	Sum rule	0.127377	0.178008	0.249290	0.262894	0.123765	0.173826
	Max rule	$\boxed{0.123519}$	0.179305	0.255617	0.268701	$\boxed{0.119691}$	$\boxed{0.172785}$
	Min rule	$\boxed{0.123519}$	0.179305	0.255617	0.268701	$\boxed{0.119691}$	$\boxed{0.172785}$
	Median rule	0.136698	0.185514	$\boxed{0.245031}$	$\boxed{0.259935}$	0.134105	0.180966
	Maj. vote rule	0.136698	0.185514	$\boxed{0.245031}$	$\boxed{0.259935}$	0.134105	0.180966

Table 8.6: Classification performance in terms of classification error for training and testing datasets by combining the outputs of a particular classifier for all feature vectors and similarity models for the COREL Database. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

	Classification error in probability space					
	Product rule		Sum rule		Max rule	
Database	Training	Testing	Training	Testing	Training	Testing
Gaussian linear classifier	0.111883	$\boxed{0.155672}$	0.130093	0.167507	$\boxed{0.105556}$	0.165187
Gaussian quadratic classifier	$\boxed{0.094815}$	0.151052	0.103395	$\boxed{0.147052}$	0.111235	0.173205
Fisher's linear classifier	0.111883	$\boxed{0.155672}$	0.130093	0.167507	$\boxed{0.105556}$	0.165187
Logistic linear classifier	$\boxed{0.084568}$	$\boxed{0.144824}$	0.095988	0.146212	0.098302	0.173460
Scaled nearest mean classifier	$\boxed{0.235278}$	0.245051	0.235340	$\boxed{0.240102}$	0.251852	0.277906
	Min rule		Median rule		Maj. vote rule	
	Training	Testing	Training	Testing	Training	Testing
Gaussian linear classifier	$\boxed{0.105556}$	0.165187	0.185309	0.212744	0.160278	0.193294
Gaussian quadratic classifier	0.111235	0.173205	0.220586	0.239608	0.195864	0.217036
Fisher's linear classifier	$\boxed{0.105556}$	0.165187	0.185309	0.212744	0.160278	0.193294
Logistic linear classifier	0.098302	0.173460	0.140123	0.177533	0.132778	0.176693
Scaled nearest mean classifier	0.251852	0.277906	0.276759	0.268811	0.261667	0.262747

Table 8.7: Classification performance in terms of classification error for training and testing datasets by combining the outputs of all classifiers for all feature vectors and similarity models for all databases. The best performing combination rules (that gave the smallest classification errors) are marked by boxes.

	Classification error in probability space					
	Product rule		Sum rule		Max rule	
Database	Training	Testing	Training	Testing	Training	Testing
ISL Database	0.028810	$\boxed{0.081865}$	0.072778	0.115348	$\boxed{0.024286}$	0.099504
VisTex Database	$\boxed{0.008152}$	$\boxed{0.028804}$	0.044988	0.045326	0.017512	0.088696
COREL Database	$\boxed{0.096173}$	$\boxed{0.149134}$	0.116142	0.155435	0.098519	0.170703
	Min rule		Median rule		Maj. vote rule	
	Training	Testing	Training	Testing	Training	Testing
ISL Database	$\boxed{0.024286}$	0.099504	0.133651	0.163636	0.130794	0.161015
VisTex Database	0.017512	0.088696	0.070652	0.068261	0.067935	0.066522
COREL Database	0.098519	0.170703	0.193179	0.212470	0.185679	0.206900

are given in Figures 8.1, 8.2 and 8.3. We can see that classifiers trained on the same feature vector were highly positively correlated (hence, they violated the independence assumptions and gave poor results classification tests). On the other hand, classifiers that used training data from different models were closer to the ideal independence case. Gabor and color histogram feature vectors performed better than other features and the multivariate Gaussian model performed better than other models. Tamura feature vectors gave the worst performance.

8.3 Retrieval Performance

8.3.1 Retrieval Using Combined Classifiers

Classifier combination methods can also be used for retrieval based on the posterior ratios as described in Section 7.4.1. The following figures present precision vs. recall using a combination of n classifiers for different databases.

Figure 8.4: Outputs of a particular classifier for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA) and all similarity models (MVG, FIT) were combined for the ISL Database ($n = 8$).

Figure 8.5: Outputs of a particular classifier for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA, COLHIST) and all similarity models (MVG, FIT) were combined for the VisTex Database ($n = 10$).

Figure 8.6: Outputs of a particular classifier for feature vectors (LAR+COOC, GABOR, MOMENTS, COLHIST) and one similarity model (MVG) were combined for the COREL Database ($n = 4$). (Additional classifiers can be used with increased computational requirements.)

Figure 8.7: Outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean) for all feature vectors (LAR+COOC,

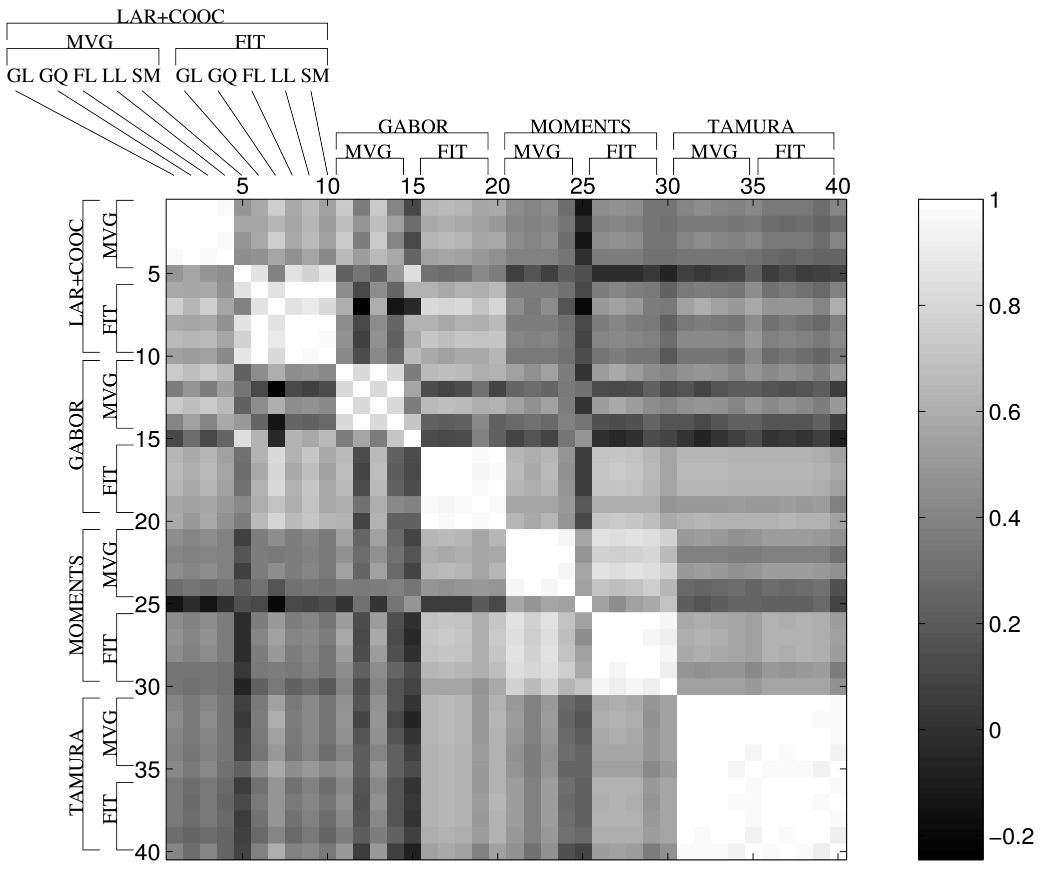


Figure 8.1: Q statistics values (mapped to gray levels according to the scale shown on the right side of the figure) for all feature vectors, similarity models and classifiers for the ISL Database. Each group of 10 classifiers correspond to one of the feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA). Within each group, groups of 5 classifiers (Gaussian linear, GL; Gaussian quadratic, GQ; Fisher's linear, FL; Logistic linear, LL; scaled nearest mean, SM) correspond to similarity models (multivariate Gaussian, MVG; independently fitted distributions, FIT). Q is positive, zero and negative for classifier pairs that are positively correlated, statistically independent, and negatively correlated, respectively.

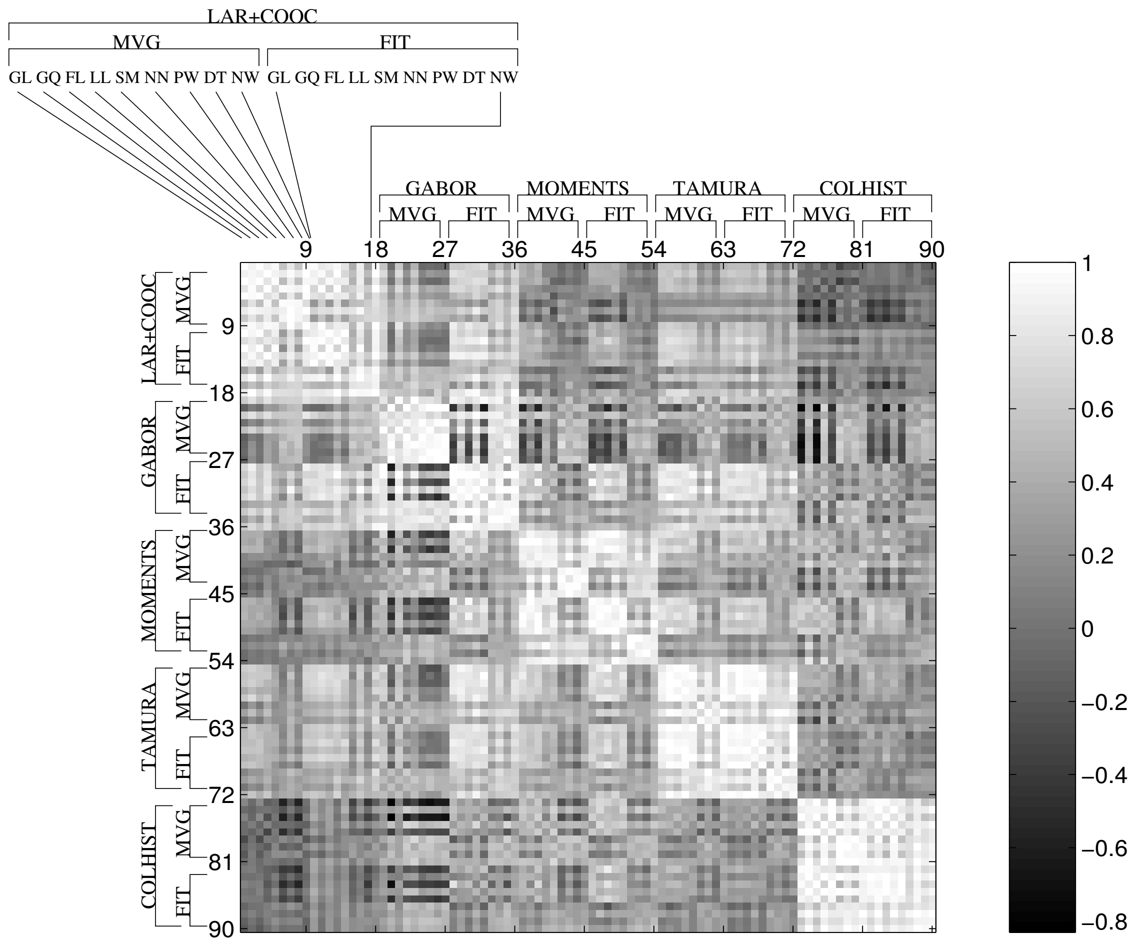


Figure 8.2: Q statistics values (mapped to gray levels according to the scale shown on the right side of the figure) for all feature vectors, similarity models and classifiers for the VisTex Database. Each group of 18 classifiers correspond to one of the feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA, COLHIST). Within each group, groups of 9 classifiers (Gaussian linear, GL; Gaussian quadratic, GQ; Fisher's linear, FL; Logistic linear, LL; scaled nearest mean, SM; nearest neighbor, NN; Parzen, PW; binary decision tree, DT; neural network, NW) correspond to similarity models (multivariate Gaussian, MVG; independently fitted distributions, FIT). Q is positive, zero and negative for classifier pairs that are positively correlated, statistically independent, and negatively correlated, respectively.

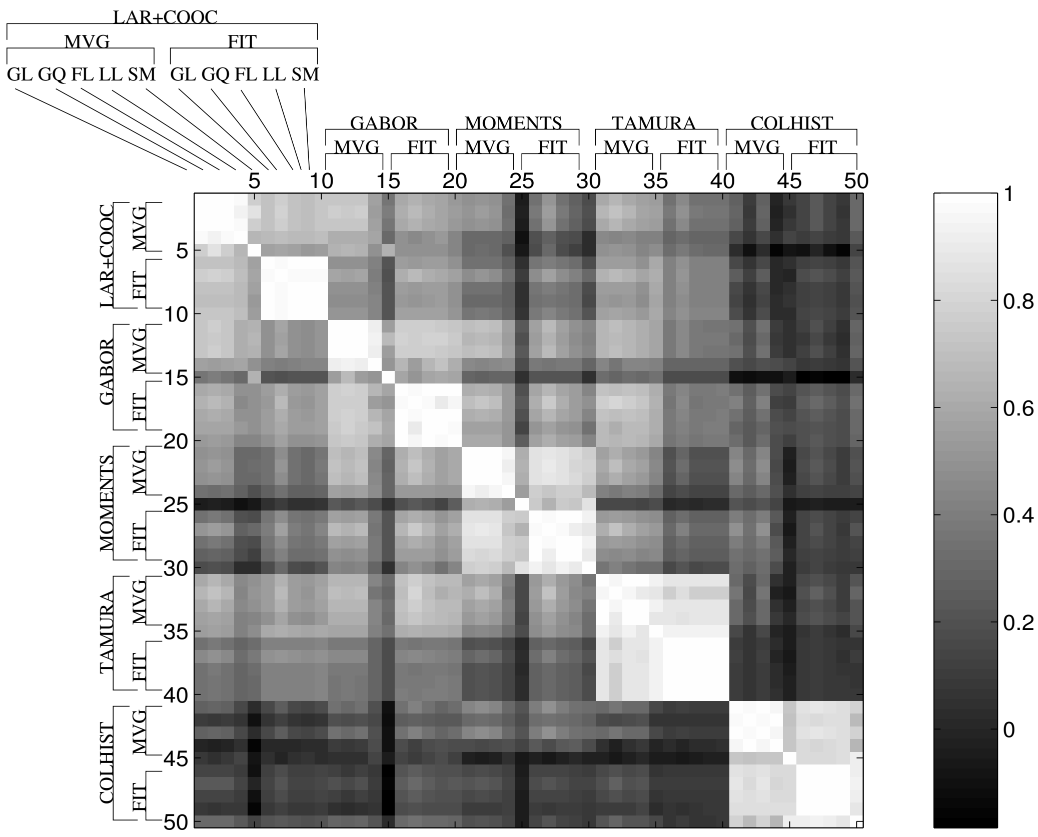


Figure 8.3: Q statistics values (mapped to gray levels accordingh to the scale shown on the right side of te figure) for all feature vectors, similarity models and classifiers for the COREL Database. Each group of 10 classifiers correspond to one of the feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA, COLHIST). Within each group, groups of 5 classifiers (Gaussian linear, GL; Gaussian quadratic, GQ; Fisher's linear, FL; Logistic linear, LL; scaled nearest mean, SM) correspond to similarity models (multivariate Gaussian, MVG; independently fitted distributions, FIT). Q is positive, zero and negative for classifier pairs that are positively correlated, statistically independent, and negatively correlated, respectively.

GABOR, MOMENTS, TAMURA) and all similarity models (MVG, FIT) were combined for the ISL Database ($n = 40$); outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean, neural network) for all feature vectors (LAR+COOC, GABOR, MOMENTS, TAMURA, COLHIST) and all similarity models (MVG, FIT) were combined for the VisTex Database ($n = 60$); and outputs of all classifiers (Gaussian linear, Gaussian quadratic, Fisher's linear, Logistic linear, scaled nearest mean) for feature vectors (LAR+COOC, GABOR, MOMENTS, COLHIST) and one similarity model (MVG) were combined for the COREL Database ($n = 20$).

The results were similar to those of the classification experiments. Using combined classifiers did not always improve the retrieval performance. However, some classifiers (e.g. Logistic linear classifier) and some combination methods (e.g. product rule), which also performed the best in classification experiments, consistently gave better results than the individual models (Figures 8.4, 8.5, 8.6).

The reasons for low precision in the low recall parts of some of the precision-recall curves were the small number of classifiers used during combination and the relatively small training data set used for training both the individual classifiers and the combination rules. Since the testing sets and training sets were different, the query images could not always be retrieved as the very top images in the retrieval set and we could not have a perfect retrieval when only a few images were retrieved. However, the precision-recall curves stayed flat for a large range of recall because the classifiers consistently retrieved more relevant images compared to the individual models. Using two-thirds of the whole data for training and one-third for testing slightly improved the results. Adding more classifiers to the combination set had a larger positive impact on the results but also increased computational requirements. Q statistics values can be useful in deciding which classifiers will be included in the combination.

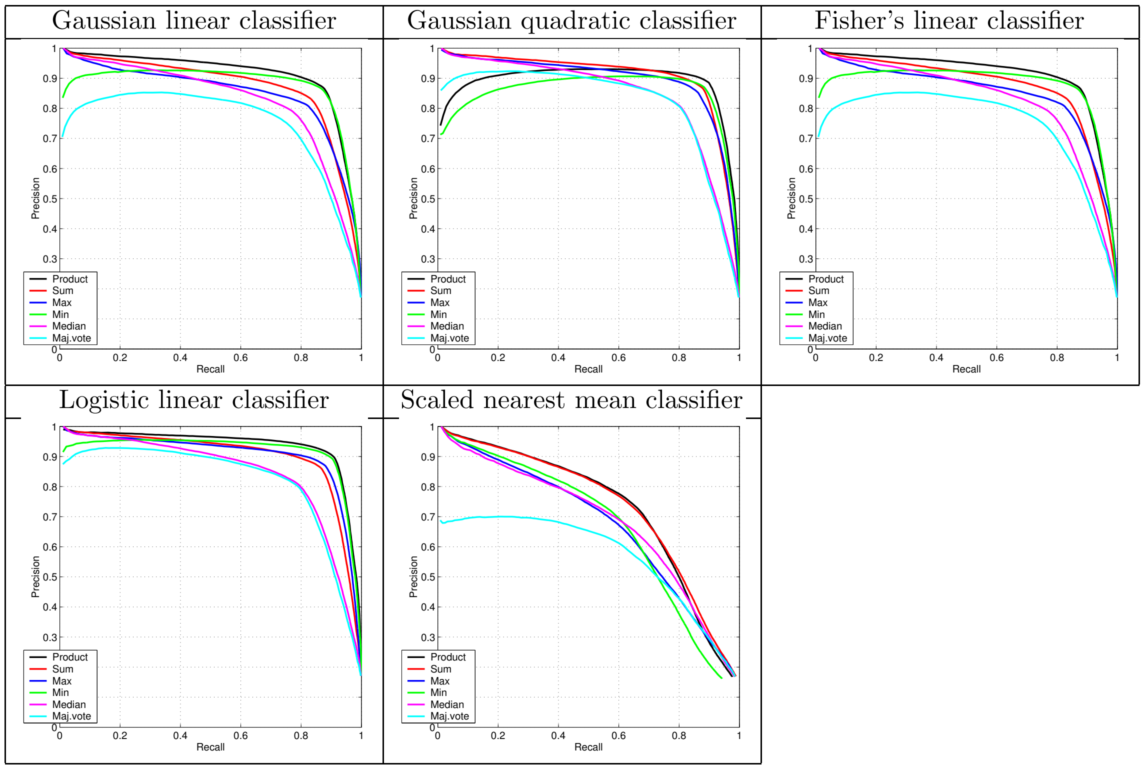


Figure 8.4: Retrieval performance in terms of precision (y-axis) and recall (x-axis) by combining the outputs of a particular classifier for all feature vectors and similarity models for the ISL Database. Different curves within the same plot represent the classifier combination methods product rule (black), sum rule (red), max rule (blue), min rule (green), median rule (magenta), and majority vote rule (cyan).

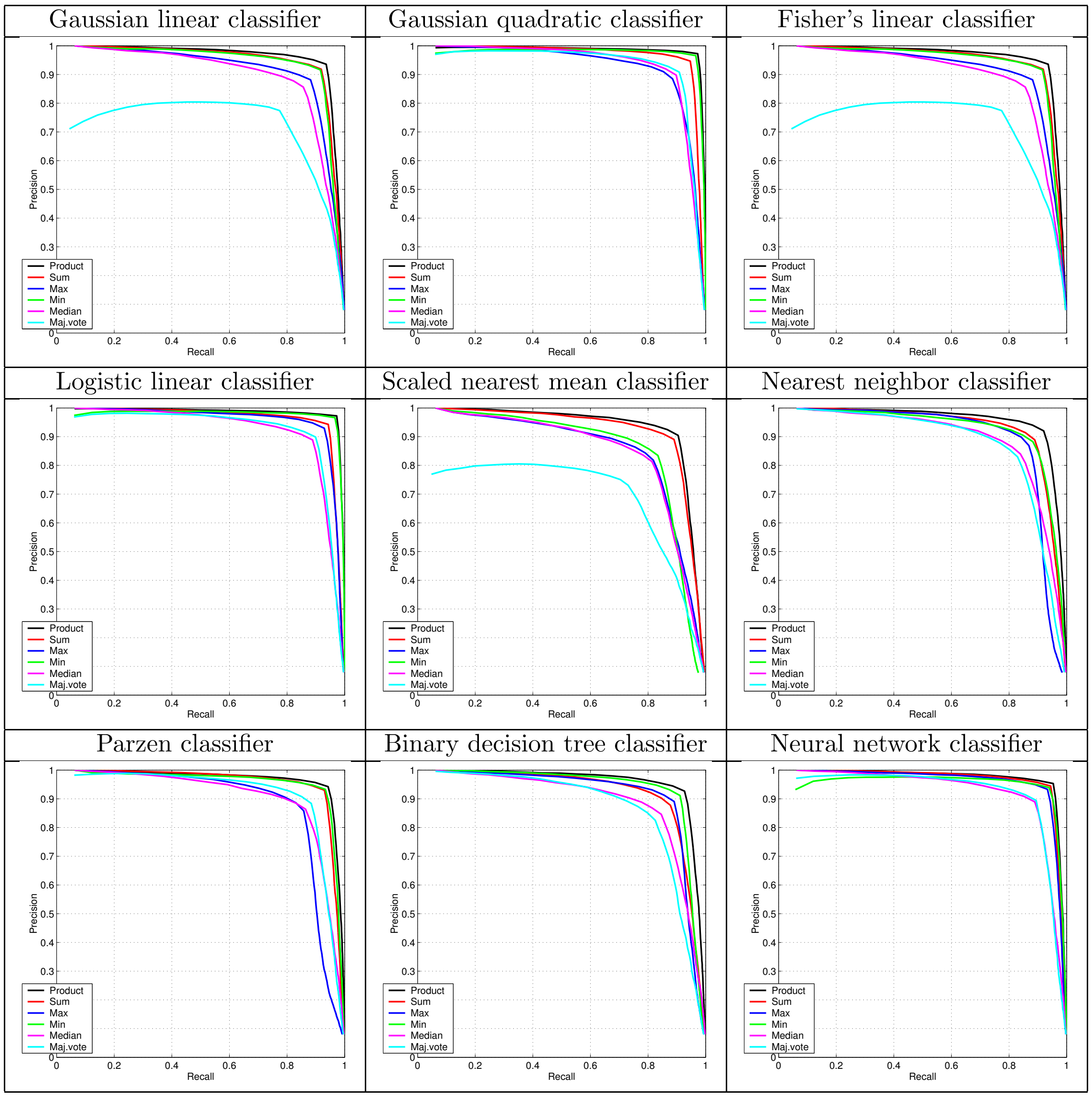


Figure 8.5: Retrieval performance in terms of precision (y-axis) and recall (x-axis) by combining the outputs of a particular classifier for all feature vectors and similarity models for the VisTex Database. Different curves within the same plot represent the classifier combination methods product rule (black), sum rule (red), max rule (blue), min rule (green), median rule (magenta), and majority vote rule (cyan).

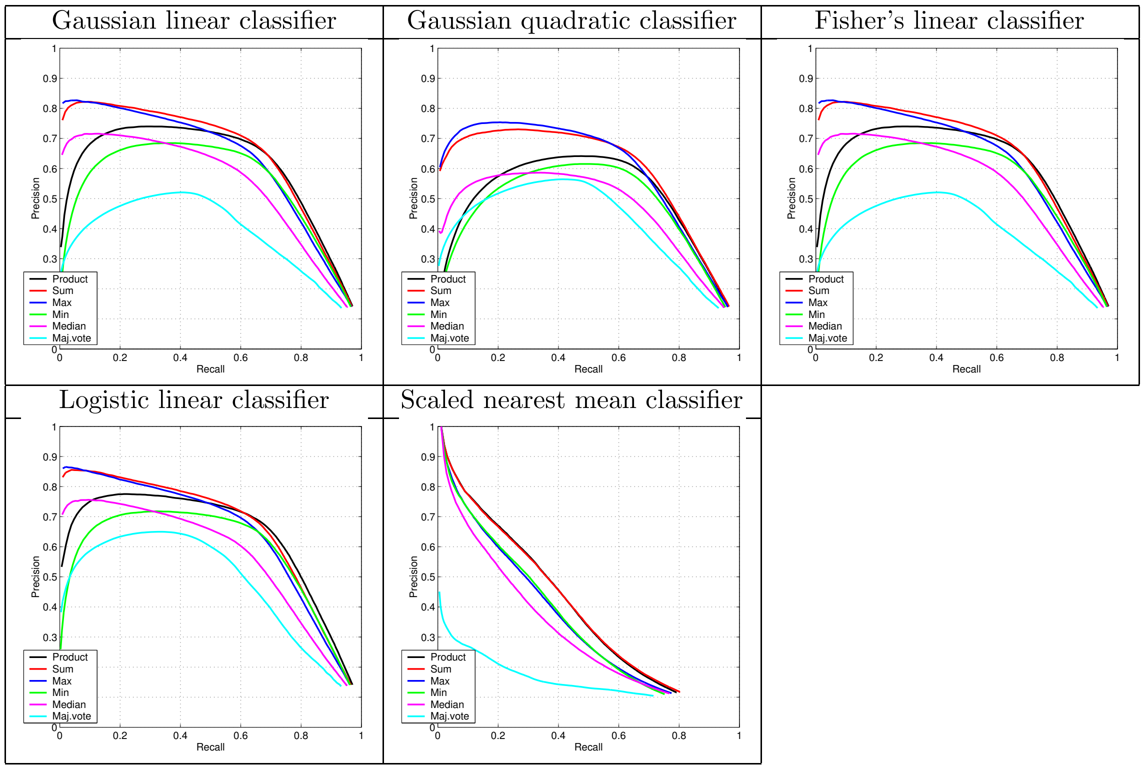


Figure 8.6: Retrieval performance in terms of precision (y-axis) and recall (x-axis) by combining the outputs of a particular classifier for all feature vectors and similarity models for the COREL Database. Different curves within the same plot represent the classifier combination methods product rule (black), sum rule (red), max rule (blue), min rule (green), median rule (magenta), and majority vote rule (cyan).

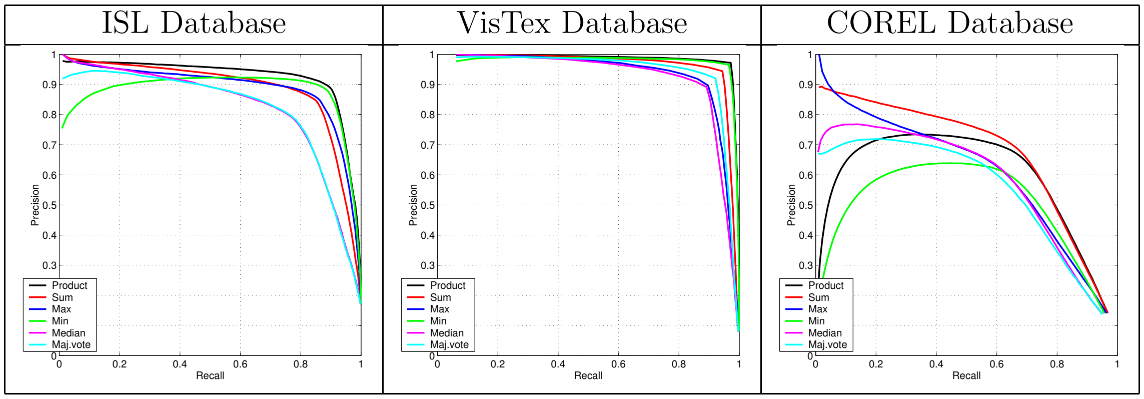


Figure 8.7: Retrieval performance in terms of precision (y-axis) and recall (x-axis) by combining the outputs of a particular classifier for all feature vectors and similarity models for all databases. Different curves within the same plot represent the classifier combination methods product rule (black), sum rule (red), max rule (blue), min rule (green), median rule (magenta), and majority vote rule (cyan).

The best performing classifier combination contained the Logistic linear classifier with the product rule for ISL and VisTex Databases and the max rule for the COREL Database. This combination especially achieved an almost perfect retrieval for the VisTex Database, which has been used as the test dataset in most of the content-based retrieval papers. Combining the outputs of all classifiers for all feature vectors and all similarity models did not give much improvement and is not worth the heavy computation (Figure 8.7).

This significant performance of the simple linear classifiers in improving the retrieval results shows the power of the probabilistic framework which simplifies the problem and allows the estimation of less complex models in multiple levels while still being very effective.

8.3.2 Retrieval Using Bayesian Network

The second part of the retrieval experiments presents results using the naive Bayesian network described in Section 7.4.2. As described in Section 7.3.1, we estimated the parameters for Gaussian and Gamma distributions from training data for the likelihood ratio and distance values for relevance and irrelevance classes. The final choice for a Gaussian or Gamma was made according to the Kolmogorov-Smirnov statistic. Example histograms and fitted distributions are given in Figure 8.8.

The results of retrieval experiments in terms of precision vs. recall are given in Figure 8.9 for all databases. We used only a small number of models because models trained on the same feature vectors may violate the conditional independence assumptions as discussed in Section 8.2. Small number of models also requires less amount of computation while giving significant performance improvements. The models used were chosen according to the classification results in Chapter 4. The leaf nodes in the networks used in these particular experiments are the measurements by the following models:

x_1 **:** Line-angle-ratio and co-occurrence feature vectors with the multivariate Gaussian model (LAR+ COOC + MVG),

x_2 **:** Gabor feature vectors with the multivariate Gaussian model (GABOR + MVG),

x_3 **:** Color histogram feature vectors with the multivariate Gaussian model (COLHIST + MVG) (for VisTex and COREL only).

Precision and recall for individual models are also given for comparison.

Combining feature vectors and similarity models using the proposed naive Bayesian network gave an improvement over the individual models in all cases. A relative improvement[1] of 1.31% for precision and recall (computed at the knee of the precision vs. recall curve) over the individually best performing model for the ISL Database, 7.14% precision and recall for the VisTex Database and 16.03% precision and 14.18% recall for the COREL Database were obtained. The improvements were not too significant for the ISL and VisTex Databases which are relatively simple enough that

[1]Relative improvement is computed as $\frac{\textit{new value - old value}}{\textit{old value}} \times 100$.

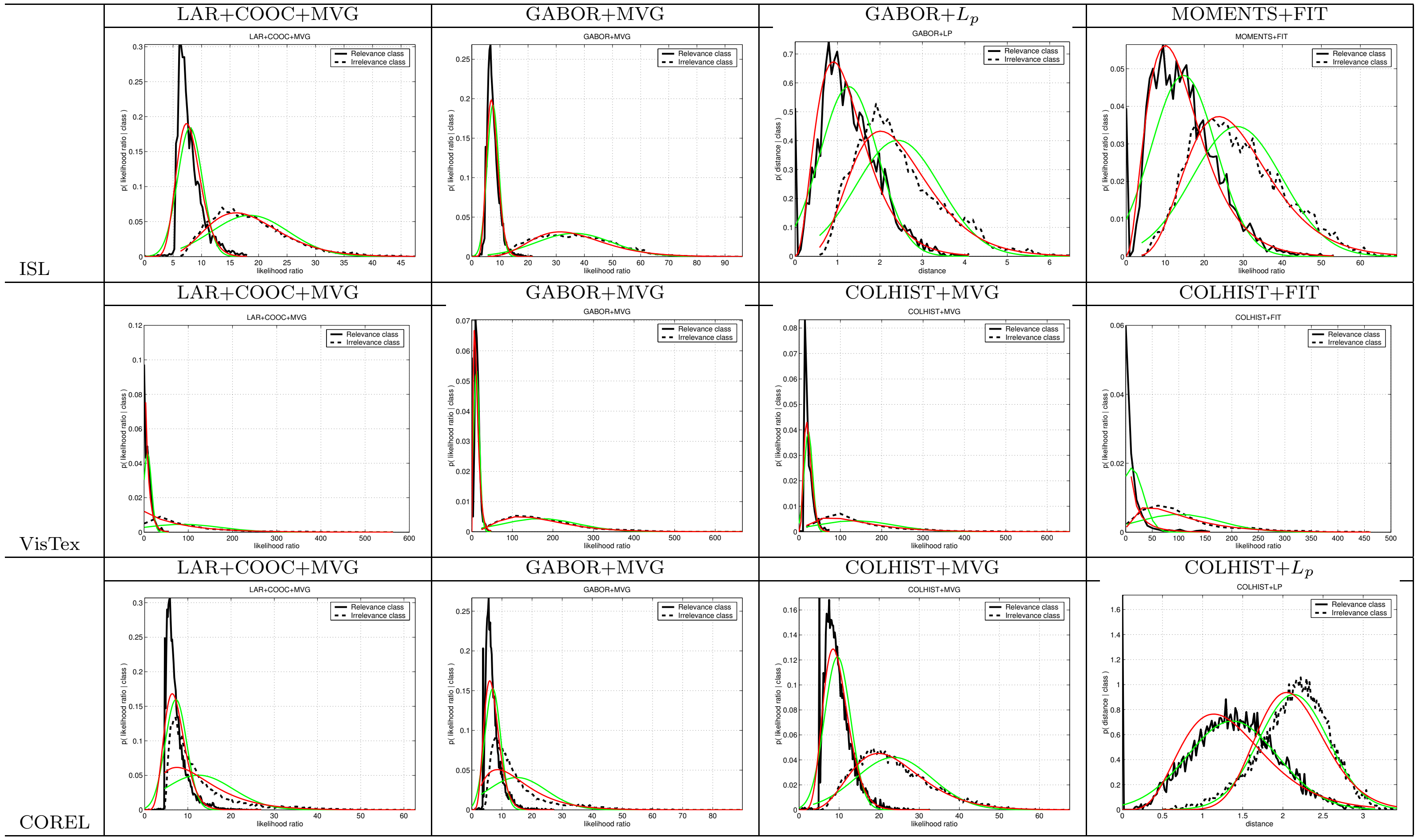


Figure 8.8: Example histograms and fitted Gaussian (green) and Gamma (red) distributions for training likelihood ratio and distance values for relevance (solid black) and irrelevance (dashed black) classes for different databases. Kolmogorov-Smirnov statistic was used to choose the best fit. Gamma model was usually a better fit than the Gaussian. (Note that likelihood ratio values are reversed to have smaller values for similar images.)

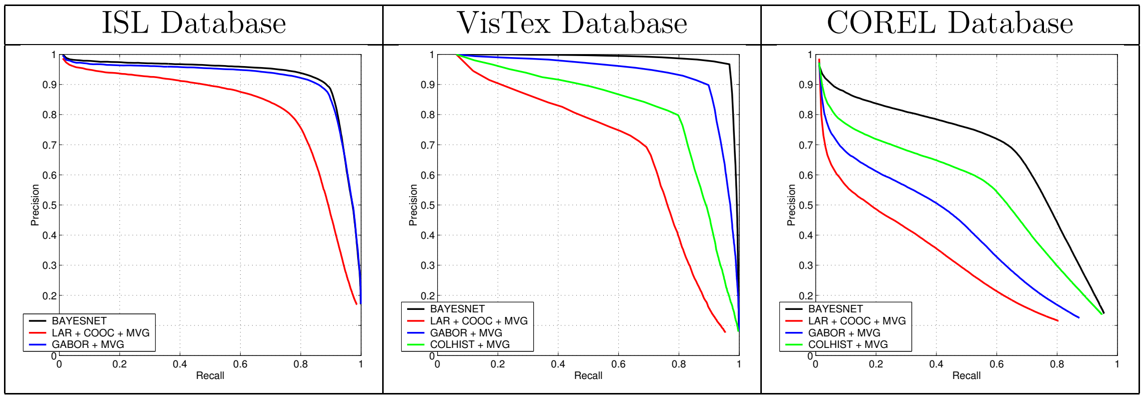
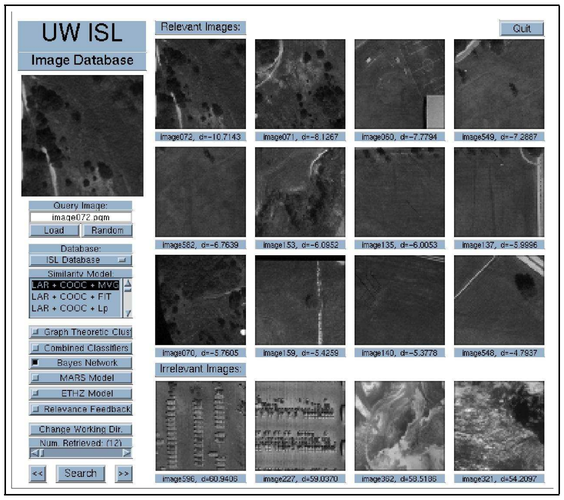


Figure 8.9: Retrieval performance in terms of precision (y-axis) and recall (x-axis) by the naive Bayesian networks for all databases. Different curves within the same plot show the best retrival performances obtained using line-angle-ratio and co-occurrence feature vector (red), Gabor feature vector (blue) and color histogram feature vector (green) individually, and retrieval using their combination with the Bayesian network model (black).

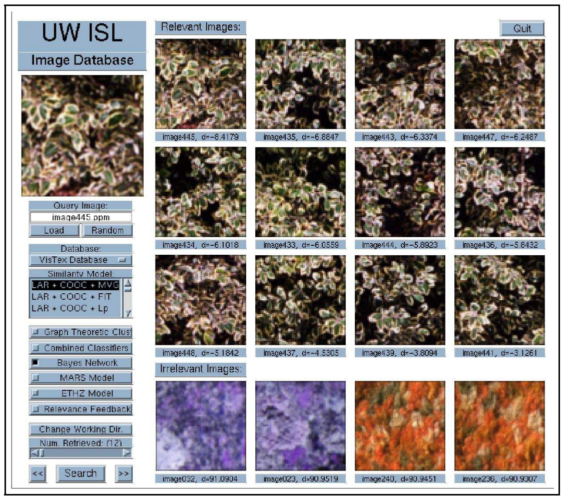
the Gabor feature vectors and the multivariate Gaussian model already achieved an acceptable performance when used alone. However, the improvement for the COREL Database was very high even when only three models were combined.

The performance of the naive Bayesian network model was almost the same as the performance of combined classifiers for the ISL and VisTex Databases. However, the improvement for the COREL Database was very significant. We did not have the training sample size and overfitting problems here because the Bayesian network model is simpler (and also more efficient) than the combined classifier models.

Example queries are given in Figure 8.10. These examples use the same query images as in the examples given at the end of Chapter 4 where only one feature vector was used at a time. We could get perfect retrieval except for the cheetah example in Figure 8.3.2 where false positives still remained after combination. However, one feedback iteration successfully removed the false positives from the retrieval list for that query. Both visual examples and precision-recall curves show that the combina-



(a) An example query for landscape using Bayesian network (12/12)



(b) An example query for leaves using Bayesian network (12/12)

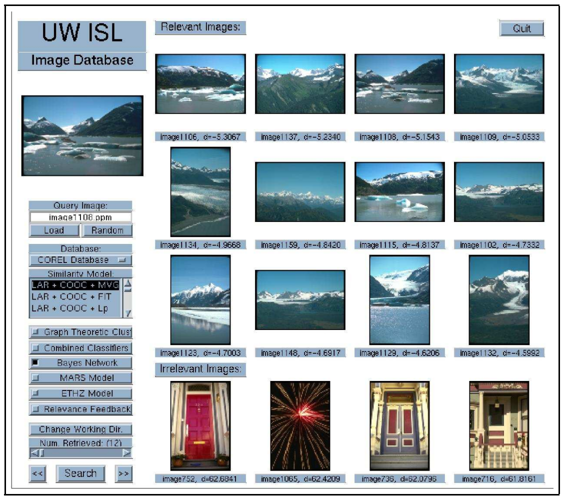
Figure 8.10: Example queries using combined feature vectors and similarity models. The query images are same as the ones used in the examples in Chapter 4. The numbers in parentheses in sub-captions show the number of correct matches for each case.

tion methods we proposed can get rid of most of the false alarms and provide effective retrieval.

8.4 Competing Algorithms

The main goals of the framework we proposed is to combine multiple feature vectors and also incorporate relevance feedback for interactive searches. Therefore, competing algorithms should also support feature combination and relevance feedback. Among the methods discussed in Chapter 2, the ones that were proposed by Rui *et al.* [172] and Schroder *et al.* [177] support feature combination, online learning and iterative retrieval.

Rui *et al.*'s MARS system [172] at the University of Illinois supports a multimedia

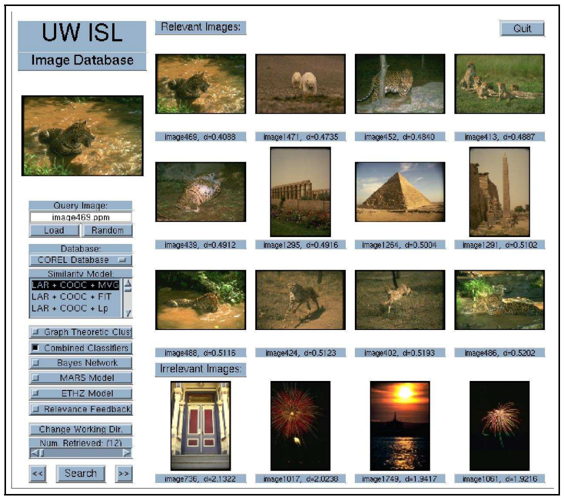


UW ISL
Image Database
Relevant Images:
Quit
Query Image: image1108.ppm
Load | Random
Database: COREL Database
Similarity Model: [✓ LAR + COOC + MVG] LAR + COOC + FIT LAR + COOC + Lp
Graph Theoretic Clust
Combined Classifiers
[✓ Bayes Network]
MARS Model
ETHZ Model
Relevance Feedback
Change Working Dir.
Num. Retrieved: (12)
<< | Search | >>

image1106, d=-5.3067 | image1137, d=-5.2340 | image1108, d=-5.1543 | image1109, d=-5.0533
image1134, d=-4.9668 | image1159, d=-4.8420 | image1115, d=-4.8137 | image1102, d=-4.7332
image1123, d=-4.7003 | image1148, d=-4.6917 | image1129, d=-4.6206 | image1132, d=-4.5992

Irrelevant Images:
image752, d=62.6841 | image1065, d=62.4209 | image736, d=62.0796 | image716, d=61.8161

(a) An example query for glaciers and mountains using Bayesian network (12/12)

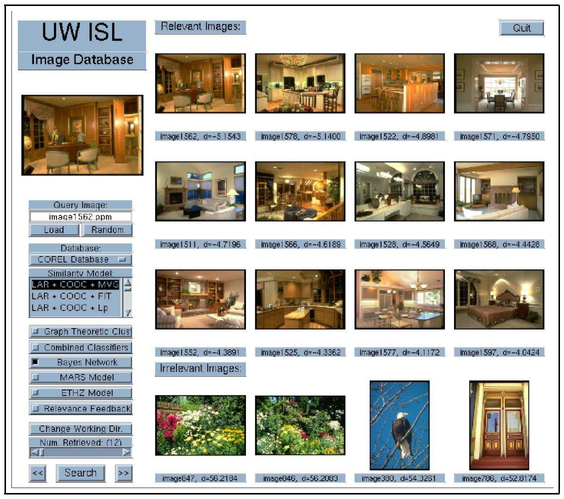


UW ISL
Image Database
Relevant Images:
Quit
Query Image: image469.ppm
Load | Random
Database: COREL Database
Similarity Model: [✓ LAR + COOC + MVG] LAR + COOC + FIT LAR + COOC + Lp
Graph Theoretic Clust
[✓ Combined Classifiers]
Bayes Network
MARS Model
ETHZ Model
Relevance Feedback
Change Working Dir.
Num. Retrieved: (12)
<< | Search | >>

image469, d=0.4088 | image1471, d=0.4735 | image452, d=0.4840 | image413, d=0.4887
image439, d=0.4912 | image1295, d=0.4916 | image1264, d=0.5004 | image1291, d=0.5102
image488, d=0.5116 | image424, d=0.5123 | image402, d=0.5193 | image486, d=0.5202

Irrelevant Images:
image736, d=2.1322 | image1017, d=2.0238 | image1749, d=1.9417 | image1061, d=1.9216

(b) An example query for cheetahs using combined classifiers (8/12)

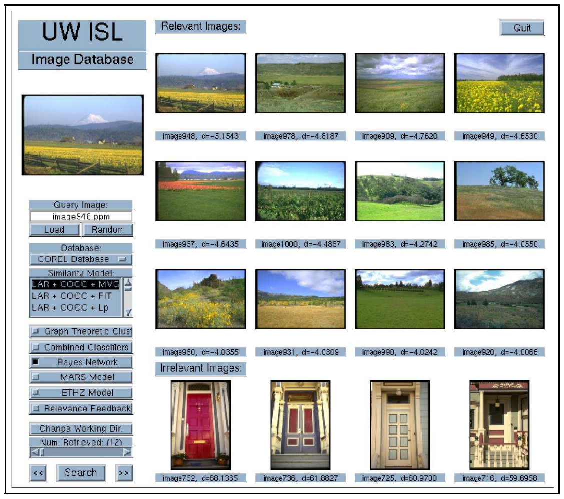


UW ISL
Image Database
Relevant Images:
Quit
Query Image: image1562.ppm
Load | Random
Database: COREL Database
Similarity Model: [✓ LAR + COOC + MVG] LAR + COOC + FIT LAR + COOC + Lp
Graph Theoretic Clust
Combined Classifiers
[✓ Bayes Network]
MARS Model
ETHZ Model
Relevance Feedback
Change Working Dir.
Num. Retrieved: (12)
<< | Search | >>

image1562, d=-5.1543 | image1578, d=-5.1400 | image1522, d=-4.8981 | image1571, d=-4.7950
image1511, d=-4.7196 | image1566, d=-4.6189 | image1528, d=-4.5649 | image1568, d=-4.4428
image1552, d=-4.3891 | image1525, d=-4.3362 | image1577, d=-4.1172 | image1597, d=-4.0424

Irrelevant Images:
image847, d=56.2184 | image846, d=56.2083 | image380, d=54.3261 | image786, d=52.8174

(c) An example query for residential interiors using Bayesian network (12/12)



UW ISL
Image Database
Relevant Images:
Quit
Query Image: image948.ppm
Load | Random
Database: COREL Database
Similarity Model: [✓ LAR + COOC + MVG] LAR + COOC + FIT LAR + COOC + Lp
Graph Theoretic Clust
Combined Classifiers
[✓ Bayes Network]
MARS Model
ETHZ Model
Relevance Feedback
Change Working Dir.
Num. Retrieved: (12)
<< | Search | >>

image948, d=-5.1543 | image978, d=-4.8187 | image909, d=-4.7620 | image949, d=-4.6530
image957, d=-4.6435 | image1000, d=-4.4857 | image983, d=-4.2742 | image985, d=-4.0550
image950, d=-4.0355 | image931, d=-4.0309 | image990, d=-4.0242 | image920, d=-4.0066

Irrelevant Images:
image752, d=68.1365 | image736, d=61.8827 | image725, d=60.9700 | image716, d=59.6958

(d) An example query for fields using Bayesian network (12/12)

Figure 8.10: (continued)

object model where multiple image representations with dynamically updated weights are used. An object is represented in terms of visual features such as color, texture, and shape in one level, and specific implementations of these feature types such as color histograms, co-occurrence matrices, Fourier descriptors in the next level. The system computes the overall similarity between images using weighted linear combinations of weighted Euclidean distances for different representations at each level. Weights at each level are independently updated by incrementing or decrementing them according to the user feedback in terms of positive and negative scores for each image. Gaussian normalization was used (like in Section 3.3.2) for the features before computing the distances and for the distances before taking linear combinations.

The MARS system is described in detail in [172] and it neither requires data-specific tuning nor has ambiguity in the implementation. In our implementation we used all 6 feature vectors listed in Section 8.1. Note that only a subset of these feature vectors were used for the proposed combined classifier and naive Bayesian network models. Both the MARS model and our models used the same training image sets. Precision vs. recall curves are given in Figure 8.11 for all databases. Although, feature combination in the MARS model improved the results over individual cases, the results of combination in our Bayesian network model were significantly better.

To perform content-based retrieval from a large remote sensing image archive at the Swiss Federal Institute of Technology at Zurich (ETHZ), Schroder *et al.* [177] use a Bayesian network with discrete variables to learn the relationships between a user-specific land cover type and low-level texture features. After feature vectors are extracted for each image, they are clustered using the k-means algorithm to obtain class labels for each feature type. These labels and a binary label for the specific land cover type are used in a naive Bayes structure by using discrete probability tables to encode the relationships. The attribute values for each pixel (or image) are the class labels for its corresponding feature vectors. Searching starts with uniform priors and relevance feedback is used to update the conditional probabilities using relative frequencies. Then, this trained classifier is used to perform a search in the database with the criterion being the posterior probabilities of a pixel (or image) belonging to the specific cover type given its feature classes. If two images have similar feature vectors (i.e. they are close to each other in the feature space), there is a high chance that they will have the same attribute value. If a particular attribute value (i.e. a cluster in the feature space) is found relevant by the user in the training examples for a particular land cover type, the pixels (or images) that have the same attribute value will have a high probability of belonging to that cover type.

Schroder *et al.* used feature vectors computed for each pixel and labeled them using the posterior probabilities, i.e. searching is done in the pixel level. In our implementation we used feature vectors computed for the whole image and ranked images according to their posterior probabilities, i.e. searching is done in the image level. The only data-specific design choice in our implementation was the choice of k in k-means clustering. Since the original paper [177] does not talk about how to choose the number of clusters in k-means, we set it to 20 by trial and error. We used all 6 feature vectors listed in Section 8.1. Training for the feature class labels was done using all images in the database; therefore, training and testing images sets were the same. Precision vs. recall curves are given in Figure 8.11 for all databases. Feature combination in the ETHZ model also improved the results over individual cases but combination using our Bayesian network model performed significantly better than both the ETHZ model and the MARS model.

Note that the numbers given in the comparisons correspond to the best performing models for the databases used in this dissertation. On the other hand, the competing algorithms were implemented using the descriptions in the corresponding journal papers (implementation details given above). The algorithms from this dissertation had the advantage of being chosen after extensive experiments in multiple levels while the competing algorithms were favored by training on the whole data (equal training and testing sets).

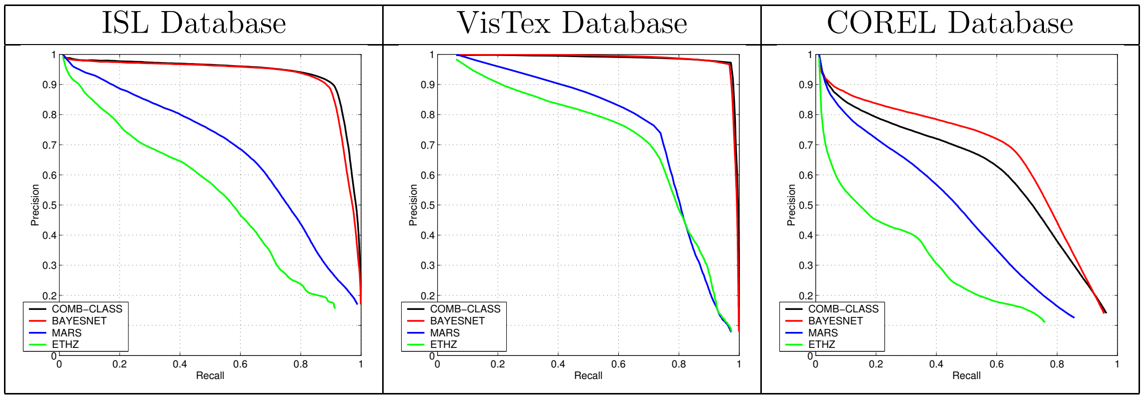


Figure 8.11: Retrieval performance in terms of precision (y-axis) and recall (x-axis) by the combined classifiers model (black), the Bayesian network model (red), the MARS model (blue) and the ETHZ model (green) for all databases. The Bayesian framework proposed in this dissertation performed significantly better than the competing algorithms.

8.5 Relevance Feedback

Retrieval experiments in the previous sections were done using only the original query image. This section presents the results of the Bayesian relevance feedback algorithm proposed in Section 7.4 and makes comparisons to the performances with relevance feedback in the MARS and ETHZ models. Since the Graphical User Interface (GUI) of our system shows the first 12 matches in the first screen, we used the feedback available from only the first 12 images in the retrieval experiments. In addition, user is allowed to select one or more of the 4 most irrelevant images for feedback in case those images are mistakenly labeled as irrelevant by the system. Automatic scripts were used to do the iterations for all test images. Each test image was used as the query and the retrieved images that belonged to the same groundtruth group as the query image were fed back as positive matches and the rest of the 12 were fed back as negative matches. The 4 irrelevant images in the last row of the GUI were not used in the automatic scripts.

Figures 8.12 - 8.14 show precision and recall results for the Bayesian relevance feedback algorithm that was run up to 5 iterations. The classifier combination rule that gave the best results in the classification experiments was used for each database (i.e. product rule for ISL and VisTex and max rule for COREL) for the combined classifiers model. Table 8.8 summarizes average precisions when 12 images were retrieved. We can see that each iteration gave an improvement over the case without feedback while the first iteration had the largest improvement. This is a desired situation because many relevant images are already available to the user after only the first feedback. We could get almost perfect retrieval (precision above 99%) for ISL and VisTex Databases and obtained significant improvement for the COREL Database.

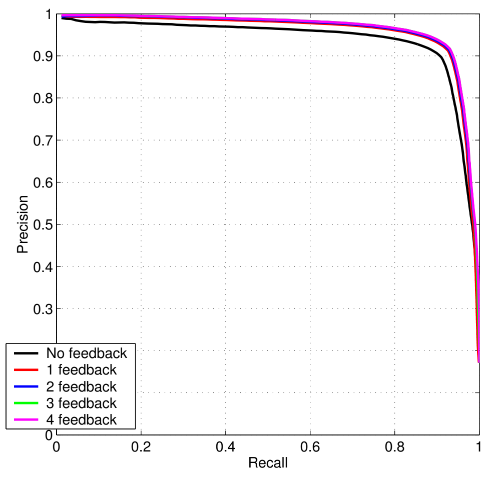
Feedback experiments using the same test images and same feedback images were also done for the MARS feedback model and the ETHZ feedback model. The results are also given in Figures 8.12 - 8.14. Both models showed significant improvements over the cases without feedback except that the MARS model gave worse results for the COREL Database. The ETHZ model gave large improvements in subsequent iterations but required more iterations than other models to achieve similar performance. However, it appeared to be more robust than the MARS model because it also used probabilities instead of heuristic weight assignments in the geometric similarity framework.

Figures 8.15 - 8.21 show example queries for the Bayesian relevance feedback algorithm. These examples show results of searches by first using a single feature vector, then using one of the combination methods and finally using relevance feedback. The queries were chosen among the images that performed poorly under both single feature vectors and combination models. We can see that Bayesian relevance feedback gave perfect retrieval after one or two iterations even though the initial results were quite bad.

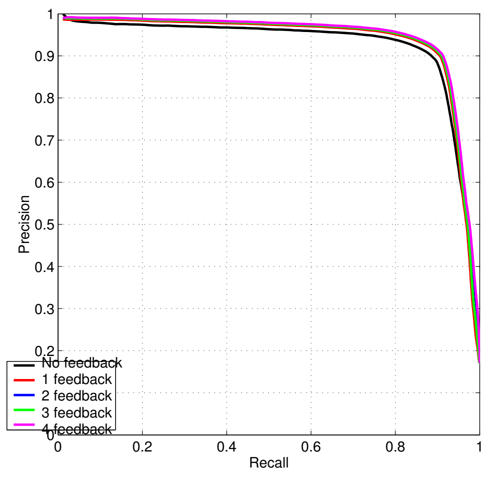
The bald eagle example in Figure 8.20 shows a case where presenting the worst matches as well as the best matches can help the user understand the results and

Table 8.8: Average precision when 12 images were retrieved using different feedback algorithms on all databases. "n r.f." represents the n'th feedback iteration. Improvements for each iteration over the case without feedback (0 r.f.) are given in parentheses. Bayesian relevance feedback achieved almost perfect retrieval.

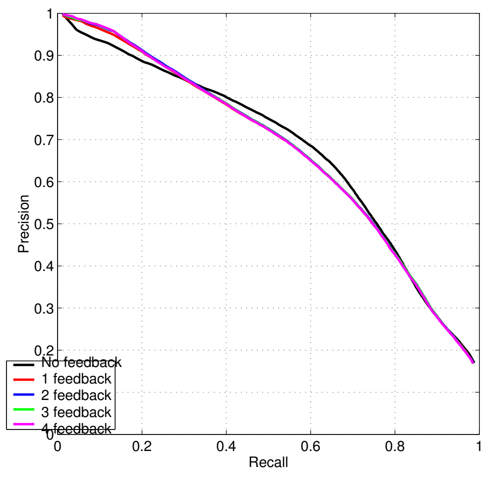
Database	Method	0 r.f.	1 r.f.	2 r.f.	3 r.f.	4 r.f.
ISL	Comb.class.	$\boxed{0.9790}$	$\boxed{0.9923}$ (1.36%)	$\boxed{0.9957}$ (1.70%)	$\boxed{0.9966}$ (1.80%)	$\boxed{0.9966}$ (1.80%)
	Bayes.net.	0.9752	0.9860 (1.10%)	0.9895 (1.46%)	0.9887 (1.38%)	0.9904 (1.55%)
	MARS	0.9066	0.9490 (4.68%)	0.9568 (5.54%)	0.9572 (5.59%)	0.9571 (5.57%)
	ETHZ	0.7459	0.8556 (14.71%)	0.8880 (19.06%)	0.9072 (21.63%)	0.9169 (22.93%)
VisTex	Comb.class.	0.9879	0.9945 (0.66%)	0.9946 (0.68%)	0.9946 (0.68%)	0.9946 (0.68%)
	Bayes.net.	$\boxed{0.9903}$	$\boxed{0.9989}$ (0.87%)	$\boxed{0.9986}$ (0.85%)	$\boxed{0.9986}$ (0.85%)	$\boxed{0.9986}$ (0.85%)
	MARS	0.8225	0.9078 (10.38%)	0.9220 (12.10%)	0.9206 (11.94%)	0.9230 (12.22%)
	ETHZ	0.7773	0.8946 (15.09%)	0.9134 (17.51%)	0.9293 (19.56%)	0.9348 (20.26%)
COREL	Comb.class.	0.8342	$\boxed{0.9113}$ (9.24%)	$\boxed{0.9363}$ (12.24%)	$\boxed{0.9407}$ (12.77%)	$\boxed{0.9421}$ (12.93%)
	Bayes.net.	$\boxed{0.8639}$	0.8857 (2.52%)	0.8904 (3.06%)	0.8924 (3.29%)	0.8931 (3.37%)
	MARS	0.7860	0.7441 (-5.34%)	0.7612 (-3.16%)	0.7716 (-1.83%)	0.7894 (0.42%)
	ETHZ	0.5757	0.7492 (30.12%)	0.7809 (35.63%)	0.8081 (40.36%)	0.8282 (43.85%)



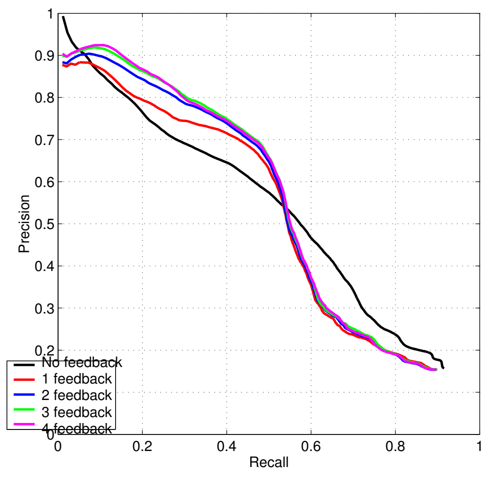
(a) Precision vs. recall for feedback iterations with combined classifiers



(b) Precision vs. recall for feedback iterations with Bayesian network

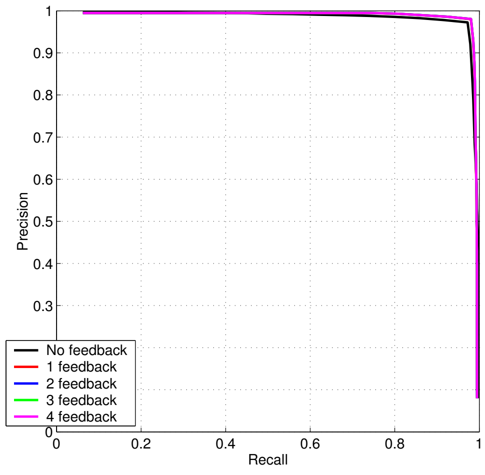


(c) Precision vs. recall for feedback iterations with MARS model

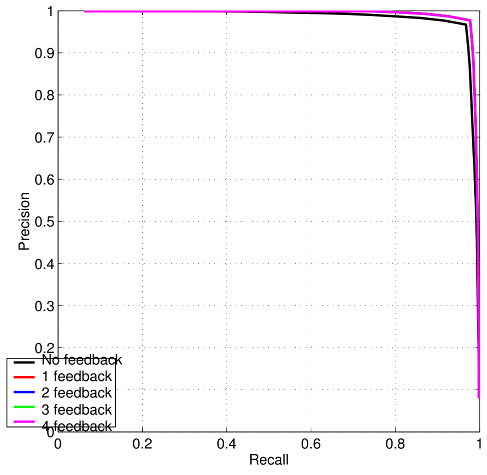


(d) Precision vs. recall for feedback iterations with ETHZ model

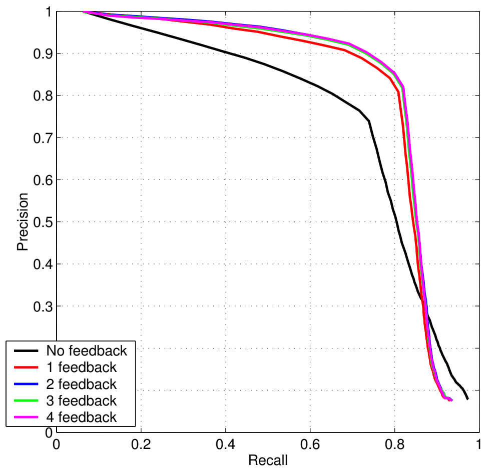
Figure 8.12: Precision and recall for multiple feedback iterations for the ISL Database. Feedback from only the first 12 images were used and precision was averaged for all images used in the tests. After obtaining the initial search results, groundtruth information was used to label images as relevant and irrelevant, and then this information was used as positive and negative feedback for iterative searches. Each curve labeled as "n feedback" within each plot shows precision and recall for the n'th feedback iteration for a particular feedback model.



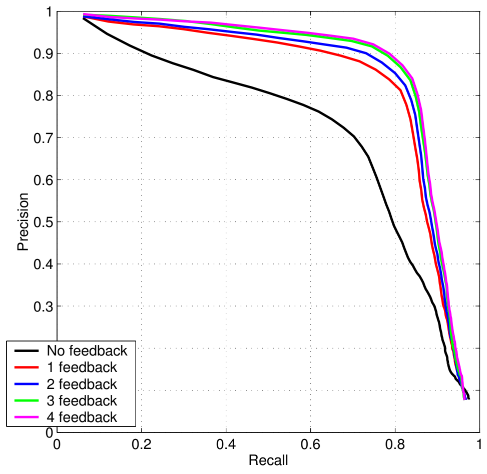
(a) Precision vs. recall for feedback iterations with combined classifiers



(b) Precision vs. recall for feedback iterations with Bayesian network

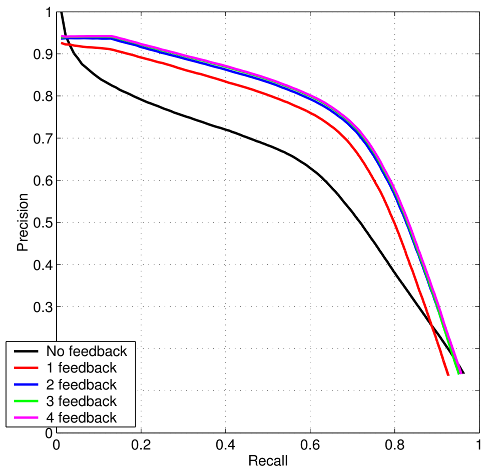


(c) Precision vs. recall for feedback iterations with MARS model

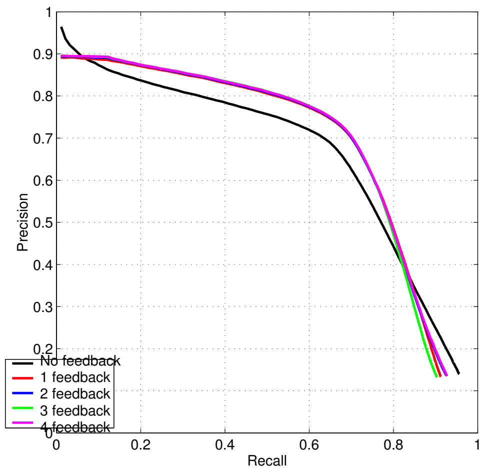


(d) Precision vs. recall for feedback iterations with ETHZ model

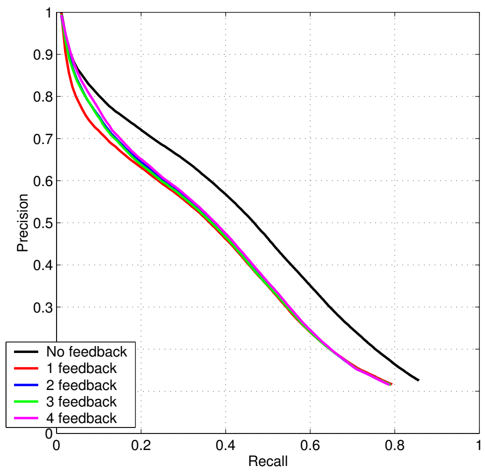
Figure 8.13: Precision and recall for multiple feedback iterations for the VisTex Database. Feedback from only the first 12 images were used and precision was averaged for all images used in the tests. After obtaining the initial search results, groundtruth information was used to label images as relevant and irrelevant, and then this information was used as positive and negative feedback for iterative searches. Each curve labeled as "n feedback" within each plot shows precision and recall for the n'th feedback iteration for a particular feedback model.



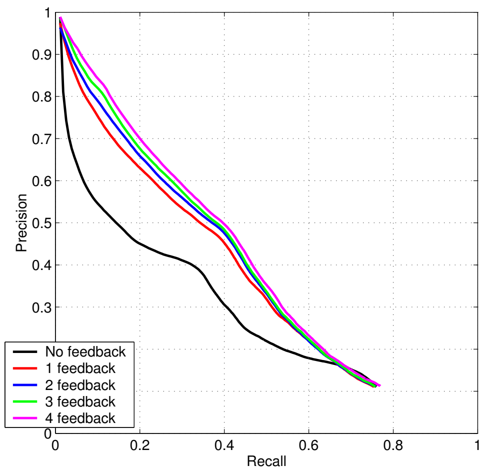
(a) Precision vs. recall for feedback iterations with combined classifiers



(b) Precision vs. recall for feedback iterations with Bayesian network

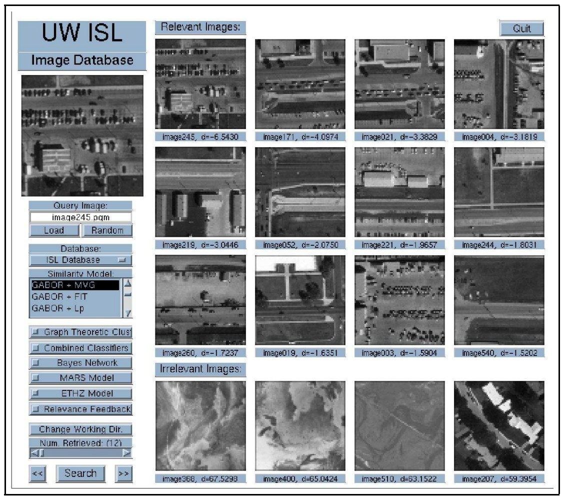


(c) Precision vs. recall for feedback iterations with MARS model



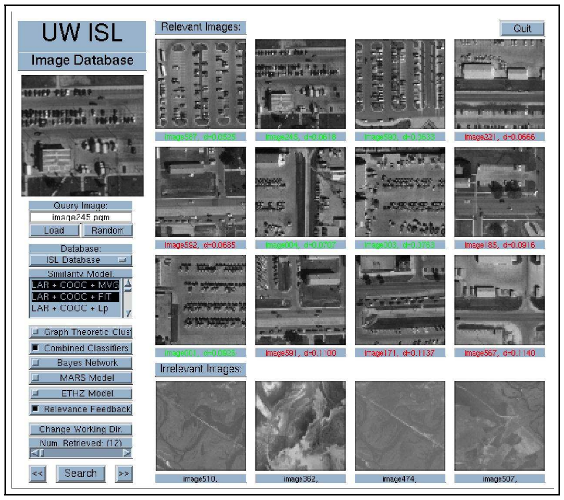
(d) Precision vs. recall for feedback iterations with ETHZ model

Figure 8.14: Precision and recall for multiple feedback iterations for the COREL Database. Feedback from only the first 12 images were used and precision was averaged for all images used in the tests. After obtaining the initial search results, groundtruth information was used to label images as relevant and irrelevant, and then this information was used as positive and negative feedback for iterative searches. Each curve labeled as "n feedback" within each plot shows precision and recall for the n'th feedback iteration for a particular feedback model.



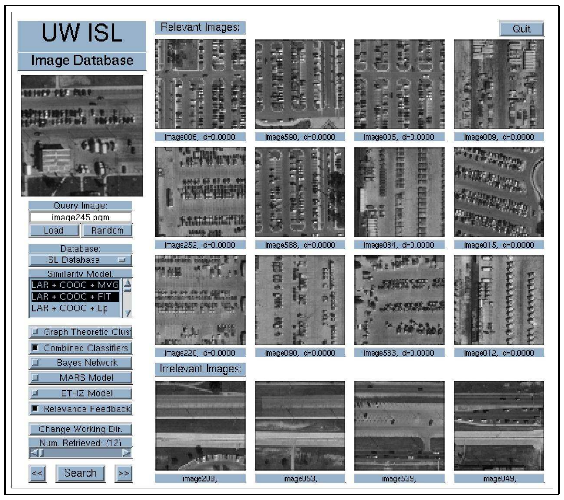
Query Image: image245.pgm
Database: ISL Database
Num. Retrieved: (12)

(a) Using only Gabor features (3/12)



Query Image: image245.pgm
Database: ISL Database
Num. Retrieved: (12)

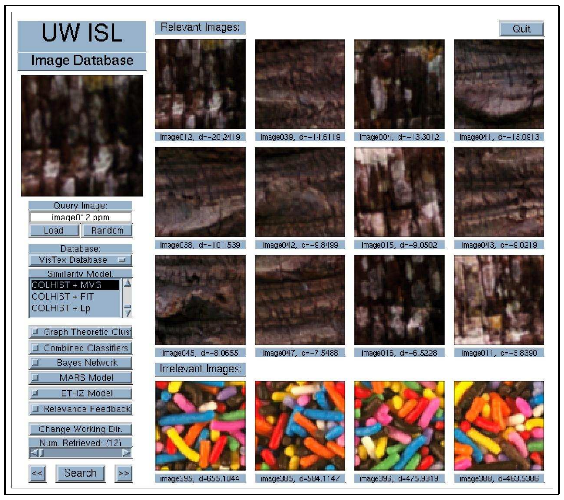
(b) Using four texture features (no feedback) (6/12)



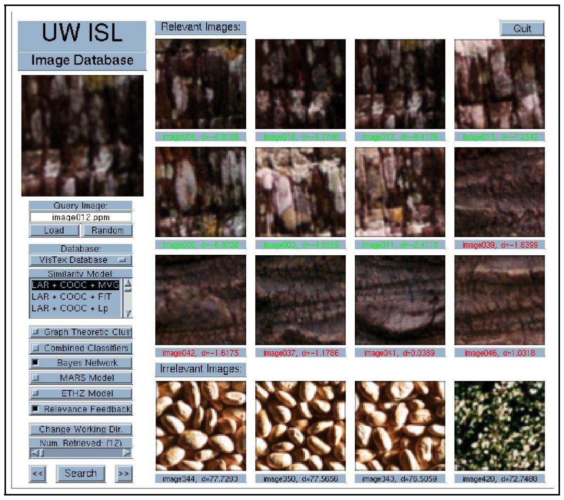
Query Image: image245.pgm
Database: ISL Database
Num. Retrieved: (12)

(c) After first feedback (12/12)

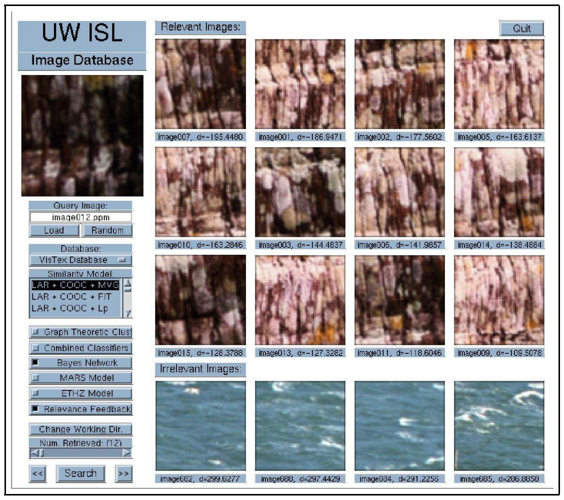
Figure 8.15: An example query for parking lots from the ISL Database. Only Gabor features were used for the first search. Then, four texture features were used together using the combined classifiers framework. Green labels under each image show images that were marked as relevant by the user. Red labels show images that were marked as irrelevant by the user. These images were used as feedback data for the following iteration. The numbers in parentheses in sub-captions show the number of correct matches for each case.



(a) Using only color histograms (5/12)

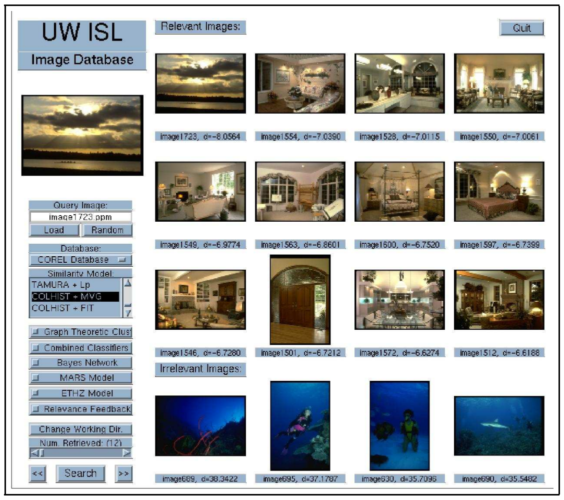


(b) Using color and texture (no feedback) (7/12)

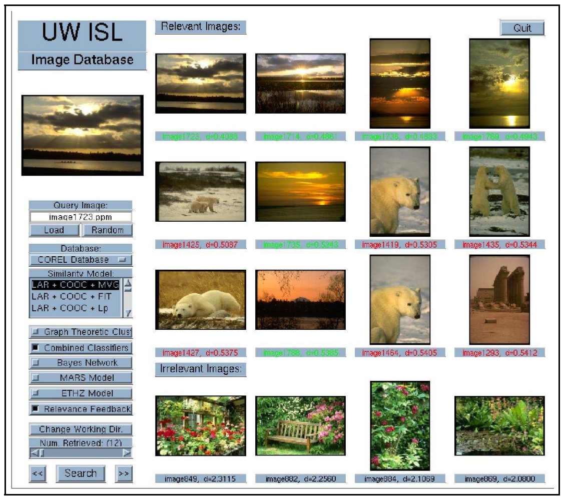


(c) After first feedback (12/12)

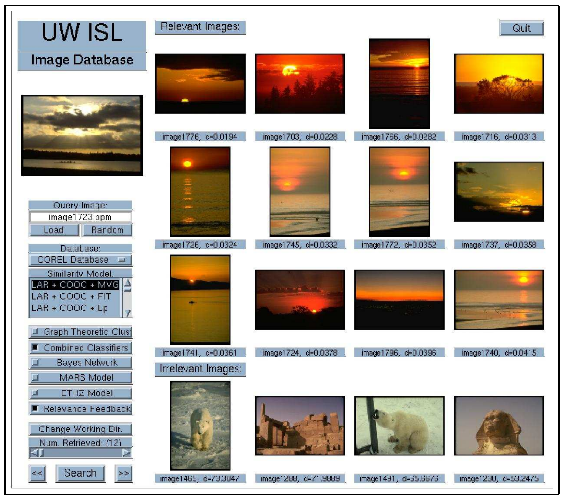
Figure 8.16: An example query for bark from the VisTex Database. Bayesian network was used for combination and feedback. The numbers in parentheses in sub-captions show the number of correct matches for each case.



(a) Using only color histograms (1/12)

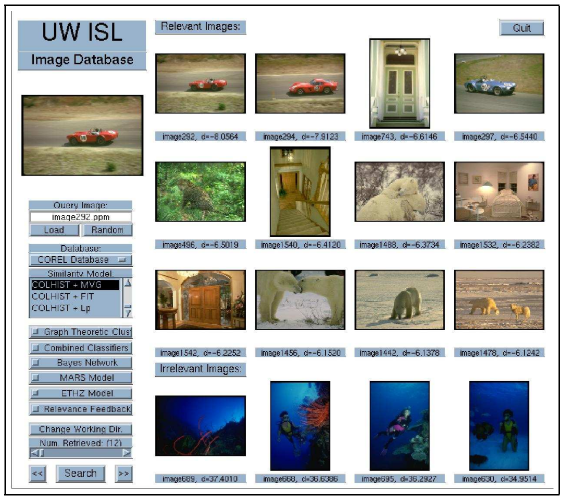


(b) Using color and texture (no feedback) (6/12)



(c) After first feedback (12/12)

Figure 8.17: An example query for sunsets from the COREL Database. Combined classifiers were used for combination and feedback. The numbers in parentheses in sub-captions show the number of correct matches for each case.

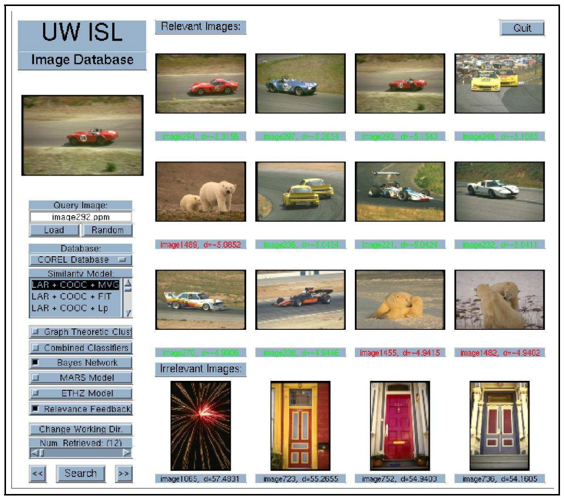


Query Image: image292.ppm
Database: COREL Database
Similarity Model: COLHIST + MVG, COLHIST + FIT, COLHIST + Lp
Num. Retrieved: (12)

image292, d=-8.0564 image294, d=-7.9123 image743, d=-6.6146 image297, d=-6.5440
image496, d=-6.5019 image1540, d=-6.4120 image1488, d=-6.3734 image1532, d=-6.2382
image1542, d=-6.2252 image1456, d=-6.1520 image1442, d=-6.1378 image1478, d=-6.1242

image689, d=37.4010 image668, d=36.6386 image695, d=36.2927 image630, d=34.9514

(a) Using only color histograms (3/12)

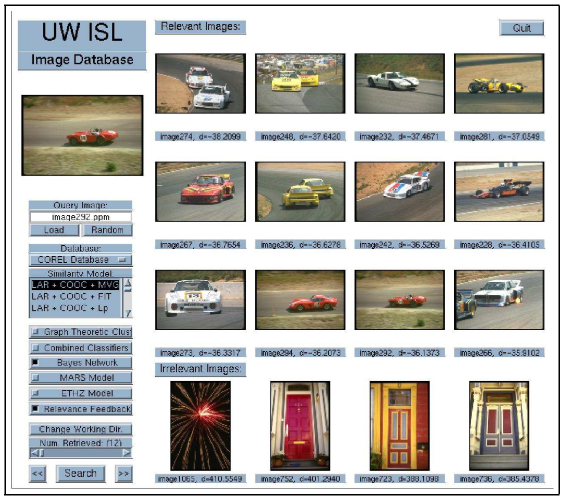


Query Image: image292.ppm
Database: COREL Database
Similarity Model: LAR + COOC + MVG, LAR + COOC + FIT, LAR + COOC + Lp
Num. Retrieved: (12)

image294, d=-5.3156 image297, d=-5.2654 image292, d=-5.1543 image248, d=-5.1085
image1489, d=-5.0852 image236, d=-5.0454 image221, d=-5.0424 image232, d=-5.0411
image270, d=-4.9906 image228, d=-4.9446 image1455, d=-4.9415 image1482, d=-4.9402

image1065, d=57.4831 image723, d=55.2655 image752, d=54.9403 image736, d=54.1605

(b) Using color and texture (no feedback) (9/12)



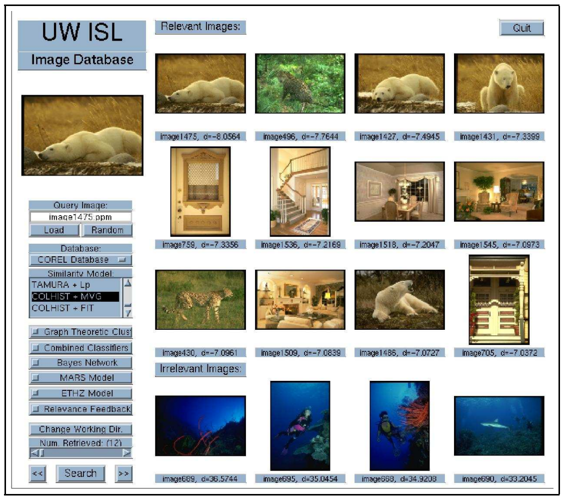
Query Image: image292.ppm
Database: COREL Database
Similarity Model: LAR + COOC + MVG, LAR + COOC + FIT, LAR + COOC + Lp
Num. Retrieved: (12)

image274, d=-38.2099 image248, d=-37.6420 image232, d=-37.4671 image281, d=-37.0549
image267, d=-36.7654 image236, d=-36.6278 image242, d=-36.5269 image228, d=-36.4105
image273, d=-36.3317 image294, d=-36.2073 image292, d=-36.1373 image266, d=-35.9102

image1065, d=410.5549 image752, d=401.2940 image723, d=388.1098 image736, d=385.4378

(c) After first feedback (12/12)

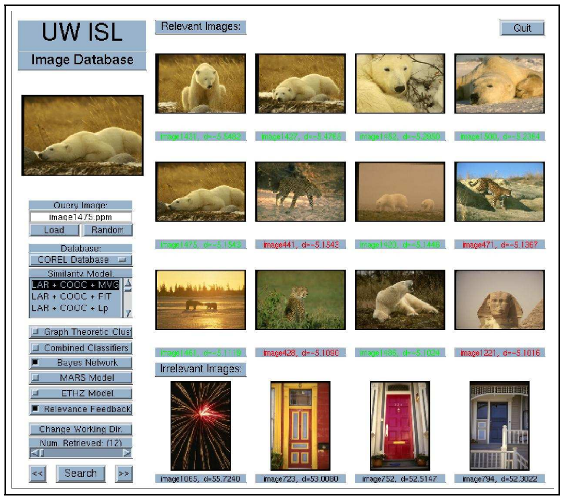
Figure 8.18: An example query for auto racing from the COREL Database. Bayesian network was used for combination and feedback. The numbers in parentheses in sub-captions show the number of correct matches for each case.



Query Image: image1475.ppm
Database: COREL Database
Similarity Model: TAMURA + Lp / COLHIST + MVG / COLHIST + FIT
Num. Retrieved: (12)

image1475, d=-8.0564 image496, d=-7.7644 image1427, d=-7.4945 image1431, d=-7.3399
image759, d=-7.3356 image1536, d=-7.2169 image1518, d=-7.2047 image1545, d=-7.0973
image430, d=-7.0961 image1509, d=-7.0839 image1486, d=-7.0727 image705, d=-7.0372
image689, d=36.5744 image695, d=35.0454 image668, d=34.9208 image690, d=33.2045

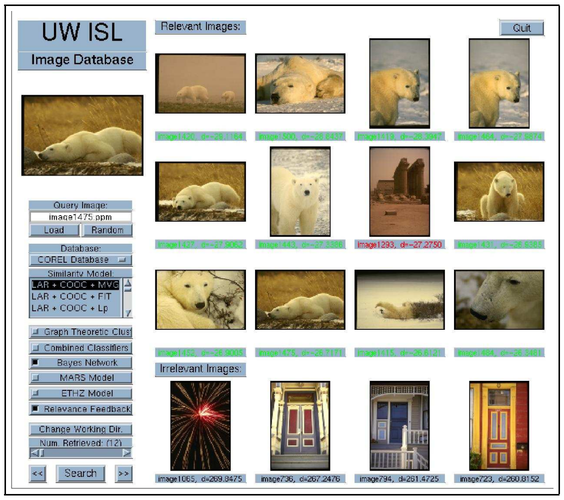
(a) Using only color histograms (4/12)



Query Image: image1475.ppm
Database: COREL Database
Similarity Model: LAR + COOC + MVG / LAR + COOC + FIT / LAR + COOC + Lp
Num. Retrieved: (12)

image1431, d=-5.5482 image1427, d=-5.4765 image1452, d=-5.2950 image1500, d=-5.2364
image1475, d=-5.1543 image441, d=-5.1543 image1420, d=-5.1446 image471, d=-5.1367
image1461, d=-5.1119 image428, d=-5.1090 image1486, d=-5.1024 image1221, d=-5.1016
image1065, d=55.7240 image723, d=53.0080 image752, d=52.5147 image794, d=52.3022

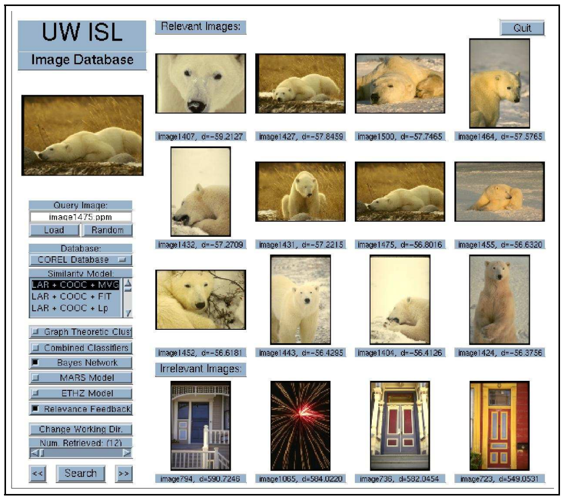
(b) Using color and texture (no feedback) (8/12)



Query Image: image1475.ppm
Database: COREL Database
Similarity Model: LAR + COOC + MVG / LAR + COOC + FIT / LAR + COOC + Lp
Num. Retrieved: (12)

image1420, d=-29.1164 image1500, d=-28.8437 image1419, d=-28.3947 image1464, d=-27.9874
image1427, d=-27.9062 image1443, d=-27.3386 image1293, d=-27.2750 image1431, d=-26.9385
image1452, d=-26.9005 image1475, d=-26.7171 image1415, d=-26.6121 image1484, d=-26.3481
image1065, d=269.8475 image736, d=267.2476 image794, d=261.4725 image723, d=260.8152

(c) After first feedback (11/12)

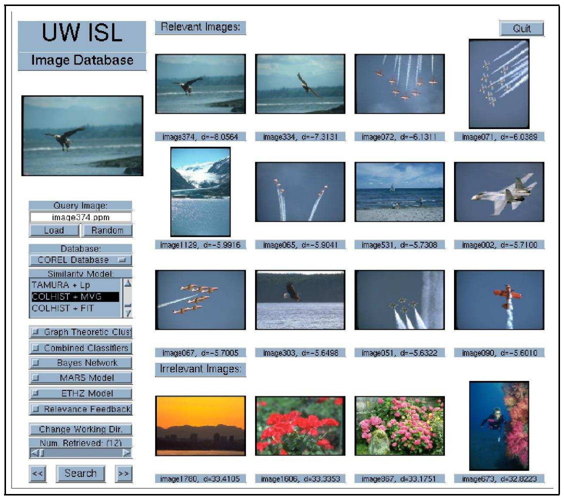


Query Image: image1475.ppm
Database: COREL Database
Similarity Model: LAR + COOC + MVG / LAR + COOC + FIT / LAR + COOC + Lp
Num. Retrieved: (12)

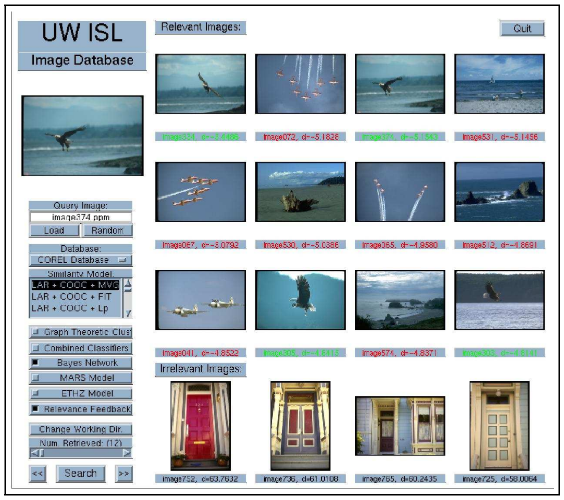
image1407, d=-59.2127 image1427, d=-57.8459 image1500, d=-57.7465 image1464, d=-57.5765
image1432, d=-57.2709 image1431, d=-57.2215 image1475, d=-56.8016 image1455, d=-56.6320
image1452, d=-56.6181 image1443, d=-56.4295 image1404, d=-56.4126 image1424, d=-56.3756
image794, d=590.7246 image1065, d=584.0220 image736, d=582.0454 image723, d=549.0531

(d) After second feedback (12/12)

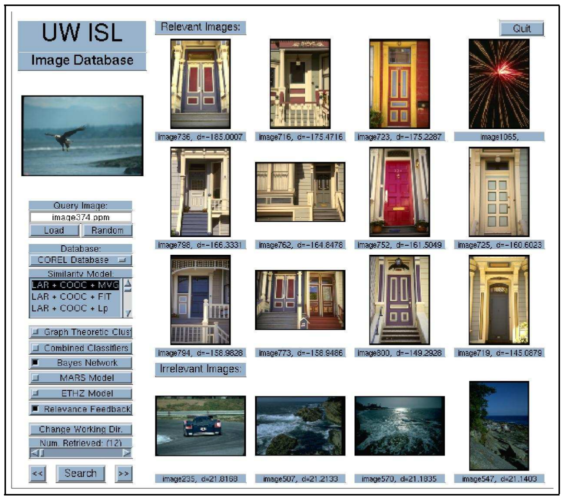
Figure 8.19: An example query for polar bears from the COREL Database. Bayesian network was used for combination and feedback. The numbers in parentheses in sub-captions show the number of correct matches for each case.



(a) Using only color histograms (3/12)

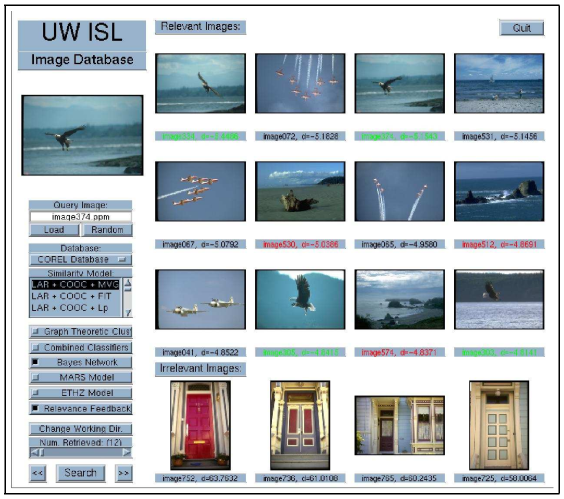


(b) Using color and texture (no feedback) (4/12)

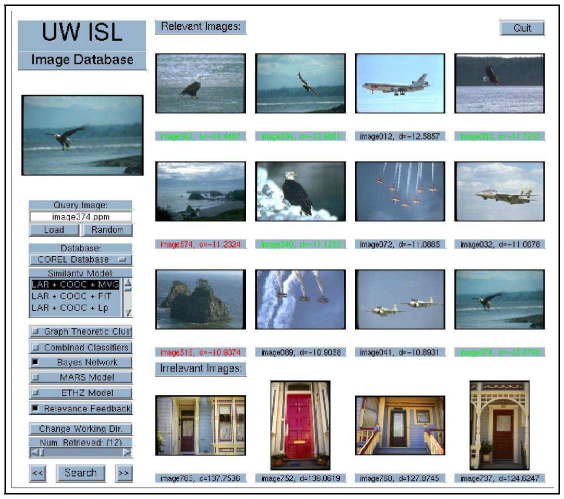


(c) After first feedback using all non-eagle (coast and air shows) images as negative feedback (0/12)

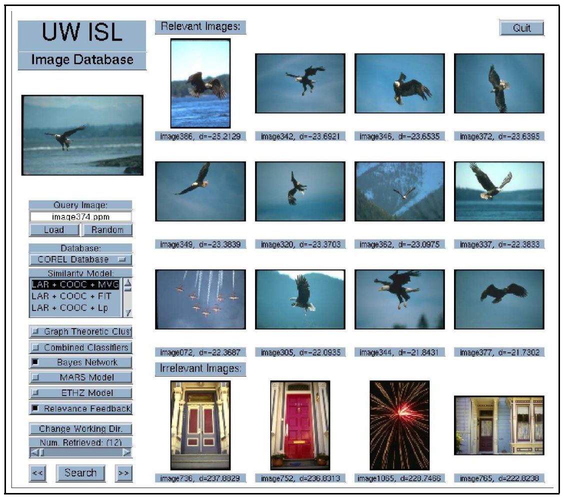
Figure 8.20: An example query for bald eagles from the COREL Database. Bayesian network was used for combination and feedback. The numbers in parentheses in sub-captions show the number of correct matches for each case.



(a) Using color and texture (no feedback) (4/12)



(b) After first feedback using only coast images as negative feedback (5/12)



(c) After second feedback (11/12)

Figure 8.21: Same query as in Figure 8.20 for bald eagles from the COREL Database. Bayesian network was used for combination and feedback. The numbers in parentheses in sub-captions show the number of correct matches for each case. See text for a detailed discussion.

form the feedback. Part (b) in Figure 8.20 shows that combining color and texture cannot still prevent some airplane and coast images showing up in the retrieval list for the eagle query. The reason for this is the presence of both sky and sea in the eagle image. We can also see that door images are the worst matches to the eagle. However, giving all eagle images in the relevant image list as positive feedback and all airplane and coast images in the relevant image list as negative feedback modifies the decision boundaries drastically and the door images which were the worst matches in the previous iteration now become the best matches. On the other hand, giving only the coast images as the negative feedback as in Figure 8.21 modifies the decision boundaries slightly so that only eagle and airplane images remain in the relevant image list while still keeping the door images as the worst matches. One more iteration of feedback can now result in 11 of the 12 best matches to be eagles.

Both visual examples as qualitative performance evaluation and precision-recall curves as quantitative performance evaluation show that the Bayesian combination methods give significant improvements over using only one feature vector, and furthermore, incorporating relevance feedback in the Bayesian framework achieves almost perfect retrieval. However, there were still some images that caused error cases. Figures 8.22, 8.23 and 8.24 show the images for which the Bayesian framework could not achieve a "perfect" retrieval[2] after at most 5 iterations. Out of 600 images in the ISL Database, the combined classifiers model could not perform perfectly for only 2 images and the naive Bayesian network model could not perform perfectly for only 7 images. These 7 images are shown in Figure 8.22. On the other hand, the MARS and ETHZ models could not achieve a perfect retrieval for 89 and 153 images respectively. Out of 736 images in the VisTex Database, the combined classifiers model could not perform perfectly for only 4 images and the naive Bayesian network model could not perform perfectly for only 1 image. These 4 images are shown in Figure

[2]In the visual examples we define perfect retrieval to be the case where all of the 12 best matches to a query image belong to the same category as the query image.

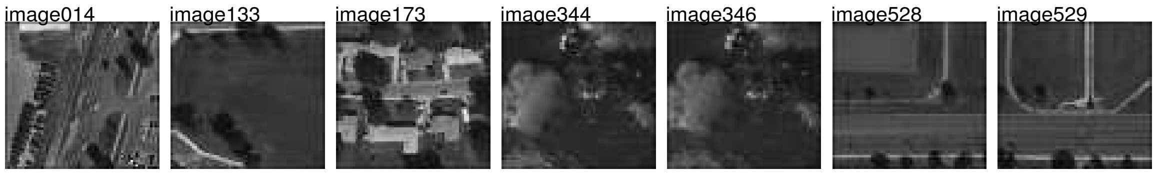


Figure 8.22: ISL Database images for which the Bayesian framework could not achieve a "perfect" (12/12) retrieval.

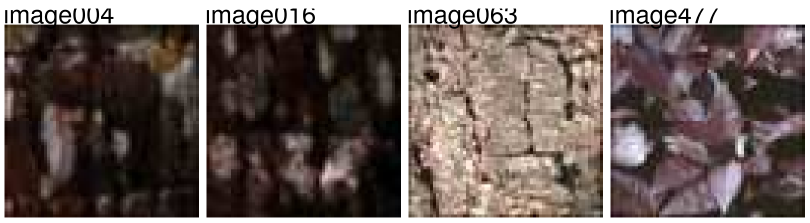


Figure 8.23: VisTex Database images for which the Bayesian framework could not achieve a "perfect" (12/12) retrieval.

8.23. On the other hand, the MARS and ETHZ models could not achieve a perfect retrieval for 177 and 157 images respectively. Out of 1,575 images in the COREL Database, the combined classifiers model could not perform perfectly for 100 images and the naive Bayesian network model could not perform perfectly for 212 images. 76 of these images were common and they are shown in Figure 8.24. On the other hand, the MARS and ETHZ models could not achieve a perfect retrieval for 780 and 662 images respectively. These results are summarized in Table 8.9.

8.6 Summary of Observations

A summary of observations from the experiments presented in this chapter is given below.

- Classification effectiveness in terms of minimizing the classification error in our two-class problem reflects well on the retrieval performance in terms of precision

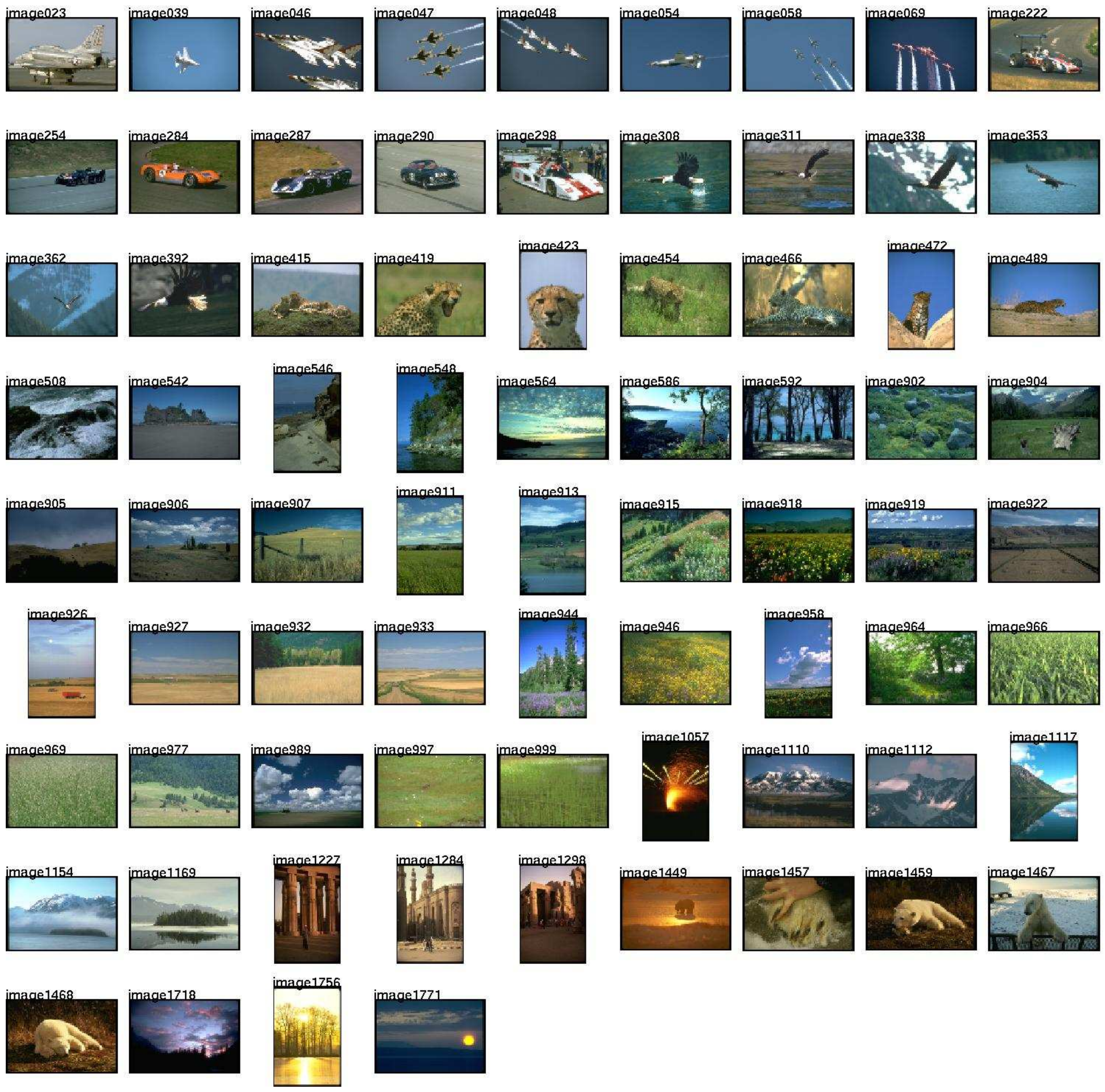


Figure 8.24: COREL Database images for which the Bayesian framework could not achieve a “perfect” (12/12) retrieval.

Table 8.9: Number of images where a “perfect” (12/12) retrieval could not be achieved after 5 feedback iterations.

Database	Total number of images	Error images for Bayesian	Error images for MARS	Error images for ETHZ
ISL	600	7	89	153
VisTex	736	4	177	157
COREL	1,575	212	780	662

and recall. The models that performed the best in classification and retrieval were consistent, as in the experiments presented in Chapter 4.

- Combining the decisions made by classifiers that were trained on different feature vectors significantly improved both classification and retrieval performances over the cases where classifiers were used individually.

- The most successful classifier combination rule was the product rule for the ISL and VisTex Databases and the max rule for the COREL Database. The most successful classifiers were the Logistic linear and Gaussian quadratic classifiers. They gave both the smallest classification error and the highest precision and recall.

- Classifiers trained on Gabor and color histogram feature vectors and the multivariate Gaussian model performed better than other models.

- Combining the outputs of all classifiers for all feature vectors and similarity models did not give much improvement and was not worth the computation. Although adding more classifiers improved the performance, blindly adding classifiers increased the chance of some highly correlated unsuccessful classifiers dominating the results. The Q statistics which measure the amount of correlation in the decisions of pairs of classifiers can be useful to choose the classifiers that will be used in the combination.

- The Bayesian retrieval algorithms achieved almost perfect retrieval for the VisTex Database which has been used as the test dataset in most of the content-based image retrieval papers. The performance improvements for both the ISL Database and the COREL Database were also very significant. Both the combined classifiers-based combination and the Bayesian network with the naive Bayes classifier structure performed significantly better than the MARS model from the University of Illinois and the discrete variable naive Bayesian network model from the Swiss Federal Institute of Technology.

- The proposed naive Bayesian network model resulted in 1.31% relative improvement in precision for the ISL Database, 7.14% relative improvement in precision for the VisTex Database and 16.03% relative improvement in precision for the COREL Database (computed at the knee of the precision vs. recall curves) over the cases where the best feature vectors were used individually. The methods described in Chapter 7 can be used to further refine the probability estimates. A more complex Bayesian network can be designed to encode more complex relationships between features and similarity models.

- The proposed Bayesian relevance feedback model was also very effective. We could obtain 99.66% precision for the ISL Database, 99.86% precision for the VisTex Database and 94.21% for the COREL Database when 12 images were retrieved after a few iterations. The Bayesian feedback model was more robust and much more powerful than the feedback models of the two competing algorithms.

- The significant performance of the simple linear classifiers in improving the retrieval results shows the power of the probabilistic framework which simplifies the problem and allows the estimation of less complex models while still being very effective. Both visual examples as qualitative performance evaluation and precision-recall curves as quantitative performance evaluation show that the Bayesian combination methods give significant improvements over using only one feature vector, and furthermore, incorporating relevance feedback in the Bayesian framework achieves almost perfect retrieval.

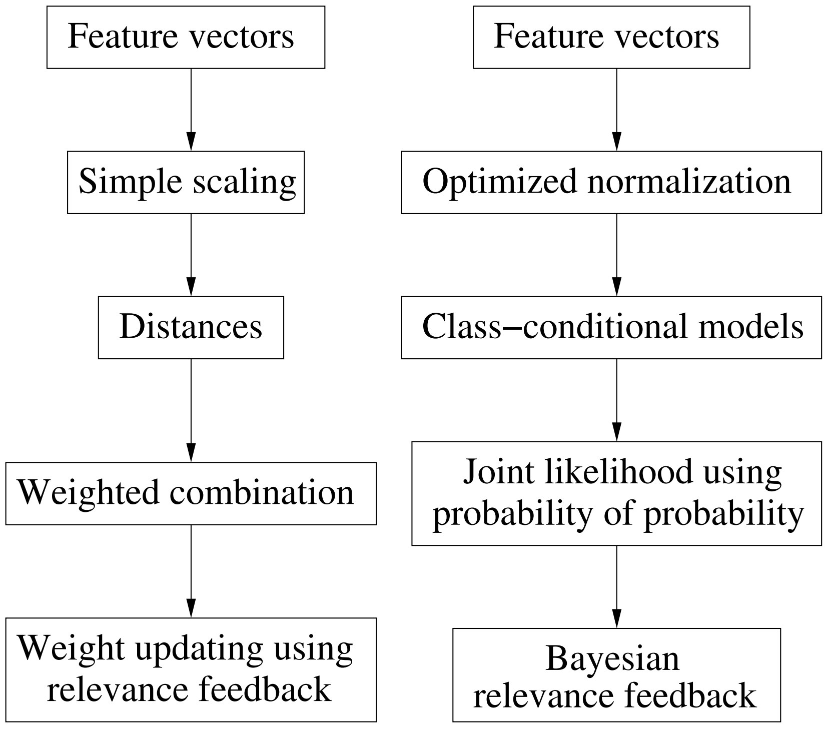
Chapter 9

CONCLUSIONS AND FUTURE WORK

9.1 Summary and Conclusions

Content-based image retrieval (CBIR) has become one of the most popular research areas in computer vision. The retrieval process can be divided into three levels; the pre-processing level (feature extraction and normalization), the similarity level (similarity computation between the query image and the images in the database), and the post-processing level (iterative retrievals to improve the performance). In the rapidly growing CBIR literature, there has been an enormous amount of work on developing features for usually restricted domains of images. However, similarity measures have not received significant attention. There is also no generally applicable and effective framework to combine multiple features and similarity measures. The most common approach has been to treat images as points in the feature space and use the nearest neighbor decision rule with geometric distance measures like the Euclidean distance to measure image similarities. Besides, most of the algorithms and decision criteria are developed by trial and error thresholds with insufficient performance evaluation that use only a few examples. The commonly used geometric framework is summarized in Figure 9.1(a).

In this dissertation, our goal has been to design each level of the content-based retrieval process with a well-defined formulation. Furthermore, we attempted to provide a solution to the challenging problem of combining decisions based on multiple feature vectors and similarity models, which also has a high potential of giving a big improvement in retrieval performance.



(a) Geometric framework (b) Probabilistic framework

Figure 9.1: Main steps of processing in the commonly used geometric framework and the proposed probabilistic framework.

We posed the retrieval problem in a classification framework where the goal was to minimize the classification error in a setting of two classes: the relevance class and the irrelevance class. When the problem became the minimization of the classification error, the obvious choice was the Bayes classifier which is known to give the theoretical minimum. Since it uses the posterior probabilities to make the decision, the posteriors were the ideal features for classification. This setting could be interpreted as a mapping from the high-dimensional feature space to the two-dimensional probability space. Therefore, we proposed models to compute the posterior probabilities or other classification information to find solutions to the three levels of the retrieval problem. However, these posterior probabilities also had uncertainty due to factors like imperfect density modeling in the feature space, quantization, high dimensionality, etc. To model these uncertainties as “probability of probability”, we proposed a two-level modeling where the first level included models to compute probabilities of feature vectors, and the second level included models to compute probabilities of these probabilities to compensate for errors in modeling in the first level. Given the posterior probabilities for the relevance and irrelevance classes, similarity could then be computed as likelihood in the probabilistic setting instead of computing distances in the geometric setting.

Given multiple feature vectors to measure different color and texture properties for each image, our solution for the pre-processing level (feature normalization) was to choose the normalization method according to a class separability criterion. We studied five normalization methods; linear scaling to unit range, linear scaling to unit variance, transformation to a uniform random variable using the cumulative distribution function, rank normalization and normalization by fitting distributions. Even though there was no single best normalization method for all databases, normalization after fitting distributions was usually among the best and class separability proved effective for choosing the normalization method that gave the best retrieval performance.

In the similarity level, we developed probabilistic similarity measures that computed the likelihood of two images being similar or dissimilar, one being the query image and the other one being an image in the database. We also studied the effects of operating in the probability space versus operating in the feature space. The simplification of the problem by doing the estimation and making the decision in the two-dimensional probability space allowed us to effectively train simple linear classifiers while complex non-linear classifiers had to be developed in the high-dimensional feature space. Furthermore, these linear classifiers gave smaller classification errors than the non-linear classifiers. The comparisons between classification performances in the probability space and the feature space, and retrieval performances of the probabilistic similarity measures and the geometric similarity measures showed that our probabilistic framework performed significantly better than the commonly used geometric framework.

Given multiple classifiers trained on different feature vectors and similarity models, classifier combination rules from the pattern recognition literature were used to combine the decisions made by individual classifiers to obtain a final measure of similarity. In addition, uncertainties in the estimation of the probabilities and likelihood values from noisy data were further incorporated in a naive Bayesian network framework. All of these solutions were very effective and efficient by using simple models like univariate or multivariate Gaussians. A more complex Bayesian network can be designed to encode more complex relationships between features and similarity models.

Our solutions for the post-processing level (iterative retrievals) can be divided into three parts. The first two parts, graph-theoretic clustering for image grouping and a weighted distance approach for relevance feedback operated on a single feature vector for each image. To address a common problem in retrieval results that sometimes images that were quite irrelevant to the query image were also retrieved simply because they were close to it in the feature space, we developed a graph-theoretic approach for image grouping and retrieval by formulating the database search as a graph clustering problem using a constraint that the retrieved images should be consistent with each other as well as being similar to the query image in the feature space. We also described a model to estimate the probability of each image being relevant to the query image under the graph-theoretic framework.

In the second part for post-processing, we used a weighted distance formulation to include the user in the retrieval loop via positive and negative labeling of the retrieved images. After formulating the weight updating problem in an estimation and regression framework, we computed the optimum weights that were used to iteratively refine the effects of different feature components in the database search.

As the third and the main part of post-processing, the probabilistic framework was extended to support relevance feedback. Given the images that were labeled by the user as relevant and irrelevant, a Bayesian formulation was used to update the posterior probabilities for two images being relevant or irrelevant. In effect, this formulation used multiple images as the query and multiple measurements for each image to compute the similarity. The ratios of these updated posteriors were used in iterative retrievals to rank database images according to their similarities to the query image. The proposed probabilistic framework is summarized in Figure 9.1(b).

Testing content-based image retrieval systems and comparing their performances is still an open question. In most of the content-based retrieval literature, researchers presented example queries to visually evaluate the performance of their systems. Performance evaluation was an important part of this dissertation. We used three groundtruth databases, namely the ISL Database that included aerial and satellite images, the VisTex Database that included images with relatively homogeneous textures, and the COREL Database that included images from a stock photo library. The performance of each proposed method was evaluated using quantitative criteria like precision, recall, misdetection and false alarm, and qualitative criteria in terms of visual examples. The results were also compared to those of the commonly used geometric framework and two competing algorithms from the CBIR literature (implementation details given in Section 8.4). The proposed framework achieved a better performance than the best performing competitor in each level. In particular, we obtained 8-20% relative improvement[1] in precision in the similarity level where multiple feature vectors were combined, and 4-14% relative improvement in precision after relevance feedback. In the final level, we could obtain 99.66% precision for the ISL Database, 99.86% precision for the VisTex Database and 94.21% precision for the COREL Database when 12 images were retrieved after a few feedback iterations. These extensive experiments showed that the probabilistic framework allowed an effective way of combining feature vectors and similarity measures with incorporated relevance feedback. Therefore, one can do the design in the classification framework and expect better results in retrieval.

9.2 Future Work

We showed that a probabilistic framework can give significant improvements in the CBIR performance. In the current setting, our probabilistic models use only low-level visual features as the source of information. However, the proposed feature and similarity combination algorithms use only the class-conditional probabilities and do not directly depend on the low-level features in the sense that any model that has an image as an input and the class-conditional or posterior probabilities as the output can be included in the Bayesian framework.

A promising research direction is to find probabilistic models for higher-level image features, for example object features and their spatial relationships. Requirements for segmentation [89, 90, 183] accuracy are quite different for precise object recognition problems and region-based image retrieval problems. An accurate segmentation is highly desirable for the former, while a coarse segmentation may be sufficient for

[1]Relative improvement is computed as $\frac{\textit{new value - old value}}{\textit{old value}} \times 100$.

the latter. Since most of the complex objects and scenes do not consist of a single homogeneous region, we can use regions of locally homogeneous features and their combination for representation. When feature measurements are computed for each region in the image, the relationships between them can be encoded in graph structures and graph matching algorithms can be used to find matches between objects or scenes. Relational matching have been extensively studied for structural pattern recognition. Shapiro and Haralick [182] described a relational distance measure that was a metric and argued that class-conditional probabilities could be defined as monotonic decreasing functions of the distance between two relational descriptions. Matas *et al.* [139] developed a color adjacency graph which combined the advantages of both histograms and the region adjacency graph representation. Christmas *et al.* [46] described a probabilistic relaxation technique for relational graph matching. The Bayesian edit distance in [148] supported matchings between corrupted relational graphs. Fuh *et al.* [77] described a relationship tree where parent-child relationships represented sub-regions in a region growing-based segmentation process. The dissimilarity score between two trees was the summation of the dissimilarity scores between each corresponding region pair in the trees. These kinds of algorithms can be either directly used or modified to output the strength of the match as a probability and can therefore be included in our Bayesian framework.

Information about an image can come from a number of different sources: the image content, keywords attached to the image, and text surrounding the image (e.g. captions and HTML tags) [187]. Even though obtaining keyword information for images may not be always feasible because of the requirement of an enormous amount of human involvement during manual annotation, keywords can also be quite useful if they are available. Information like time and place of image capture as well as special names for objects or scenes in the image most probably cannot be captured by visual information alone. For example, a query for “sunrise in Seattle” can be made feasible by searching among the images with an associated keyword “Seattle”. Information retrieval literature provides many algorithms for probabilistic modeling of relevancies between documents indexed by keywords [175, 115] and the Bayesian setting is a promising framework to incorporate this information. Furthermore, the probabilistic framework can help the database designer assign keywords to the images. If there are some database images that already have labels assigned to them, the probabilistic measures can be used to propagate those labels to new images based on the likelihood values, and therefore facilitate automatic keyword assignment.

Another popular and increasingly common source of information is video [33]. It can be considered as the next dimension after an image. It is much larger in size and is also more complex. However, it also has its own advantages. One advantage is the temporal information. For example, we can get more information about the importance of objects in the video by looking at their or camera's motion. We can guess what is the main point of interest and what is the background. The MPEG-4 standard has been developed to separately code multiple video objects. Video also contains audio. Integration of audio and speech recognition [118] with visual information will help improve the overall quality of recognition and will give us information that cannot be captured with images alone.

The Moving Picture Experts Group (MPEG) of the International Organization of Standardization (ISO) is currently developing the MPEG-7 standard which aims at providing standardized core technologies to allow description of audiovisual data content in multimedia environments [137, 138]. The other MPEG standards, MPEG-1, -2 and -4 are designed to represent the information itself, while MPEG-7 describes how to represent information about the information. Therefore, the MPEG-7 standard will not standardize the feature extraction and searching methods but will standardize the description of various types of multimedia content that may include [137]:

- information describing the creation and production processes of the content (director, title, short feature movie),

- information related to the usage of the content (copyrights, usage history, broadcast schedule),
- information of the storage of the content (storage format, encoding),
- structural information on spatial, temporal or spatio-temporal components of the content (scene cuts, segmentation in regions, region motion tracking),
- information about low-level features in the content (colors, textures, sound timbres, melody description), and
- conceptual information of the reality captured by the content (objects and events, interactions among objects).

Therefore, new methods have to be developed to extract information from data at different levels of abstraction. Information fusion is imperative for improving retrieval performance and building practical systems for browsing, searching and retrieving multimedia data, and the probability theory is a strong candidate for doing that.

BIBLIOGRAPHY

[1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* National Bureau of Standards, 1972.

[2] S. Aksoy. Textural features for content-based image database retrieval. Master's thesis, University of Washington, Seattle, WA, June 1998. Online: http://isl.wtc.washington.edu/~aksoy/thesis.shtml.

[3] S. Aksoy and R. M. Haralick. Content-based image database retrieval using variances of gray level spatial dependencies. In *Proceedings of IAPR International Workshop on Multimedia Information Analysis and Retrieval*, pages 3–19, Hong Kong, August 13–14 1998. as Lecture Notes in Computer Science, vol. 1464.

[4] S. Aksoy and R. M. Haralick. Textural features for image database retrieval. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries, in conjunction with CVPR'98*, pages 45–49, Santa Barbara, CA, June 1998.

[5] S. Aksoy and R. M. Haralick. A graph–theoretic approach to image database retrieval. In *Proceedings of the Third International Conference on Visual Information Systems*, pages 341–348, Amsterdam, The Netherlands, June 2–4 1999.

[6] S. Aksoy and R. M. Haralick. Graph–theoretic clustering for image grouping and retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 63–68, Colorado, June 1999.

[7] S. Aksoy and R. M. Haralick. Using texture in image similarity and retrieval. In *Proceedings of the Intl. Workshop on Texture Analysis in Machine Vision*, pages 111–117, Oolu, Finland, June 14–15 1999.

[8] S. Aksoy and R. M. Haralick. Probabilistic vs. geometric similarity measures for image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 357–362, Hilton Head Island, South Carolina, June 2000.

[9] S. Aksoy and R. M. Haralick. Using texture in image similarity and retrieval. In M. Pietikainen, editor, *Texture Analysis in Machine Vision*, volume 40 of *Series in Machine Perception and Artificial Intelligence*, pages 129–149. World Scientific, 2000.

[10] S. Aksoy and R. M. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5):563–582, May 2001.

[11] S. Aksoy, R. M. Haralick, F. A. Cheikh, and M. Gabbouj. A weighted distance approach to relevance feedback. In *Proceedings of 15th IAPR International Conference on Pattern Recognition*, Barcelona, Spain, September 2000.

[12] D. Androutsos, K. N. Plataniotis, and A. N. Venetsanopoulos. A novel vector-based approach to color image retrieval using a vector angular-based distance measure. *Computer Vision and Image Understanding, Special Issue on Content-Based Access of Image and Video Libraries*, 75(1/2):46–58, July/August 1999.

[13] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and semi-automatic methods for image annotation

and retrieval in QBIC. In *SPIE Storage and Retrieval of Image and Video Databases*, volume 2420, pages 24–35, 1995.

[14] Association for Uncertainty in Artificial Intelligence. Online: http://www.auai.org.

[15] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C.-F. Shu. The Virage search engine: An open framework for image management. In *SPIE Storage and Retrieval of Image and Video Databases*, February 1996.

[16] J. R. Bach, S. Paul, and R. Jain. A visual information management system for the interactive retrieval of faces. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):619–628, 1993.

[17] J. Barros, J. French, W. Martin, and P. Kelly. System for indexing multispectral satellite images for efficient content based retrieval. In *SPIE Storage and Retrieval of Image and Video Databases III*, February 1995.

[18] J. Barros, J. French, W. Martin, P. Kelly, and M. Cannon. Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval. In *SPIE Storage and Retrieval of Image and Video Databases IV*, January 1996.

[19] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using EM and its application to content-based image retrieval. In *Proceedings of IEEE International Conference on Computer Vision*, 1998.

[20] A.B. Benitez, M. Beigi, and S.F. Chang. Using relevance feedback in content-based image metasearch. *IEEE Internet Computing*, 2(4):59–69, July-August 1998.

[21] A. P. Berman and L. G. Shapiro. Efficient image retrieval with multiple distance measures. In *SPIE Storage and Retrieval of Image and Video Databases*, pages 12–21, February 1997.

[22] A. P. Berman and L. G. Shapiro. A flexible image database system for content-based retrieval. In *Proceedings of 14th IAPR International Conference on Pattern Recognition*, volume 1, pages 894–898, Brisbane, Australia, August 16–20 1998.

[23] A. P. Berman and L. G. Shapiro. Selecting good keys for triangle-inequality-based pruning algorithms. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Databases*, pages 12–18, January 1998.

[24] A. P. Berman and L. G. Shapiro. A flexible image database system for content-based retrieval. *Computer Vision and Image Understanding, Special Issue on Content-Based Access of Image and Video Libraries*, 75(1/2):175–195, July/August 1999.

[25] J.N. Bhuyan, J.S. Deogun, and V.V. Raghavan. Cluster-based adaptive information retrieval. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, volume 1, pages 307–316, 1991.

[26] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models. Technical Report TR-97-021, International Computer Science Institute, University of California, Berkeley, April 1998.

[27] J. A. Bilmes and K. Kirchhoff. Directed graphical models of classifier combination: Application to phone recognition. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Beijing, October 2000.

[28] A. Del Bimbo, P. Pala, and S. Santini. Visual image retrieval by elastic deformation of object sketches. In *IEEE Symposium on Visual Languages*, pages 216–223, 1994.

[29] P.V. Biron and D.H. Kraft. Sub-symbolic approaches to information retrieval. In *Intelligent Systems for the 21st Century., IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3567 –3572, 1995.

[30] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[31] G. Borgefors. Hierarchical Chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[32] M. Bouet and C. Djeraba. Visual content based retrieval in an image database with relevant feedback. In *Proceedings of International Workshop on Multimedia Database Management Systems*, pages 98–105, 1998.

[33] R. Brunelli, O. Mich, and C. M. Modena. A survey on the automatic indexing of video data. *Journal of Visual Communication and Image Representation*, 10(2):78–112, June 1999.

[34] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, December 1994.

[35] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, April 1996.

[36] K. V. Bury. *Statistical Models in Applied Science.* John Wiley & Sons, Inc., 1975.

[37] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[38] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1997.

[39] G. Casella and R. L. Berger. *Statistical Inference.* Duxbury Press, California, 1990.

[40] G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28:781–793, 1995.

[41] F. A. Cheikh, B. Cramariuc, C. Reynaud, M. Qinghong, B. D. Adrian, B. Hnich, M. Gabbouj, P. Kerminen, T. Makinen, and H. Jaakkola. MUVIS: A system for content-based indexing and retrieval in large image databases. In *SPIE Storage and Retrieval of Image and Video Databases VII*, pages 98–106, San Jose, CA, January 1999.

[42] Jau-Yuen Chen, C.A. Bouman, and J.C. Dalton. Similarity pyramids for browsing and organisation of large image databases. In *Proceedings of SPIE/IS&T Conference on Human Vision and Electronic Imaging III*, volume 3299, pages 563–575, San Jose, CA, January 24-30 1998.

[43] Jau-Yuen Chen, C.A. Bouman, and J.C. Dalton. Active browsing using similarity pyramids. In *Proceedings of the VII IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 3656, pages 144–154, San Jose, CA,, January 1999.

[44] Jau-Yuen Chen, C. Taskiran, E.J. Delp, and C.A. Bouman. Vibe: A new paradigm for video database browsing and search. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 96 –100, 1998.

[45] Y.-Y. Chou and L. G. Shapiro. A hierarchical multiple classifier learning algorithm. In *Proceedings of 15th IAPR International Conference on Pattern Recognition*, volume 2, pages 152–155, Barcelona, Spain, September 2000.

[46] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, August 1995.

[47] Tat-Seng Chua, Wai-Chee Low, and Chun-Xin Chu. Relevance feedback techniques for color-based image retrieval. In *Proceedings of Multimedia Modeling*, pages 24–31, 1998.

[48] COREL Photo Stock Library 1. Online: http://www.corel.com/products/clipartandphotos/photos/index.htm.

[49] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The Bayesian image retrieval system, PicHunter: Theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*, 9(1):20–37, January 2000.

[50] I.J. Cox, M.L. Miller, T.P. Minka, and P.N. Yianilos. An optimized interaction strategy for bayesian relevance feedback. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 553–558, June 1998.

[51] I.J. Cox, M.L. Miller, S.M. Omohundro, and P.N. Yianilos. Pichunter: Bayesian relevance feedback for image retrieval. In *Proceedings of the 13th IAPR International Conference on Pattern Recognition*, volume 3, pages 361–369, 1996.

[52] I.J. Cox, T.V. Papathomas, J. Ghosn, P.N. Yianilos, and M.L. Miller. Hidden annotation in content based image retrieval. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 76–81, 1997.

[53] F. Crestani. Comparing neural and probabilistic relevance feedback in an interactive information retrieval system. In *IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on Neural Networks*, volume 5, pages 3426 –3430, 1994.

[54] W.B. Croft. Effective text retrieval based on combining evidence from the corpus and users. *IEEE Expert*, 10:59–63, December 1995.

[55] S. Davies and A. Moore. Mix-nets: Factored mixtures of Gaussians in Bayesian networks with mixed continuous and discrete variables. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[56] M. H. DeGroot. *Optimal Statistical Decisions.* McGraw-Hill, 1970.

[57] J.S. Deogun, V.V. Raghavan, and P. Rhee. Formulation of the term refinement problem for user-oriented information retrieval. In *Proceedings of the Annual AI Systems in Government Conference*, pages 72–78, 1989.

[58] T. G. Dietterich. Machine-learning research, four current directions. *AI Magazine*, 18(4):97–136, 1997.

[59] B. Dom. MDL estimation for small sample sizes and its application to linear regression. Technical Report RJ 10030, IBM Almaden Research Center, San Jose, CA, June 1996.

[60] B. Dom. The minimum description length principle and ill-posed problems in computer vision. Technical Report RJ 10116, IBM Almaden Research Center, San Jose, CA, April 1998.

[61] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of 13th International Conference on Machine Learning*, pages 195–112, Italy, 1996.

[62] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[63] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, Inc., 1973.

[64] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2000.

[65] R. P. W. Duin. Classifiers in almost empty spaces. In *Proceedings of 15th IAPR International Conference on Pattern Recognition*, volume 2, pages 1–7, Barcelona, Spain, September 2000.

[66] R. P. W. Duin. PRTools 3.0, A Matlab toolbox for pattern recognition, 2000. Online: http://www.ph.tn.tudelft.nl/~bob/PRTOOLS.html.

[67] R. P. W. Duin and D. M. J. Tax. Experiments with classifier combining rules. In *Proceedings of First International Workshop on Multiple Classifier Systems*, pages 16–29, Cagliari, Italy, June 2000. as Lecture Notes in Computer Science, vol. 1857.

[68] J. G. Dy, C. E. Brodley, A. Kak, C. Shyu, and L. S. Broderick. The customized-queries approach to CBIR using EM. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 400–406, Colorado, June 1999.

[69] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 2nd edition, 1994.

[70] M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions*. John Wiley & Sons, Inc., 2nd edition, 1993.

[71] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 98–104, Santa Barbara, CA, June 1998.

[72] P. F. Felzenszwalb and D. P. Huttenlocher. Efficiently computing a good segmentation. In *DARPA Image Understanding Workshop*, 1998.

[73] M. M. Fleck, D. A. Forsyth, and C. Pregler. Finding naked people. In *Proceedings of the European Conference on Computer Vision*, pages 593–602. Springer-Verlag, 1996.

[74] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. The QBIC project: Querying images by content using color, texture and shape.

In *SPIE Storage and Retrieval of Image and Video Databases*, pages 173–181, 1993.

[75] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[76] Fort Hood Datasets. Online: http://www.mbvlab.wpafb.af.mil/public/sdms/datasets/fthood/index.htm.

[77] C.-S. Fuh, S.-W. Cho, and K. Essig. Hierarchical color image region segmentation for content-based image retrieval system. *IEEE Transactions on Image Processing*, 9(1):156–162, January 2000.

[78] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[79] D. Geiger and D. Heckerman. Learning Gaussian networks. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 235–243, San Francisco, CA, June 1994. Morgan Kaufmann Publishers. MSR-TR-94-10.

[80] D. Geiger and D. Heckerman. A characterization of the Dirichlet distribution through global and local parameter independence. *The Annals of Statistics*, 25(3):1344–1369, 1997. MSR-TR-94-16.

[81] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[82] T. Gevers and A. W. M. Smeulders. The PicToSeek WWW image search system. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 264–269, Florence, Italy, June 1999.

[83] T. Gevers and A. W. M. Smeulders. PicToSeek: Combining color and shape invariant features for image retrieval. *IEEE Transactions on Image Processing*, 9(1):102–119, January 2000.

[84] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[85] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems. *IEEE Computer Magazine*, 28(9):18–22, September 1995.

[86] N. Haering and N. de Vitoria Lobo. Features and classification methods to locate deciduous trees in images. *Computer Vision and Image Understanding, Special Issue on Content-Based Access of Image and Video Libraries*, 75(1/2):133–149, July/August 1999.

[87] N. Haering, Z. Myles, and N. de Vitoria Lobo. Locating deciduous trees. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1997.

[88] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, May 1979.

[89] R. M. Haralick and L. G. Shapiro. Survey: Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(12):100–132, December 1985.

[90] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.

[91] D. Heckerman. Bayesian networks for knowledge discovery. In U. M. Fayyad,

G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 273–305. AAAI Press, 1996.

[92] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. Kluwer Academic Publishers, 1998. MSR-TR-95-06.

[93] D. Heckerman, A. Mamdani, and M. Wellman, editors. *Communications of the ACM, Special Issue on Real-World Applications of Bayesian Networks*, volume 38, March 1995.

[94] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, January 1994.

[95] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:179–187, February 1962.

[96] J. Huang and D. Mumford. Statistics of natural images and models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 541–547, Colorado, June 1999.

[97] Y. S. Huang and C. Y. Suen. A method for combining multiple classifiers - a neural network approach. In *Proceedings of 12th IAPR International Conference on Pattern Recognition*, volume 2, pages 473–475, 1994.

[98] B. Huet and E. Hancock. Fuzzy relational distance for large-scale object recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 138–143, Santa Barbara, CA, June 1998.

[99] A. Hyvarinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[100] A. Hyvarinen and E. Oja. Independent Component Analysis: Algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000.

[101] A. Hyvarinen, J. Sarela, and R. Vigario. Spikes and bumps: Artefacts generated by Independent Component Analysis with insufficient sample size. In *Proceedings of International Workshop on Independent Component Analysis and Blind Signal Separation*, pages 425–429, France, 1999.

[102] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of SIGGRAPH'95*, pages 277–285, Los Angeles, CA, August 1995.

[103] A. Jain and L. Hong. On-line fingerprint verification. In *Proceedings of the 13th IAPR International Conference on Pattern Recognition*, volume 3, pages 596–600, August 1996.

[104] A. Jain, L. Hong, and Y. Kulkarni. F2ID: A personal identification system using faces and fingerprints. In *Proceedings of 14th IAPR International Conference on Pattern Recognition*, volume 2, pages 1373–1375, Brisbane, Australia, August 16–20 1998.

[105] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, NJ, 1988.

[106] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[107] A. K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29:1233–1244, 1996.

[108] F. V. Jensen. Bayesian network basics. *AISB Quarterly*, 94:9–22, 1996.

[109] F. V. Jensen. *An Introduction to Bayesian Networks.* Springer-Verlag, New York, USA, 1996.

[110] F. V. Jensen and S. L. Lauritzen. Probabilistic networks. In *Handbook of Defeasible and Uncertainty Management Systems*, volume 5: Algorithms for Uncertainty and Defeasible Reasoning, pages 289–320. Kluwer Academic Publishers, 2000.

[111] G. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, 1995.

[112] N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. John Wiley & Sons, Inc., 2nd edition, 1994.

[113] J. W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[114] T. Kanungo, B. Dom, W. Niblack, and D. Steele. A fast algorithm for MDL-based multi-band image segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–616, 1994.

[115] M. Keim. *Bayesian Information Retrieval.* PhD thesis, University of Washington, Seattle, WA, 1997.

[116] P. M. Kelly and T. M. Cannon. CANDID: Comparison algorithm for navigating digital image databases. In *Proceedings of the Seventh International Working Conference on Scientific and Statistical Database Management*, pages 252–258, September 1994.

[117] P. M. Kelly, T. M. Cannon, and D. R. Hush. Query by image example: The CANDID approach. In *SPIE Storage and Retrieval of Image and Video Databases III*, pages 238–248, 1995.

[118] K. Kirchhoff and J. A. Bilmes. Combination and joint training of acoustic classifiers for speech recognition. In *Proceedings of Automatic Speech Recognition*, France, 2000.

[119] J. Kittler, M. Hatef, and R. P. W. Duin. Combining classifiers. In *Proceedings of IAPR International Conference on Pattern Recognition*, volume 2, pages 897–901, 1996.

[120] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[121] J. Kittler and S. A. Hojjatoleslami. A weighted combination of classifiers employing shared and distinct representations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 924–929, 1998.

[122] P. J. Krause. Learning probabilistic networks. Technical report, Philips Research Laboratories, 1998.

[123] R. F. Krichevskiy. Laplace's law of succession and universal encoding. *IEEE Transactions on Information Theory*, 44(1):296–303, January 1998.

[124] V. P. Kumar and U. B. Desai. Image interpretation using bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):74–77, January 1996.

[125] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. W. Duin. Is independence good for combining classifiers? In *Proceedings of 15th IAPR International Conference on Pattern Recognition*, volume 2, pages 168–171, Barcelona, Spain, September 2000.

[126] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

[127] L. Lam and C. Y. Suen. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 27(5):553–568, September 1997.

[128] S. L. Lauritzen. *Graphical Models.* Oxford Statistical Science Series. Clarendon Press, Oxford, UK, 1996.

[129] S. L. Lauritzen and F. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11(2):191–203, April 2001. Research Report R-99-2014, Aalborg University, Denmark.

[130] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice Hall, 1974.

[131] C. S. Li and V. Castelli. Deriving texture set for content based retrieval of satellite image database. In *Proceedings of IEEE International Conference on Image Processing*, pages 576–579, 1997.

[132] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28:84–95, January 1980.

[133] F. Liu and R. W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):722–733, July 1996.

[134] L. Liu and S. Sclaroff. Deformable shape detection and description via model-based region grouping. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 21–27, Colorado, June 1999.

[135] W. Y. Ma and B. S. Manjunath. NETRA: A toolbox for navigating large image databases. In *Proceedings of IEEE International Conference on Image Processing*, 1997.

[136] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[137] J. M. Martinez. Introduction to MPEG-7. Technical Report ISO/IEC JTC1/SC29/WG11 N3545, International Organization for Standardization, Beijing, July 2000.

[138] J. M. Martinez. Overview of the MPEG-7 standard. Technical Report ISO/IEC JTC1/SC29/WG11 N4031, International Organisation for Standardisation, Singapore, March 2001.

[139] J. Matas, R. Marik, and J. Kittler. On representation and matching of multi-coloured objects. In *Proceedings of IEEE International Conference on Computer Vision*, pages 726–732, 1995.

[140] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* John Wiley & Sons, Inc., 1997.

[141] C. Meilhac and C. Nastar. Relevance feedback and category search in image databases. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 512–517, Florence, Italy, June 1999.

[142] H. Melin, J. W. Koolwaaij, J. Lindberg, and F. Bimbot. A comparative evaluation of variance flooring techniques in HMM-based speaker verification. In *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998.

[143] T. P. Minka and R. W. Picard. Interactive learning using a "society of models". *Pattern Recognition Special Issue on Image Databases: Classification and Retrieval*, 1996. Also appears as MIT Media Lab. Perceptual Computing Section Tech. Rep. No: 349, 1995.

[144] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[145] B. Moghaddam, T. Jebera, and A. Pentland. Efficient MAP/ML similarity matching for visual recognition. In *Proceedings of 14th IAPR International Conference on Pattern Recognition*, volume 1, pages 876–881, 1998.

[146] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, July 1997.

[147] K. P. Murphy. Inference and learning in hybrid Bayesian networks. Technical Report CSD-98-990, University of California, Berkeley, January 1998.

[148] R. Myers, R. C. Wilson, and E. R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):628–635, June 2000.

[149] C. Nastar. The image shape spectrum for image retrieval. Technical Report 3206, INRIA Technical Report, France, July 1997.

[150] C. Nastar, M. Mitschke, and C. Meilhac. Efficient query refinement for image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 547–552, Santa Barbara, CA, June 1998.

[151] A. W. Naylor and G. R. Sell. *Linear Operator Theory in Engineering and Science*. Springer-Verlag, New York, 1982.

[152] I.-S. Oh, J.-S. Lee, and C. Y. Suen. Using class separation for feature analysis and combination of class-dependent features. In *Proceedings of IAPR International Conference on Pattern Recognition*, volume 1, pages 453–455, 1998.

[153] I.-S. Oh, J.-S. Lee, and C. Y. Suen. Analysis of class separation and combination of class-dependent features for handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1089–1099, October 1999.

[154] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, NY, 3rd edition, 1991.

[155] Young-Woo Park and Eun-Seok Lee. A new generation method of a user profile for information filtering on the internet. In *Twelfth International Conference on Information Networking*, pages 261 –264, 1998.

[156] E. J. Pauwels and G. Frederix. Finding salient regions for image segmentation and grouping. *Computer Vision and Image Understanding, Special Is-*

sue on Content-Based Access of Image and Video Libraries, 75(1/2):73–85, July/August 1999.

[157] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Series in Representation and Reasoning. Morgan Kaufmann, San Francisco, CA, 1988.

[158] R. Pekalska and R. P. W. Duin. Classifiers for dissimilarity-based pattern recognition. In *Proceedings of 15th IAPR International Conference on Pattern Recognition*, volume 2, pages 12–16, Barcelona, Spain, September 2000.

[159] J. Peng, B. Bhanu, and S. Qing. Probabilistic feature relevance learning for content-based image retrieval. *Computer Vision and Image Understanding, Special Issue on Content-Based Access of Image and Video Libraries*, 75(1/2):150–164, July/August 1999.

[160] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. In *SPIE Storage and Retrieval of Image and Video Databases II*, pages 34–47, February 1994.

[161] W. H. Press, B. P. Flannary, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recepies in C.* Cambridge University Press, 1990.

[162] B.A. Ribeiro-Neto and G.T. Assis. Reactive ranking for cooperative databases. In *Proceedings of XVII International Conference of the Chilean Computer Science Society*, pages 199–206, 1997.

[163] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[164] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[165] E. S. Ristad. A natural law of succession. Technical Report CS-TR-495-95, Department of Computer Science, Princeton University, July 1995.

[166] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, Inc., 1987.

[167] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[168] W. Rucklidge. Locating objects using the Housdorff distance. In *Proceedings of IEEE International Conference on Computer Vision*, pages 457–464, 1995.

[169] Y. Rui, , A. C. She, and T.S. Huang. Automated shape segmentation using attraction-based grouping in spatial-color-texture space. In *Proceedings of IEEE International Conference on Image Processing*, 1996.

[170] Y. Rui and T. Huang. Optimizing learning in image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 236–243, Hilton Head Island, South Carolina, June 2000.

[171] Y. Rui, T. S. Huang, and S-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, March 1999.

[172] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions*

on Circuits and Systems for Video Technology, Special Issue on Segmentation, Description, and Retrieval of Video Content, 8(5):644–655, September 1998.

[173] Y. Rui, T.S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proceedings of IEEE International Conference on Image Processing*, 1997.

[174] Y. Rui, T.S. Huang, S. Mehrotra, and M. Ortega. Automatic matching tool selection via relevance feedback in MARS. In *Proc. of the 2nd Int. Conf. on Visual Information Systems*, pages 109–116, San Diego, California, December 1997.

[175] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill, 1968.

[176] A. Saranli and M. Demirekler. A statistical unified framework for rank-based multiple classifer decision combination. *Pattern Recognition*, 34(4):865–884, April 2001.

[177] M. Schroder, H. Rehrauer, K. Siedel, and M. Datcu. Interactive learning and probabilistic retrieval in remote sensing image archives. *IEEE Transactions on Geoscience and Remote Sensing*, 38(5):2288–2298, September 2000.

[178] S. Sclaroff, L. Taycher, and M. L. Cascia. Imagerover: A content-based image browser for the world wide web. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1997.

[179] N. Sebe, M. Lew, and D. P. Huijsmans. Which ranking metric is optimal? with applications in image retrieval and stereo matching. In *Proceedings of 14th IAPR International Conference on Pattern Recognition*, volume 1, pages 265–271, Brisbane, Australia, August 16–20 1998.

[180] N. Sebe, M. Lew, and D. P. Huijsmans. Toward improved ranking metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1132–1143, October 2000.

[181] L. G. Shapiro and R. M. Haralick. Decomposition of two-dimensional shapes by graph-theoretic clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):10–20, January 1979.

[182] L. G. Shapiro and R. M. Haralick. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:90–94, 1985.

[183] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[184] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737, Puerto Rico, June 1997.

[185] J. Shi and J. Malik. Self inducing relational distance and its application to image segmentation. In *Proceedings of European Conference on Computer Vision*, June 1998.

[186] B. Simonnot and M. Smail. Model for interactive retrieval of videos and still images. In *Proceedings of International Workshop on Multimedia Database Management Systems*, pages 128–142, 1995.

[187] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[188] J. R. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression.* PhD thesis, Columbia University, 1997.

[189] J. R. Smith and C.-S. Li. Image classification and querying using composite region templates. *Computer Vision and Image Understanding, Special Issue on Content-Based Access of Image and Video Libraries*, 75(1/2):165–174, July/August 1999.

[190] M. D. Springer. *The Algebra of Random Variables.* John Wiley & Sons, Inc., 1979.

[191] M. Stricker and M. Orengo. Similarity of color images. In *SPIE Storage and Retrieval of Image and Video Databases*, February 1995.

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

[193] A. M. Tam and C. H. C. Leung. Structured high-level indexing of visual data content. In *Proceedings of the Third International Conference on Visual Information Systems*, pages 409–417, Amsterdam, The Netherlands, June 2–4 1999.

[194] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(6):460–473, June 1978.

[195] C. Taskiran, Jau-Yuen Chen, C.A. Bouman, and E.J. Delp. A compressed video database structured for active browsing and search. In *Proceedings of 1998 International Conference on Image Processing, ICIP'98*, volume 3, pages 133 –137, 1998.

[196] K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 228–235, Hilton Head Island, South Carolina, June 2000.

[197] M. Tuceryan and A. K. Jain. *Handbook of Pattern Recognition and Computer Vision*, chapter Texture Analysis, pages 235–276. World Scientific Publishing Company, River Edge, NJ, 1993.

[198] University of Washington, Department of Computer Science and Engineering, Groundtruth Database. Online: http://www.cs.washington.edu/research/imagedatabase/reportfin.html.

[199] A. Vailaya, M. Figueiredo, A. Jain, and H. J. Zhang. Content-based hierarchical classification of vacation images. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, June 1999.

[200] A. Vailaya, A. Jain, and H. J. Zhang. On image classification: City images vs. landscapes. *Pattern Recognition*, 31:1921–1936, December 1998.

[201] A. Vailaya, A. Jain, and H. J. Zhang. On image classification: City vs. landscape. In *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries, in conjunction with CVPR'98*, Santa Barbara, CA, June 1998.

[202] A. Vailaya, H. J. Zhang, and A. Jain. Automatic image orientation detection. In *Proceedings of IEEE International Conference on Image Processing*, Kobe, Japan, October 1999.

[203] A. Vailaya, Y. Zhong, and A. K. Jain. A hierarchical system for efficient image retrieval. In *Proceedings of the 13th IAPR International Conference on Pattern Recognition*, volume 3, pages 356–360, August 1996.

[204] N. Vasconcelos and A. Lippman. A Bayesian framework for semantic content characterization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 566–571, Santa Barbara, CA, June 1998.

[205] N. Vasconcelos and A. Lippman. Learning from user feedback in image retrieval systems. In *Proceedings of Neural Information Processing Systems*, Denver, CO, 1999.

[206] N. Vasconcelos and A. Lippman. A probabilistic architecture for content-based image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 216–221, Hilton Head Island, South Carolina, June 2000.

[207] Vision texture database. Online: http://vismod.www.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.

[208] K. Voigt. Skipper: A tool that lets browsers adapt to changes in document relevance to its user. In *Proceedings of Sixth International Workshop on Research Issues in Data Engineering*, pages 61–68, 1996.

[209] M. Welling and M. Weber. A constrained EM algorithm for Independent Component Analysis. *Neural Computation*, 13:677–689, 2001.

[210] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 34(4):1085–1094, July 1991.

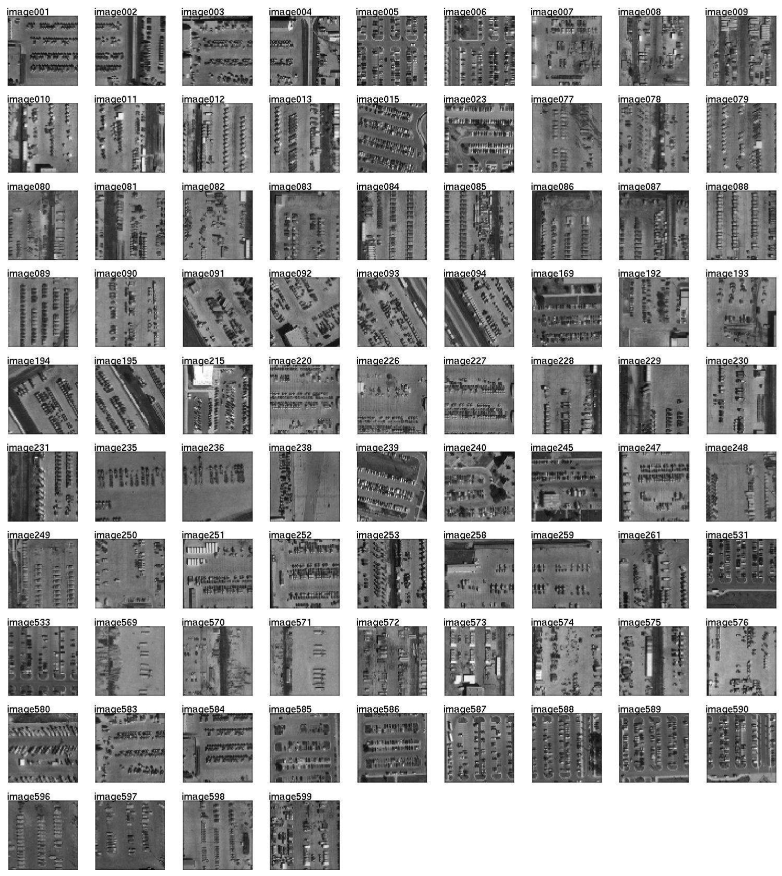
[211] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, May/June 1992.

[212] H. J. Zhang and D. Zhong. A scheme for visual feature-based image retrieval. In *SPIE Storage and Retrieval of Image and Video Databases*, volume 2420, pages 36–46, 1995.

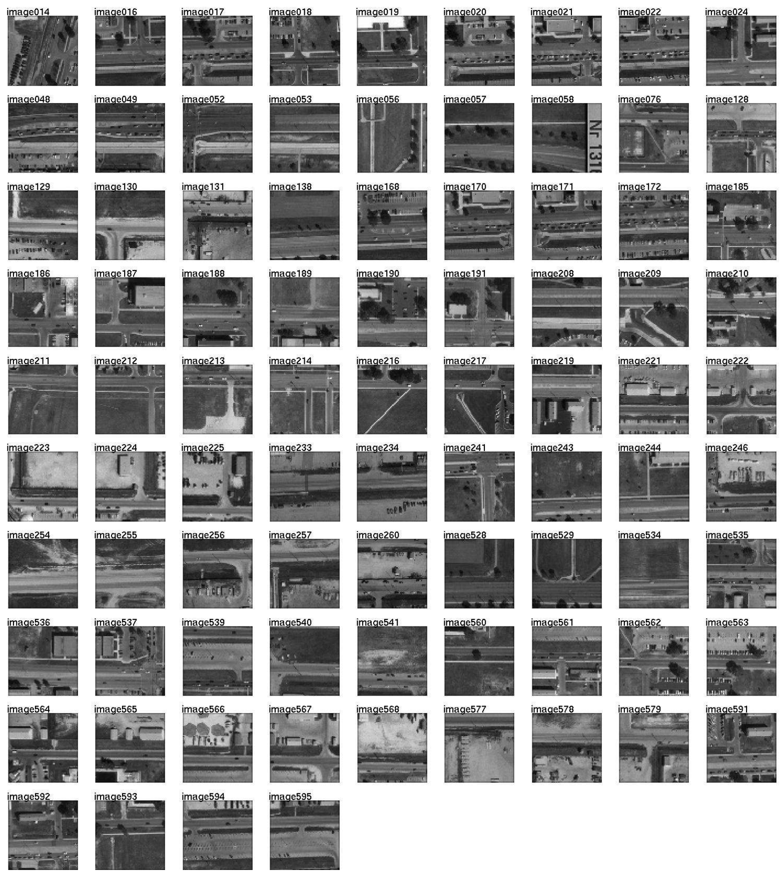
[213] X. S. Zhou, Y. Rui, and T.S. Huang. Water-filling: A novel way for image structural feature extraction. In *Proceedings of IEEE International Conference on Image Processing*, 1999.

Appendix A

ISL DATABASE GROUNDTRUTH

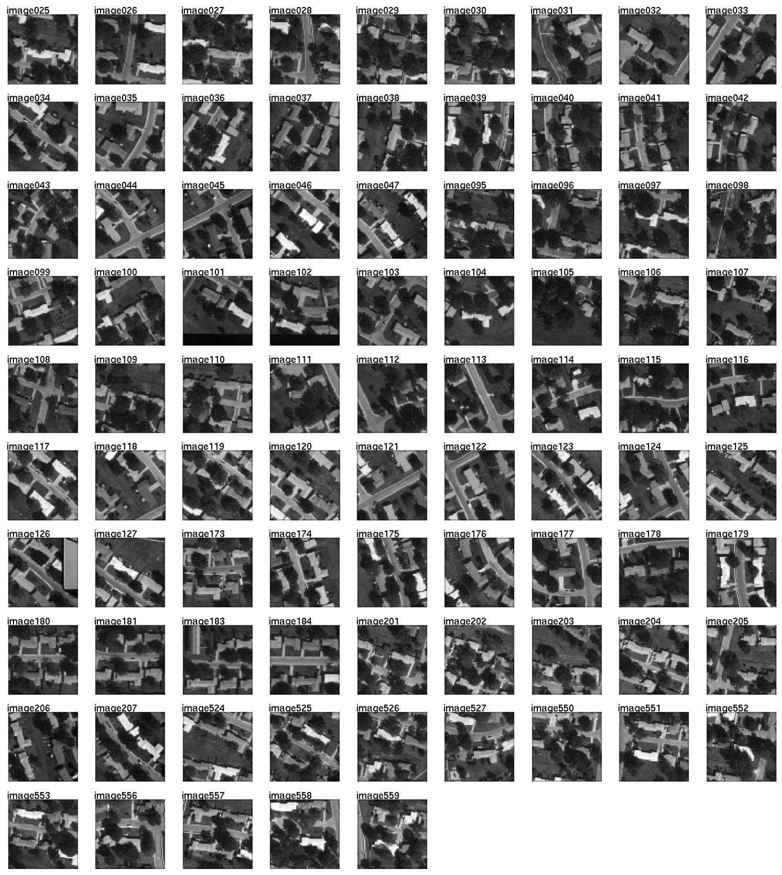


(i) Parking lots

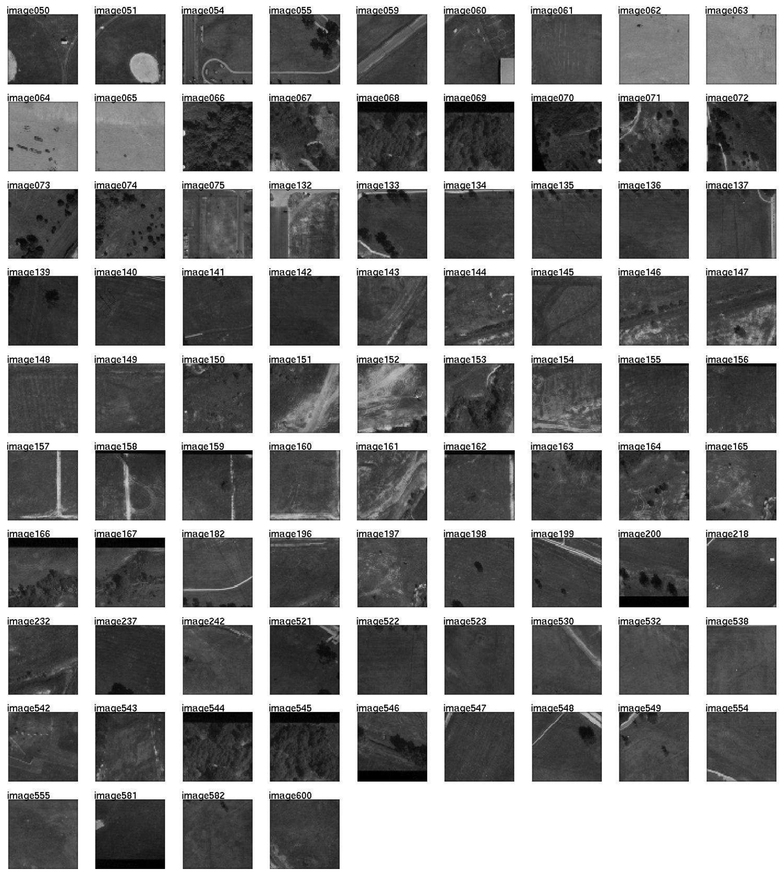


(ii) Roads

Figure A.1: ISL Database groundtruth.



(i) Residential areas



(ii) Landscapes

Figure A.1: ISL Database groundtruth (continued).

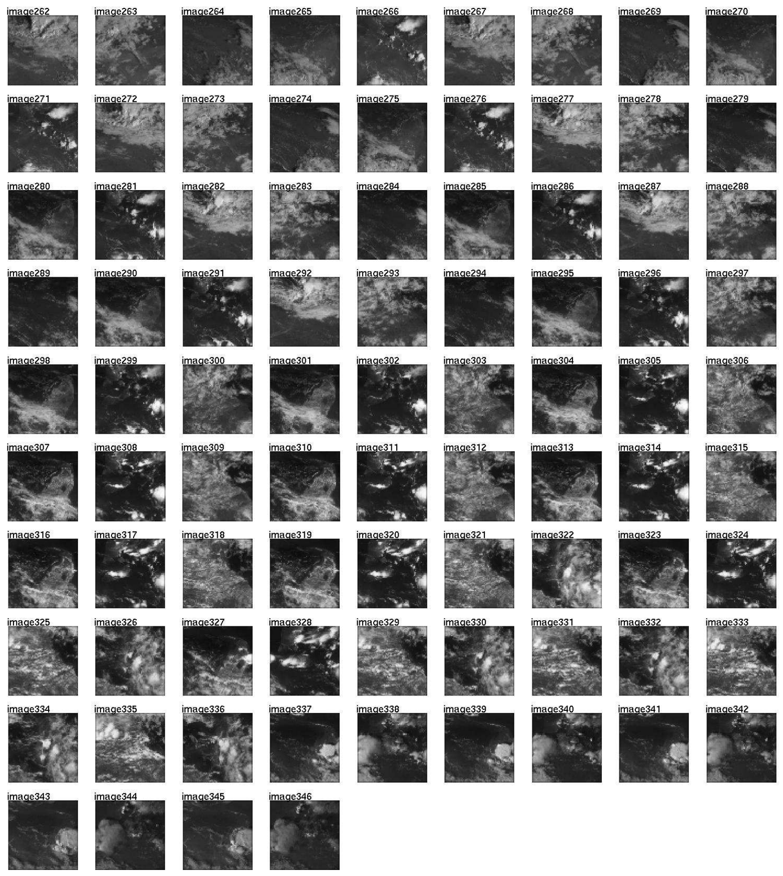


image262 image263 image264 image265 image266 image267 image268 image269 image270
image271 image272 image273 image274 image275 image276 image277 image278 image279
image280 image281 image282 image283 image284 image285 image286 image287 image288
image289 image290 image291 image292 image293 image294 image295 image296 image297
image298 image299 image300 image301 image302 image303 image304 image305 image306
image307 image308 image309 image310 image311 image312 image313 image314 image315
image316 image317 image318 image319 image320 image321 image322 image323 image324
image325 image326 image327 image328 image329 image330 image331 image332 image333
image334 image335 image336 image337 image338 image339 image340 image341 image342
image343 image344 image345 image346

(iii) LANDSAT USA

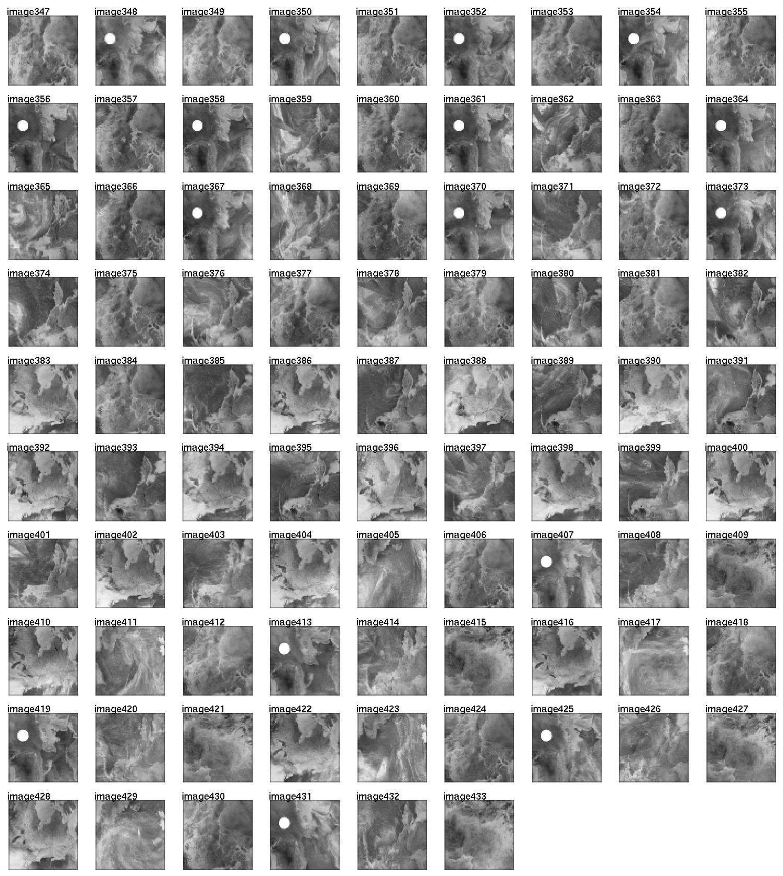
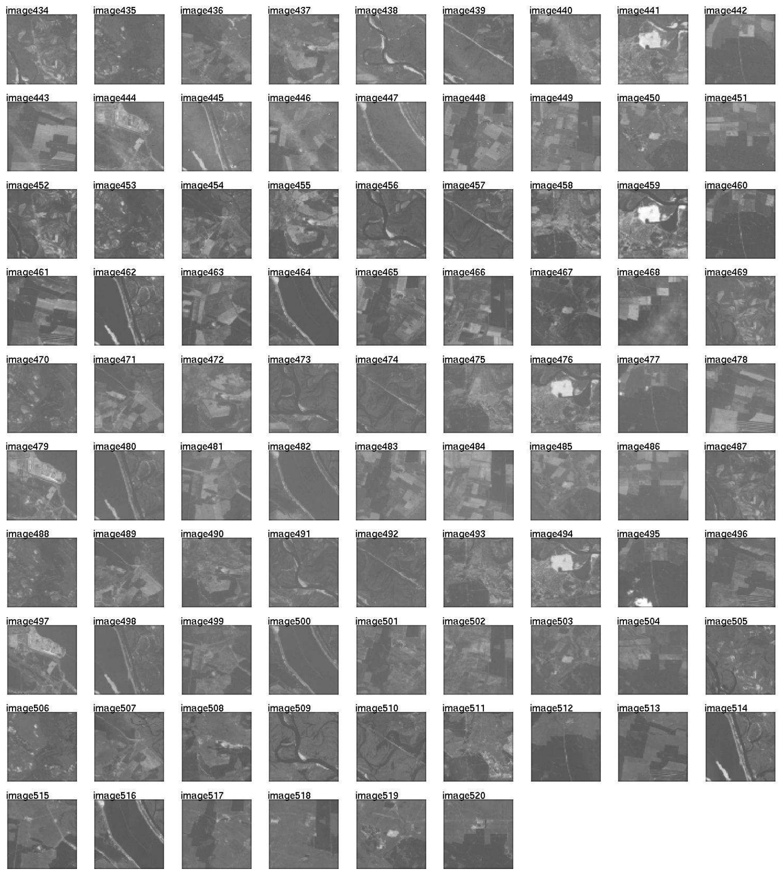


image347 image348 image349 image350 image351 image352 image353 image354 image355
image356 image357 image358 image359 image360 image361 image362 image363 image364
image365 image366 image367 image368 image369 image370 image371 image372 image373
image374 image375 image376 image377 image378 image379 image380 image381 image382
image383 image384 image385 image386 image387 image388 image389 image390 image391
image392 image393 image394 image395 image396 image397 image398 image399 image400
image401 image402 image403 image404 image405 image406 image407 image408 image409
image410 image411 image412 image413 image414 image415 image416 image417 image418
image419 image420 image421 image422 image423 image424 image425 image426 image427
image428 image429 image430 image431 image432 image433

(iv) DMSP North Pole

Figure A.1: ISL Database groundtruth (continued).

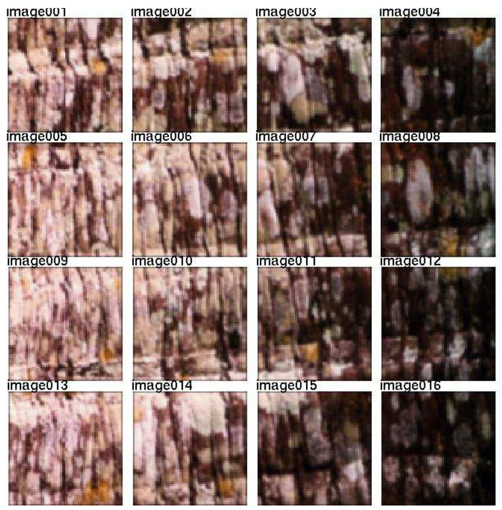


(v) LANDSAT Chernobyl

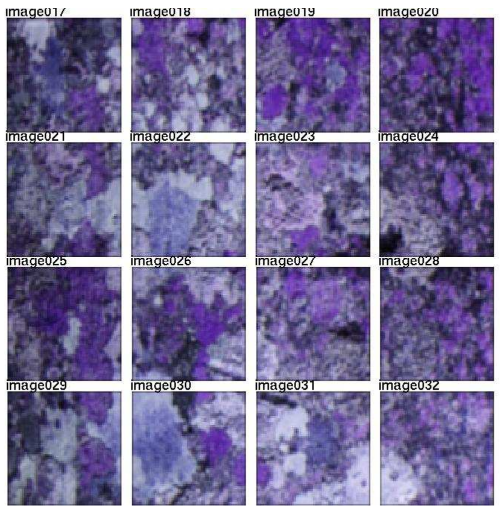
Figure A.1: ISL Database groundtruth (continued).

Appendix B

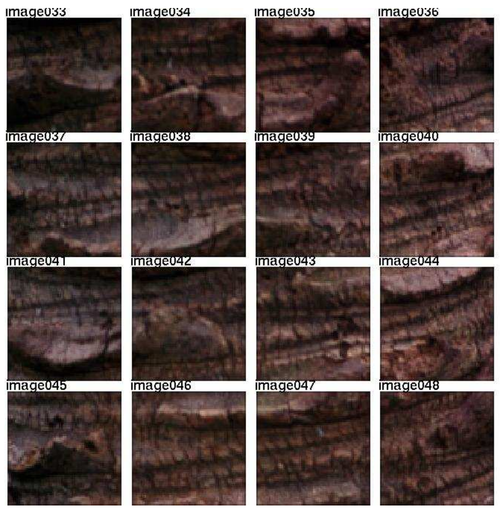
VISTEX DATABASE GROUNDTRUTH



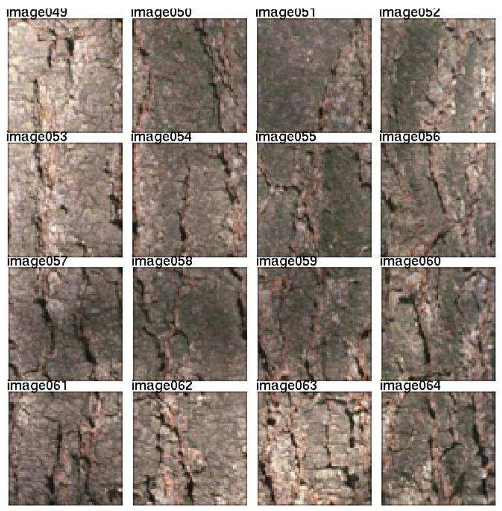
(vi) Bark.0000



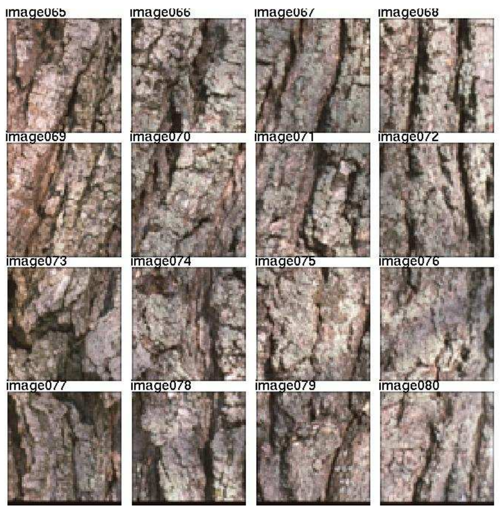
(vii) Bark.0001



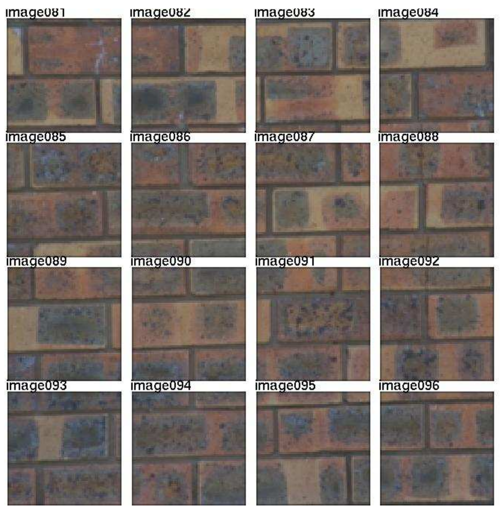
(viii) Bark.0006



(ix) Bark.0008

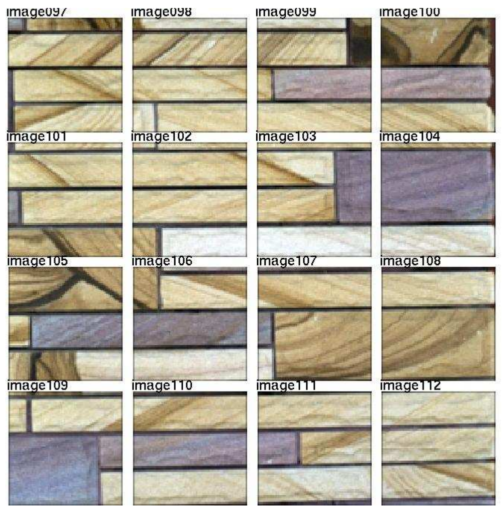


(x) Bark.0012

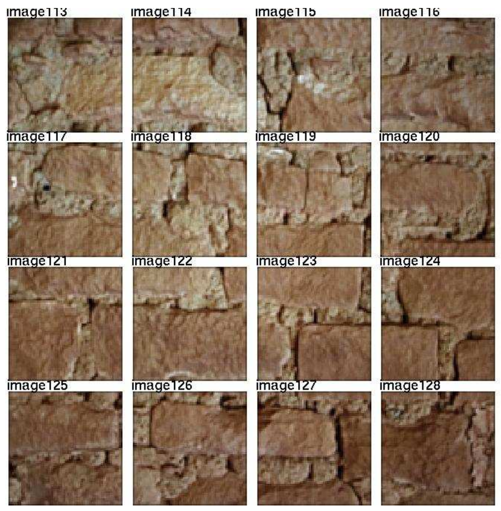


(xi) Brick.0000

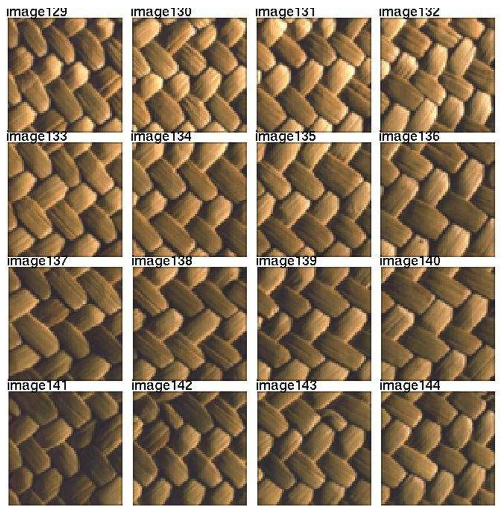
Figure B.1: VisTex Database groundtruth.



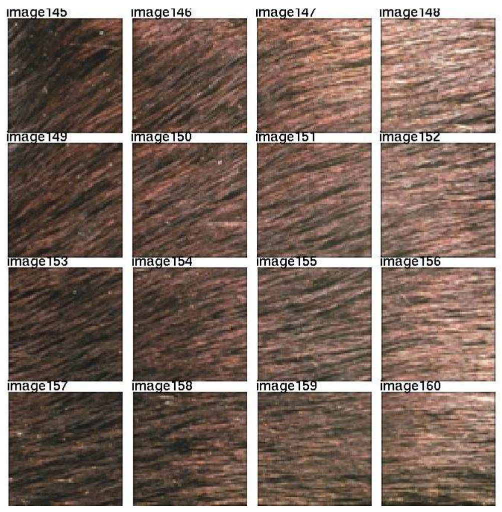
(i) Brick.0002



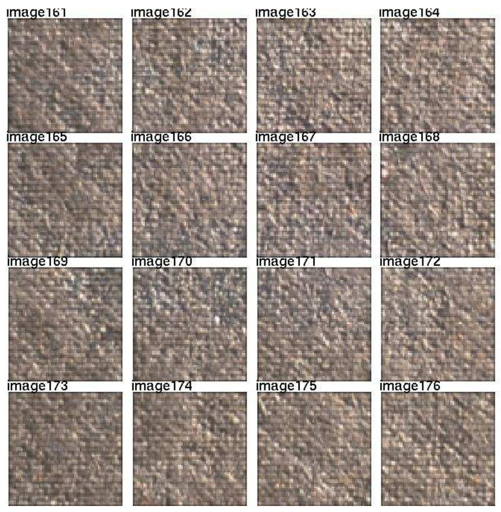
(ii) Brick.0005



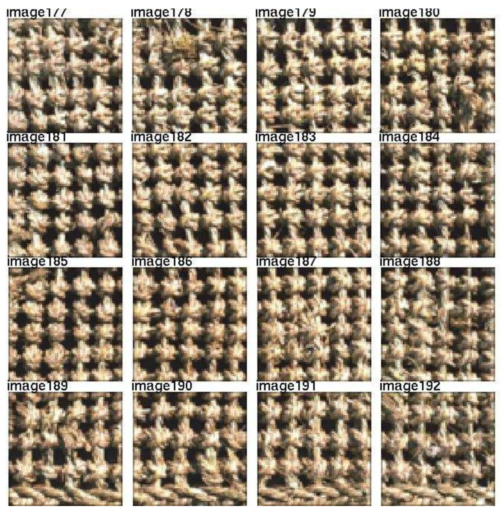
(iii) Fabric.0002



(iv) Fabric.0005



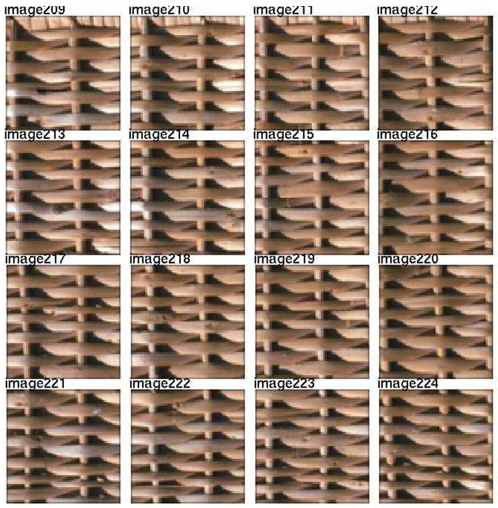
(v) Fabric.0007



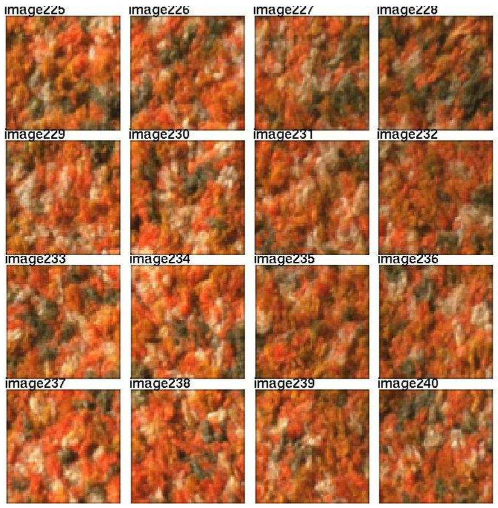
(vi) Fabric.0009

Figure B.1: VisTex Database groundtruth (continued).

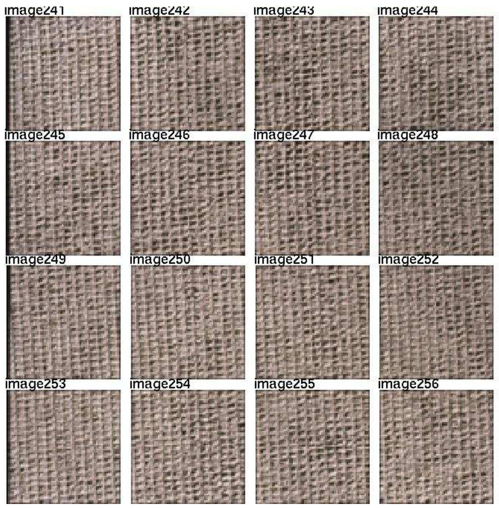
(vii) Fabric.0011



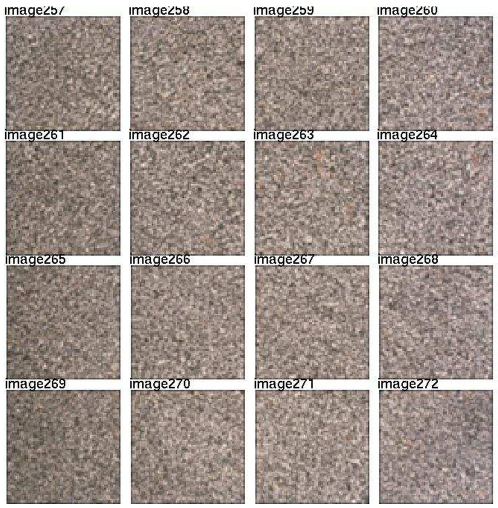
(viii) Fabric.0013



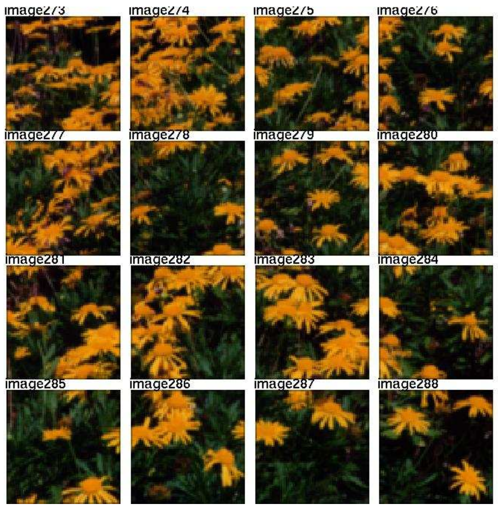
(ix) Fabric.0015



(x) Fabric.0017

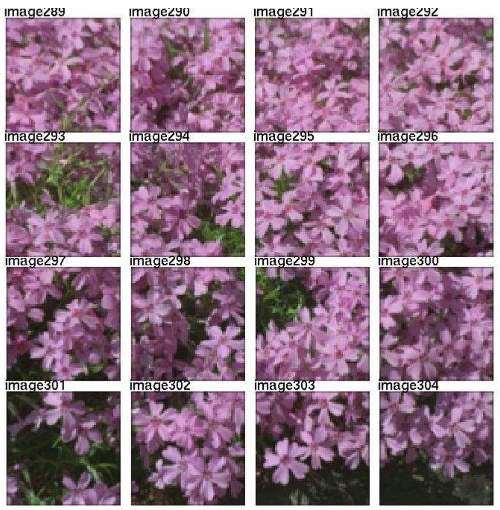


(xi) Fabric.0019

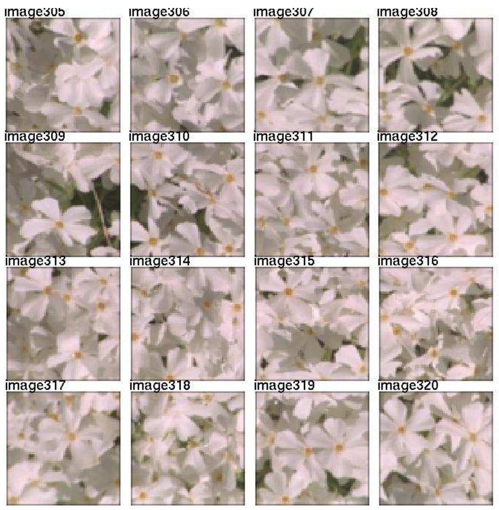


(xii) Flowers.0000

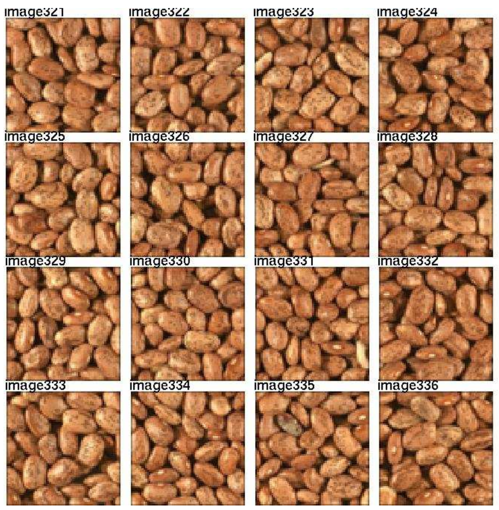
Figure B.1: VisTex Database groundtruth (continued).



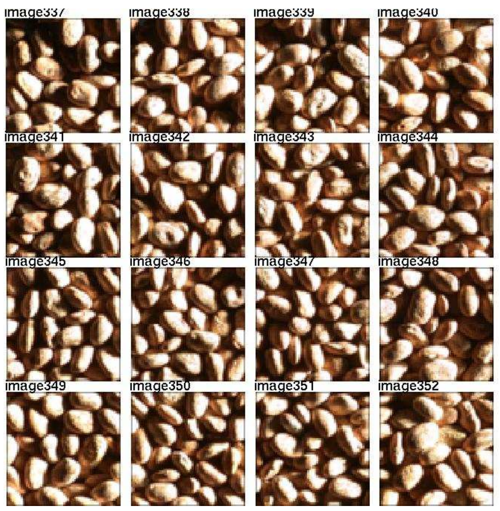
(xiii) Flowers.0002



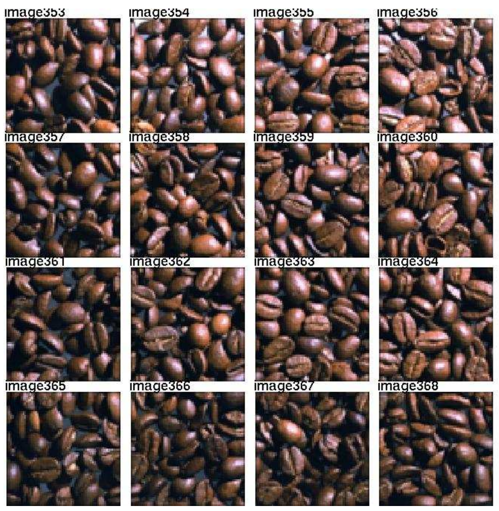
(xiv) Flowers.0007



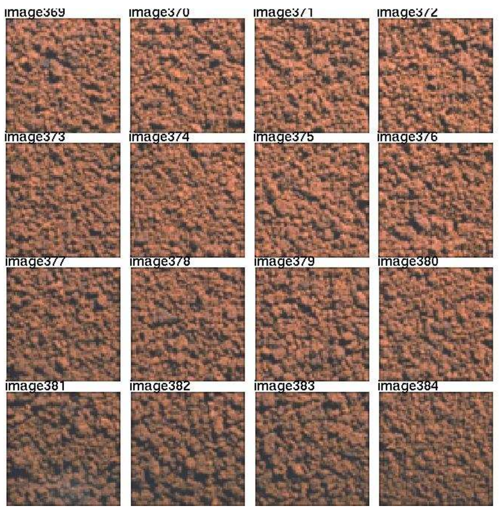
(xv) Food.0000



(xvi) Food.0001



(xvii) Food.0004

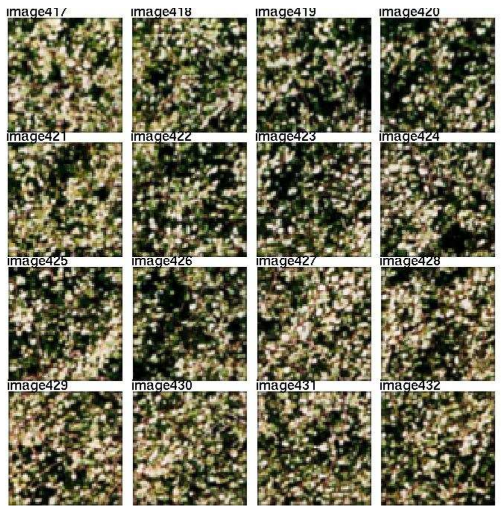


(xviii) Food.0005

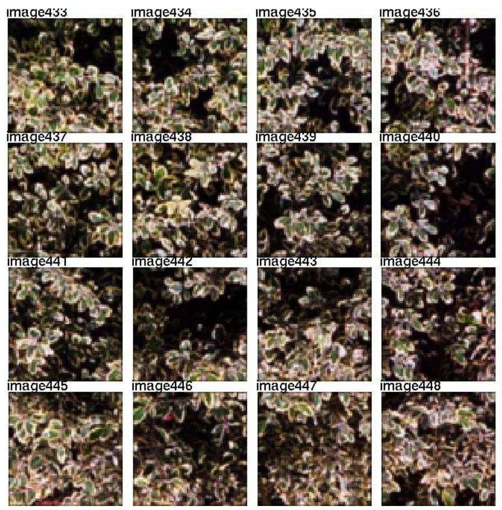
Figure B.1: VisTex Database groundtruth (continued).

(xix) Food.0006

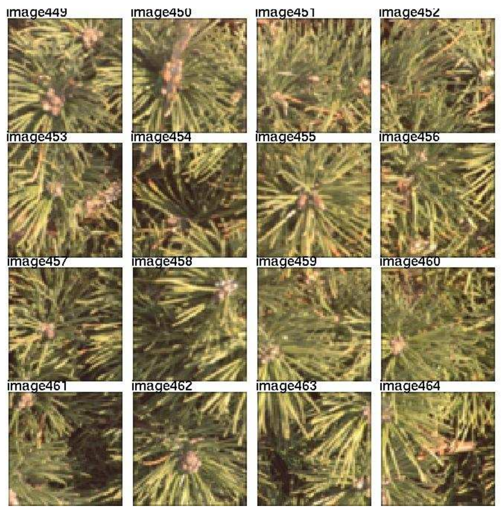
(xx) Grass.0002



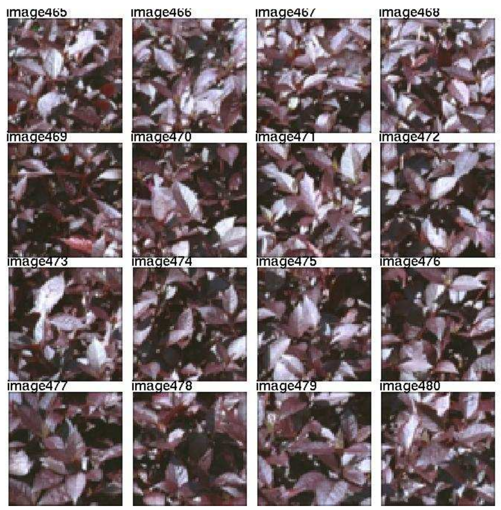
(xxi) Leaves.0003



(xxii) Leaves.0008



(xxiii) Leaves.0010

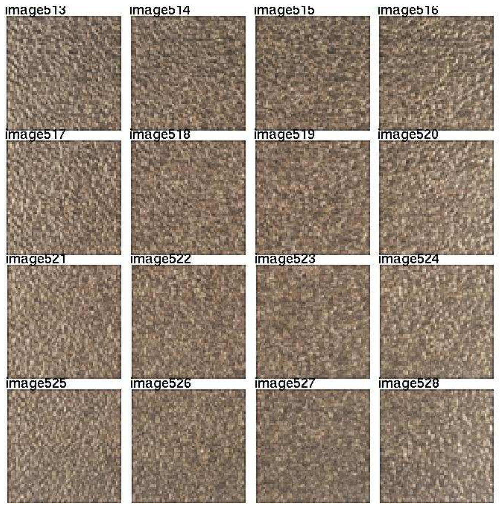


(xxiv) Leaves.0011

Figure B.1: VisTex Database groundtruth (continued).

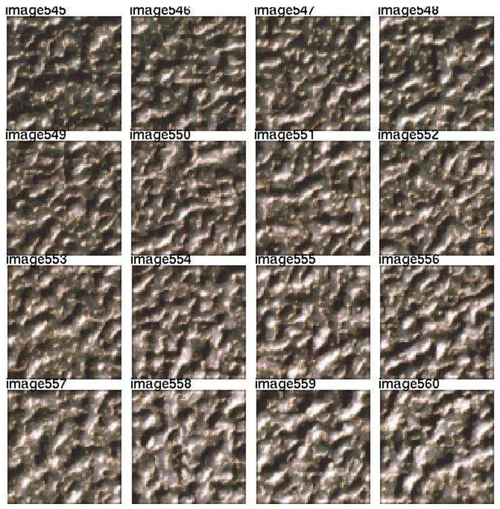
(xxv) Leaves.0014

(xxvi) Leaves.0016

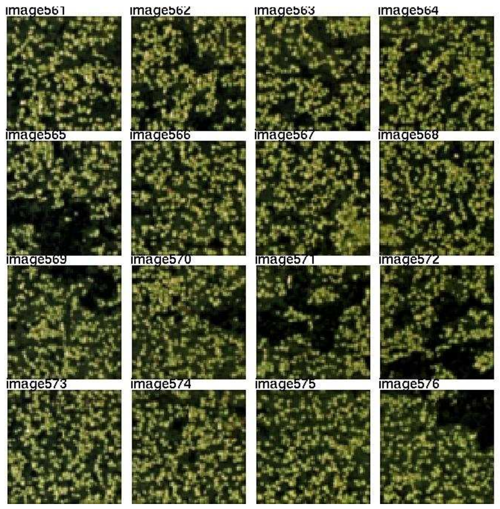


(xxvii) Metal.0000

(xxviii) Metal.0002

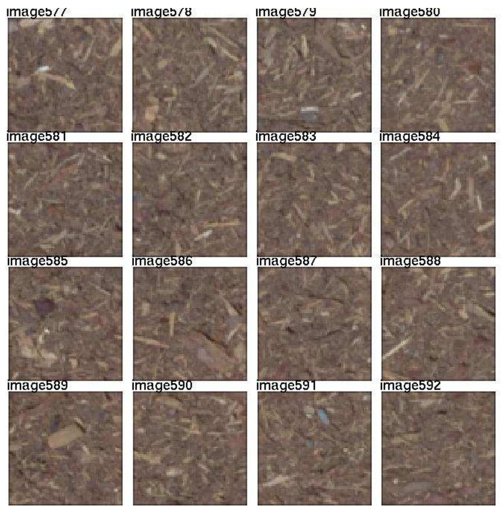


(xxix) Metal.0004

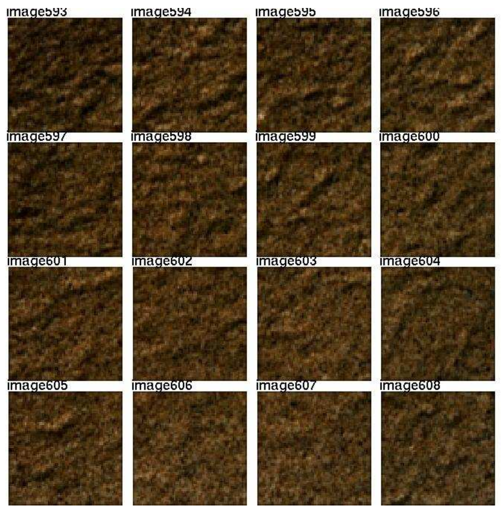


(xxx) Misc.0000

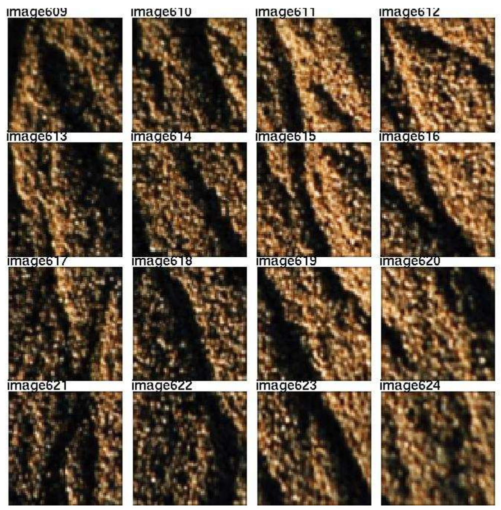
Figure B.1: VisTex Database groundtruth (continued).



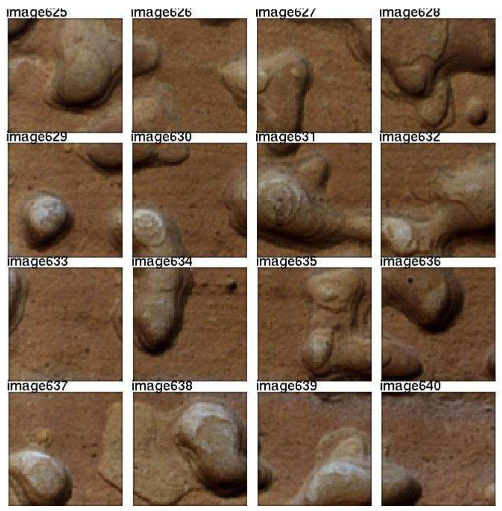
(xxxi) Misc.0002



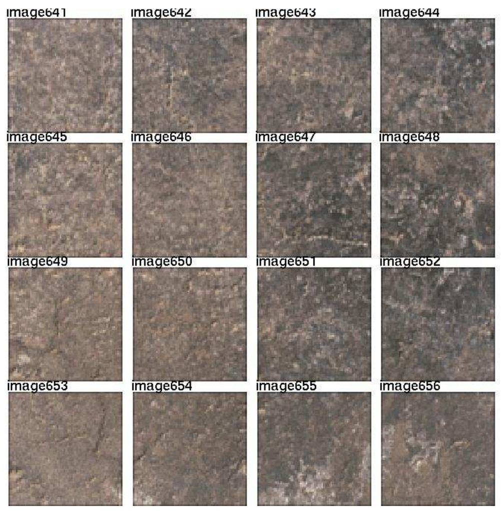
(xxxii) Sand.0000



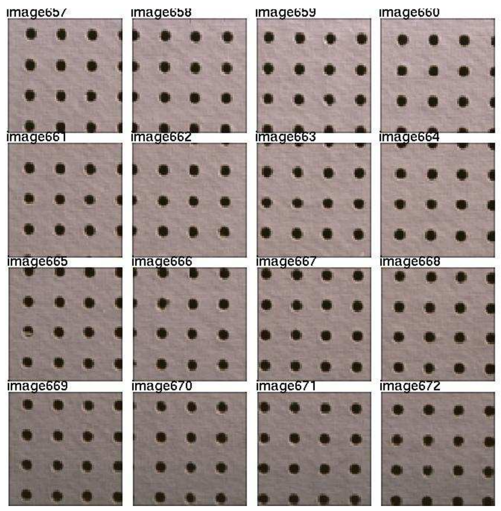
(xxxiii) Sand.0005



(xxxiv) Stone.0002



(xxxv) Stone.0005

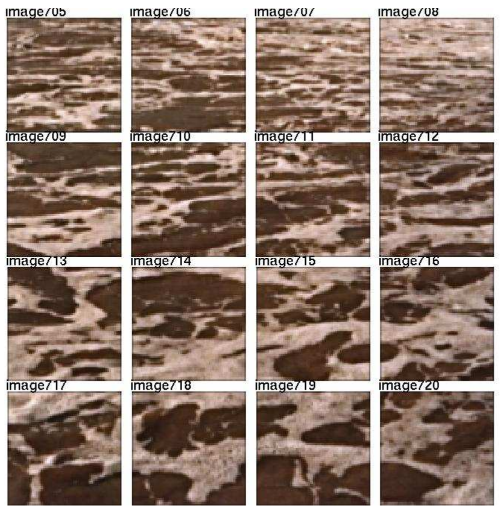


(xxxvi) Tile.0007

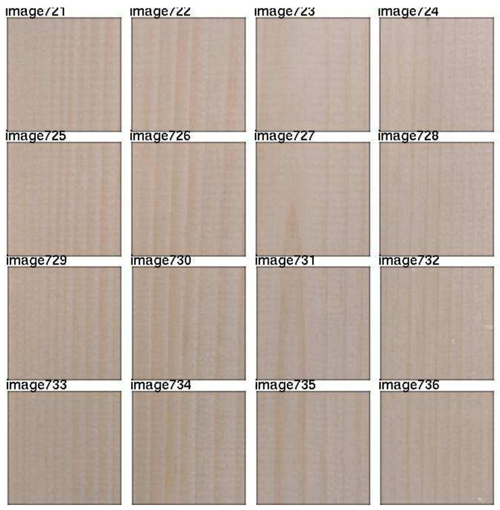
Figure B.1: VisTex Database groundtruth (continued).

(xxxvii) Water.0002

(xxxviii) Water.0003



(xxxix) Water.0004



(xl) Wood.0002

Figure B.1: VisTex Database groundtruth (continued).

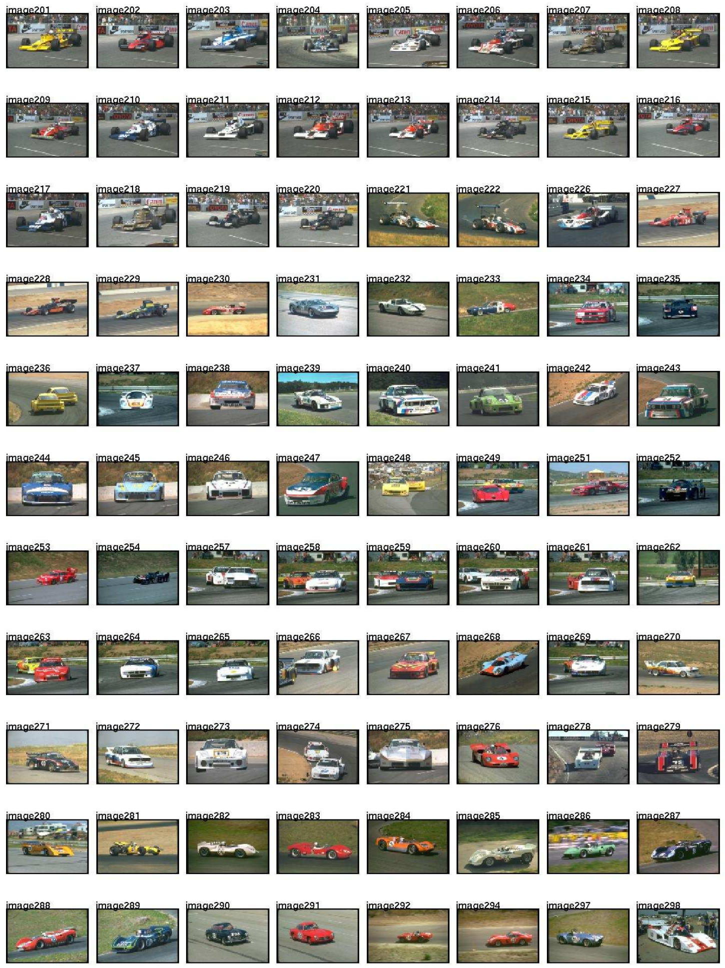
Appendix C

COREL DATABASE GROUNDTRUTH

(xli) Air shows

(xlii) Arabian horses

Figure C.1: COREL Database groundtruth.



(i) Auto racing

(ii) Bald eagles

Figure C.1: COREL Database groundtruth (continued).

image401 image402 image403 image404 image405 image406 image407 image408
image409 image410 image411 image412 image413 image414 image415 image416
image417 image418 image419 image420 image421 image422 image423 image424
image425 image426 image427 image428 image429 image430 image431 image432
image433 image434 image435 image436 image437 image438 image439 image440
image441 image442 image443 image444 image445 image446 image447 image448
image449 image451 image452 image453 image454 image455 image456 image457
image458 image459 image460 image465 image466 image467 image469 image471
image472 image473 image474 image475 image476 image477 image479 image480
image481 image482 image483 image484 image485 image486 image488 image489
image490 image491 image492 image493 image494 image495 image496 image497
image498 image499 image500

(iii) Cheetahs

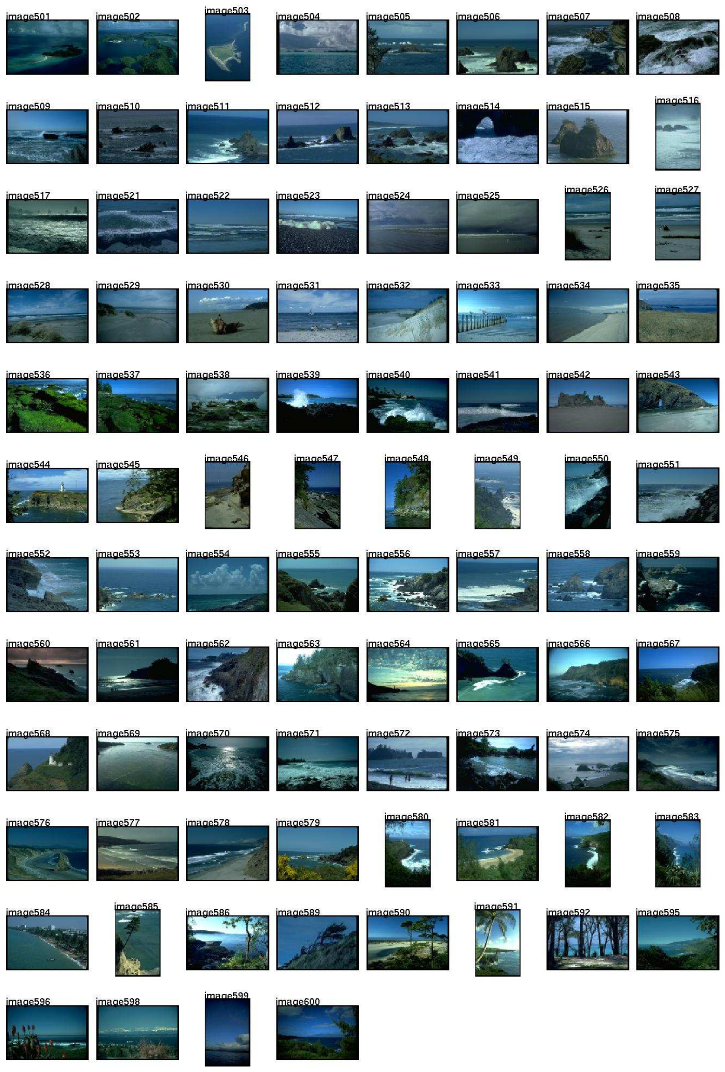
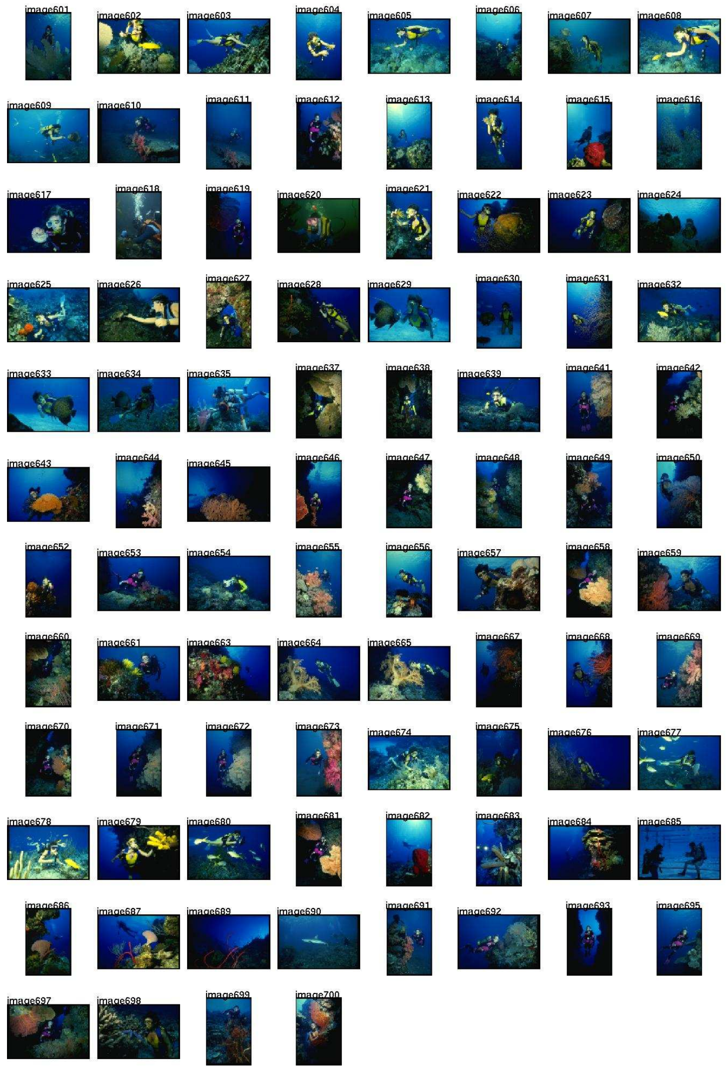


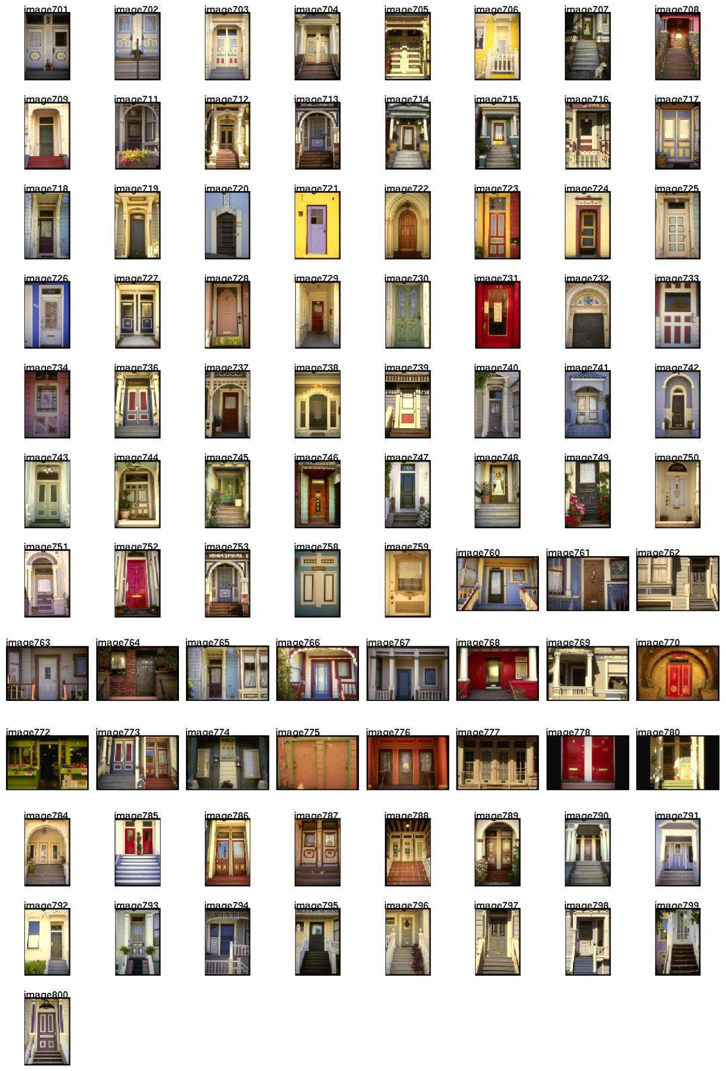
image501 image502 image503 image504 image505 image506 image507 image508
image509 image510 image511 image512 image513 image514 image515 image516
image517 image521 image522 image523 image524 image525 image526 image527
image528 image529 image530 image531 image532 image533 image534 image535
image536 image537 image538 image539 image540 image541 image542 image543
image544 image545 image546 image547 image548 image549 image550 image551
image552 image553 image554 image555 image556 image557 image558 image559
image560 image561 image562 image563 image564 image565 image566 image567
image568 image569 image570 image571 image572 image573 image574 image575
image576 image577 image578 image579 image580 image581 image582 image583
image584 image585 image586 image589 image590 image591 image592 image595
image596 image598 image599 image600

(iv) Coasts

Figure C.1: COREL Database groundtruth (continued).

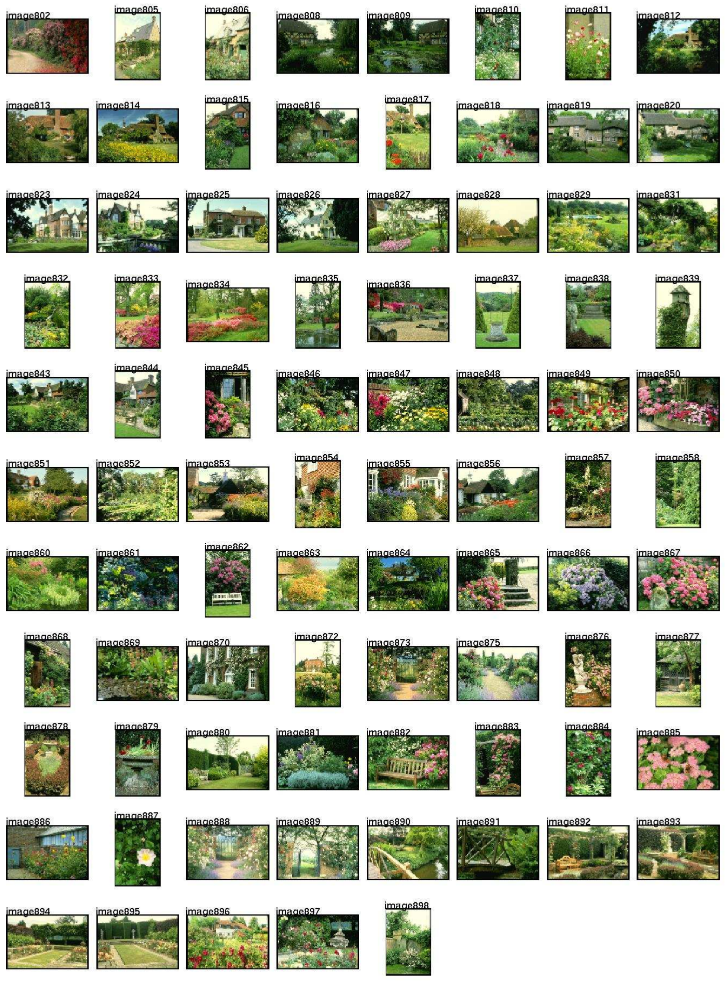


(v) Divers and diving

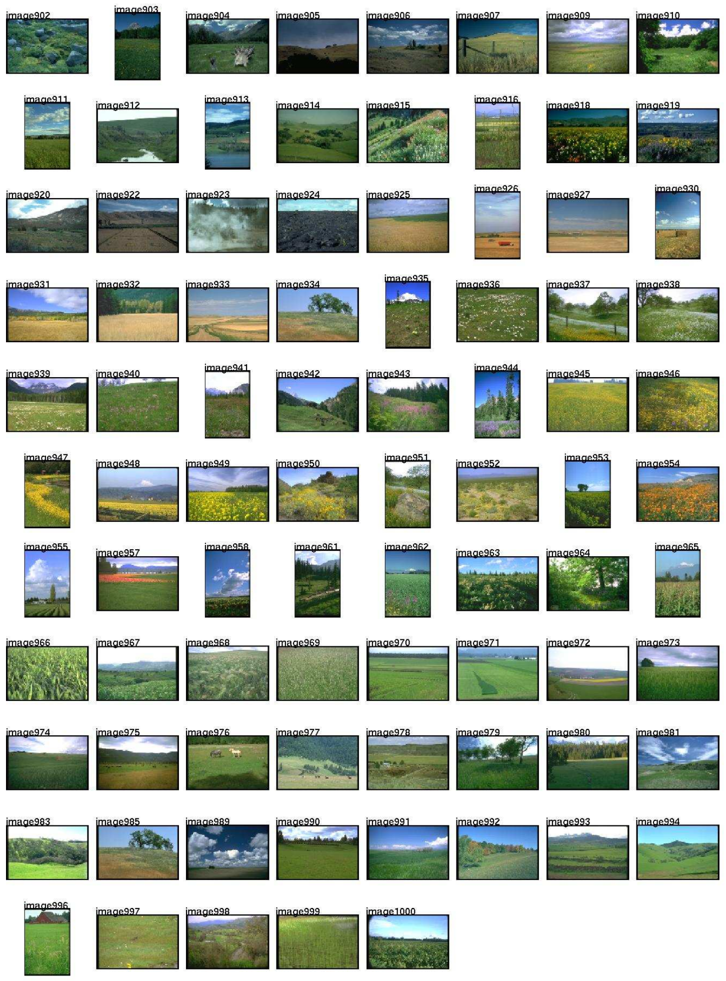


(vi) Doors of San Francisco

Figure C.1: COREL Database groundtruth (continued).



(vii) English country gardens



(viii) Fields

Figure C.1: COREL Database groundtruth (continued).

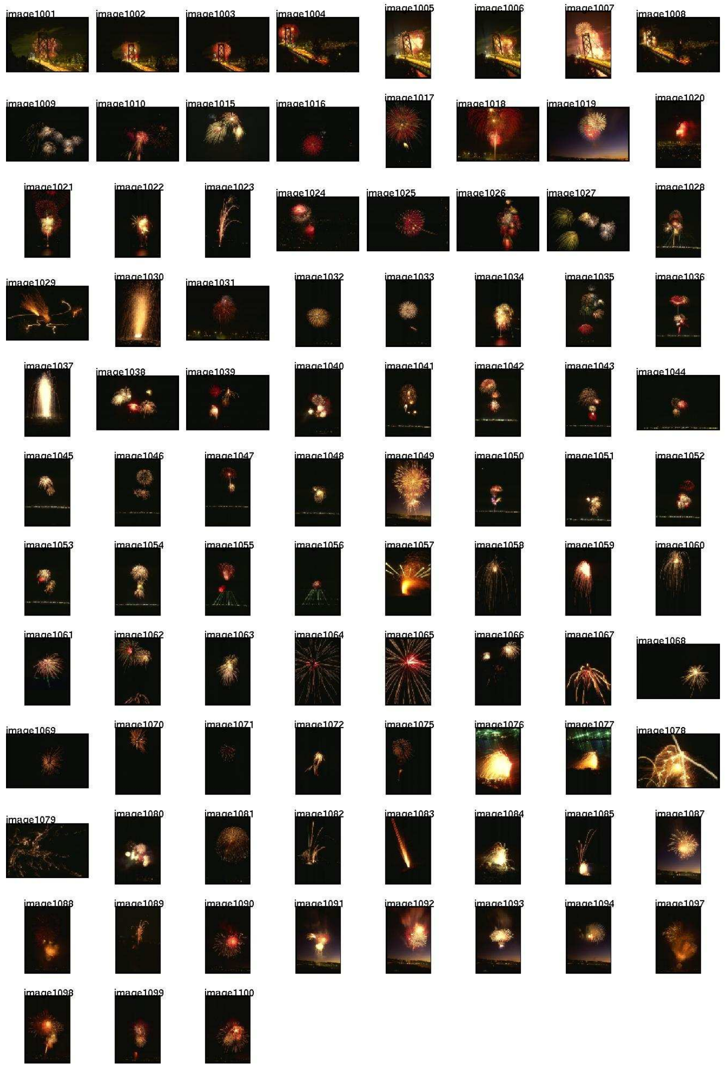


image1001 image1002 image1003 image1004 image1005 image1006 image1007 image1008
image1009 image1010 image1015 image1016 image1017 image1018 image1019 image1020
image1021 image1022 image1023 image1024 image1025 image1026 image1027 image1028
image1029 image1030 image1031 image1032 image1033 image1034 image1035 image1036
image1037 image1038 image1039 image1040 image1041 image1042 image1043 image1044
image1045 image1046 image1047 image1048 image1049 image1050 image1051 image1052
image1053 image1054 image1055 image1056 image1057 image1058 image1059 image1060
image1061 image1062 image1063 image1064 image1065 image1066 image1067 image1068
image1069 image1070 image1071 image1072 image1075 image1076 image1077 image1078
image1079 image1080 image1081 image1082 image1083 image1084 image1085 image1087
image1088 image1089 image1090 image1091 image1092 image1093 image1094 image1097
image1098 image1099 image1100

(ix) Fireworks

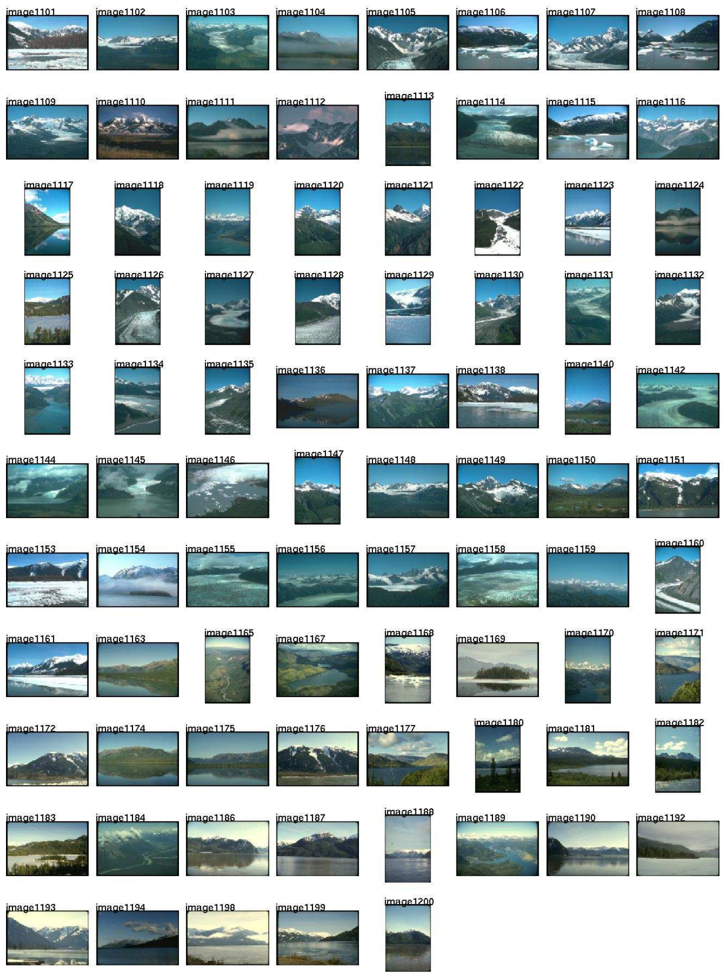
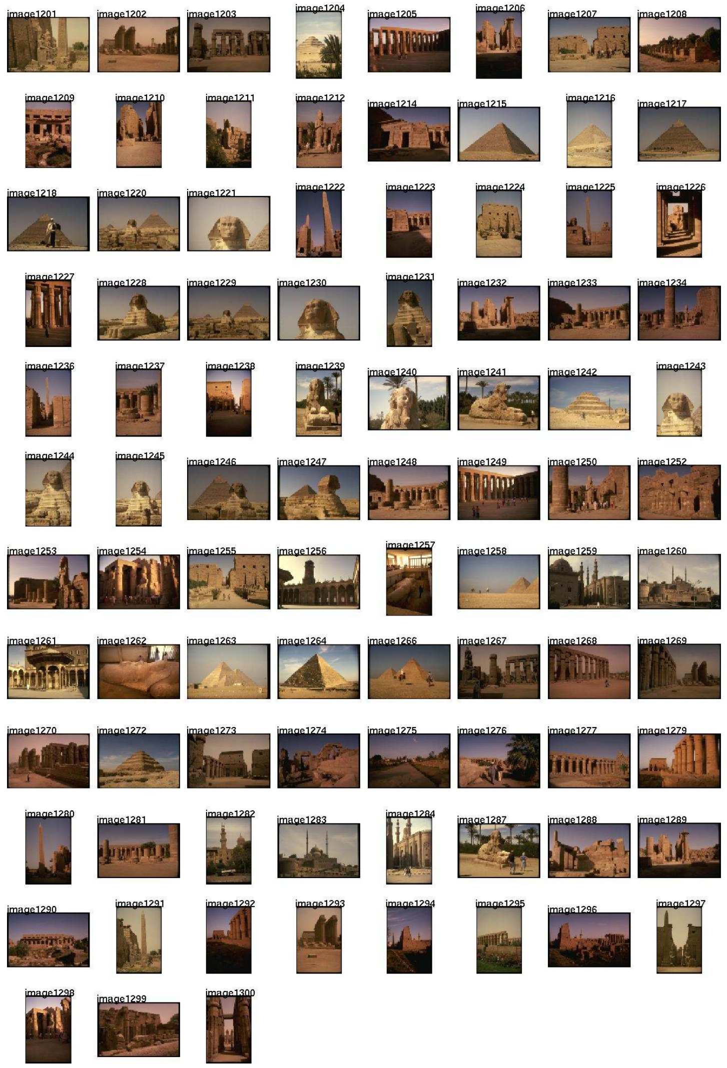


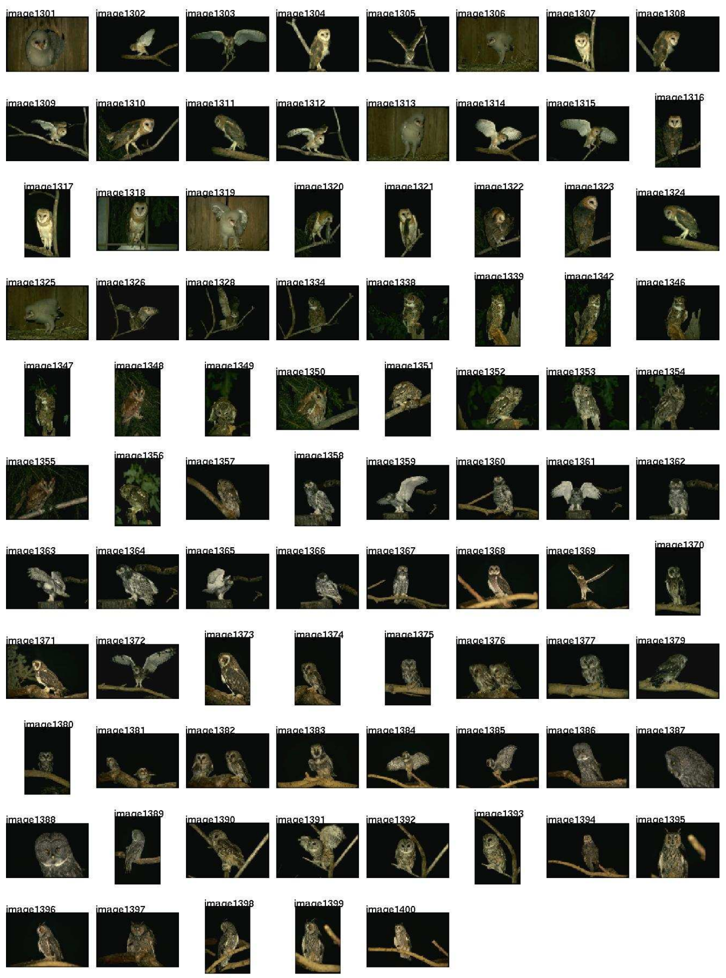
image1101 image1102 image1103 image1104 image1105 image1106 image1107 image1108
image1109 image1110 image1111 image1112 image1113 image1114 image1115 image1116
image1117 image1118 image1119 image1120 image1121 image1122 image1123 image1124
image1125 image1126 image1127 image1128 image1129 image1130 image1131 image1132
image1133 image1134 image1135 image1136 image1137 image1138 image1140 image1142
image1144 image1145 image1146 image1147 image1148 image1149 image1150 image1151
image1153 image1154 image1155 image1156 image1157 image1158 image1159 image1160
image1161 image1163 image1165 image1167 image1168 image1169 image1170 image1171
image1172 image1174 image1175 image1176 image1177 image1180 image1181 image1182
image1183 image1184 image1186 image1187 image1188 image1189 image1190 image1192
image1193 image1194 image1198 image1199 image1200

(x) Glaciers and mountains

Figure C.1: COREL Database groundtruth (continued).

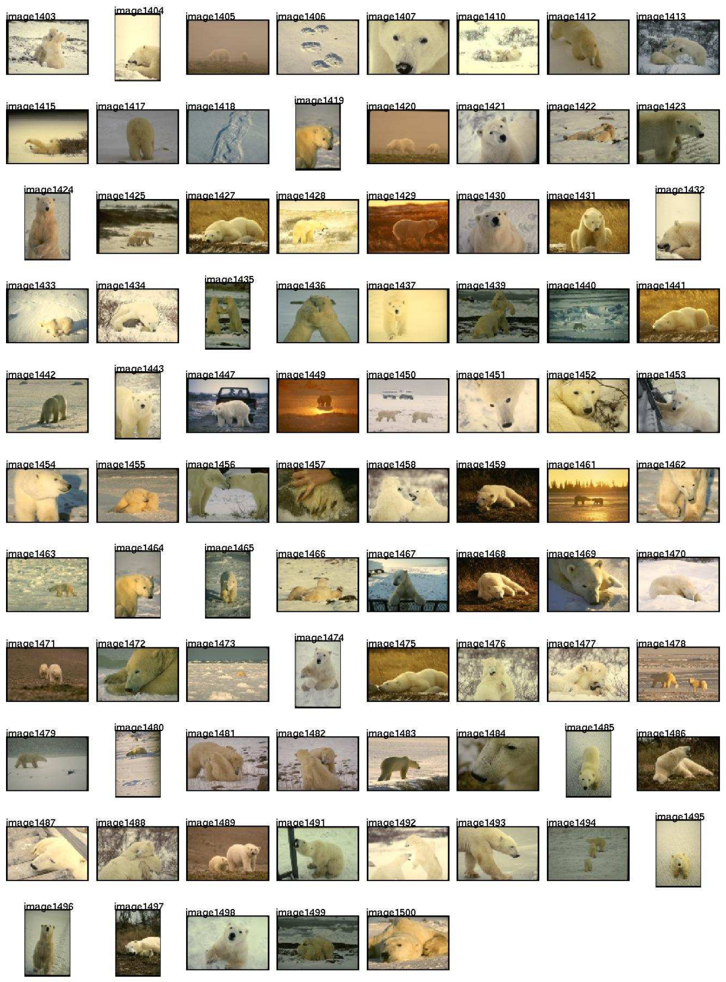


(xi) Land of the pyramids

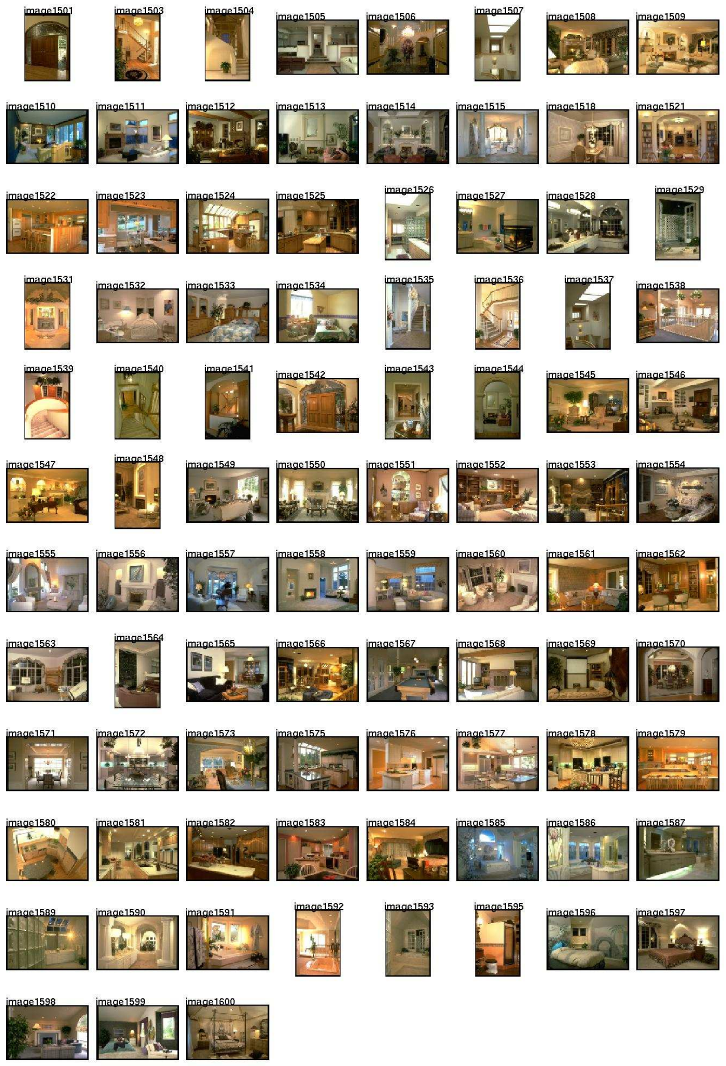


(xii) Owls

Figure C.1: COREL Database groundtruth (continued).

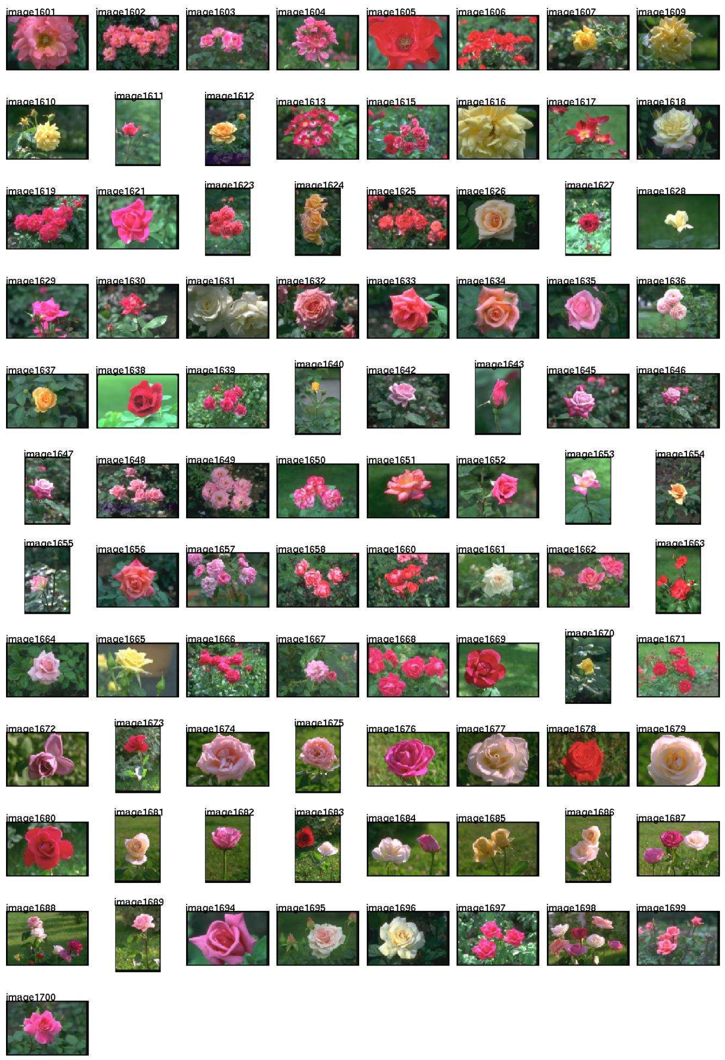


(xiii) Polar bears

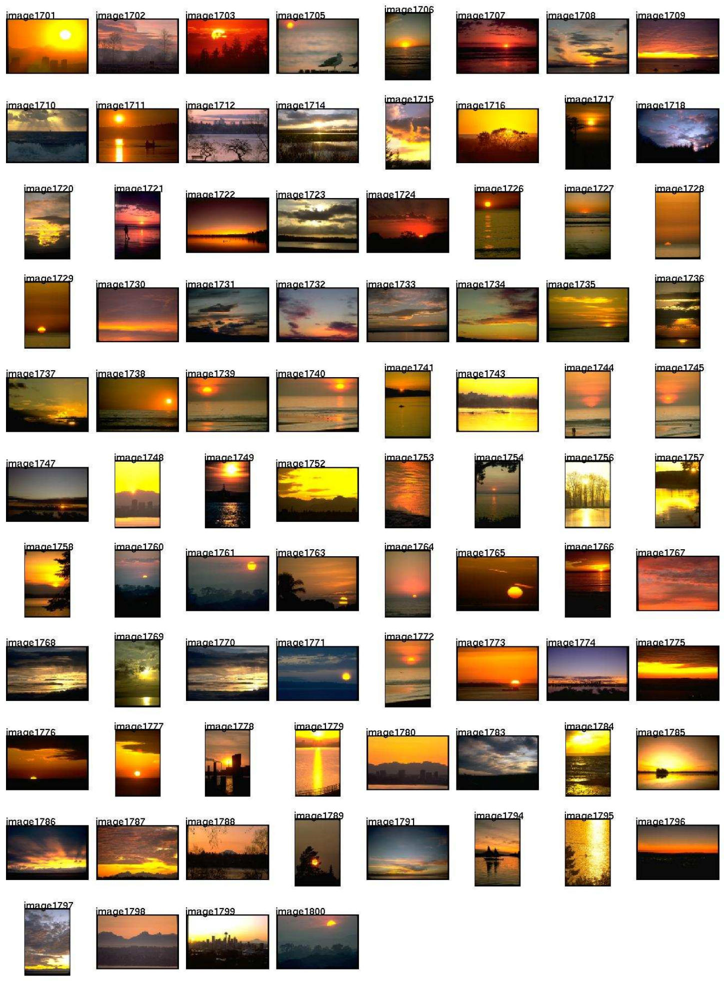


(xiv) Residential interiors

Figure C.1: COREL Database groundtruth (continued).



(xv) Roses



(xvi) Sunsets and sunrises

Figure C.1: COREL Database groundtruth (continued).

VITA

Selim Aksoy was born in Ankara, Turkey, in 1976. He received his B.S. degree from the Department of Electrical and Electronics Engineering at Middle East Technical University in Ankara, and his M.S. degree from the Department of Electrical Engineering at the University of Washington, Seattle in 1996 and 1998, respectively. He spent the summers of 1998 and 1999 as a visiting researcher at Tampere International Center for Signal Processing at Tampere University of Technology, Tampere, Finland. He was a research intern at MathSoft Inc. in Seattle during the summer of 2000. His research interests include computer vision, pattern recognition and image processing. He is particularly interested in probabilistic modeling of image content, multimedia information analysis and statistical pattern recognition.