This is to certify that the

dissertation entitled

SEMANTIC CLASSIFICATION IN IMAGE DATABASES

presented by

ADITYA VAILAYA

has been accepted towards fulfillment
of the requirements for

PhD degree in Computer Science & Engineering

Major professor

Date May 10, 2000

Semantic Classification in Image Databases

By

Aditya Vailaya

Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science & Engineering

2000

Abstract

Semantic Classification in Image Databases

By

Aditya Vailaya

Due to the huge amount of potentially interesting documents available over the Internet, searching for relevant information has become very difficult. Since image and video are a major source of these data, grouping images into (semantically) meaningful categories using low-level visual features is an important (and challenging) problem in content-based image retrieval. Using Bayesian classifiers, we attempt to capture high-level concepts from low-level image features. Specifically, we have developed Bayesian classifiers for semantic image classification (indoor vs. outdoor, city vs. landscape, and sunset vs. forest vs. mountain), image orientation detection, and object detection (detecting regions of sky and vegetation in outdoor images). We demonstrate that a small codebook (the optimal codebook size is selected using a modified MDL criterion) extracted from a learning vector quantizer can be used to estimate the class-conditional densities of the observed features needed for image classification. We have developed an incremental learning paradigm, a feature selection scheme, a rejection scheme, and a classifier combination strategy using bagging

to improve classifier performance. Empirical results on a large database ($\sim 24,000$ images) show that semantic categorization and organization of the database using the proposed classification schemes improves both retrieval accuracy and efficiency.

To My Family

ACKNOWLEDGMENTS

I would like to acknowledge all the people who have assisted me throughout my studies at Michigan State University. I am extremely grateful to my advisor, Professor Anil Jain, for his continuous guidance and encouragement, and the valuable time that he spent with me. I owe my great interest in research to him.

I am extremely grateful to Dr. HongJiang Zhang and the technical staff at Hewlett Packard Labs, Palo Alto for their guidance and support during my stay there in summers of 1997 and 1998, for the fellowship granted to me for the period 1997-1999, and for providing me with the image database for my research.

I would like to dedicate this thesis to my family who have constantly encouraged me to achieve new heights. Without their love, patience, understanding, and support, I would not have finished this thesis.

I would like to acknowledge Mário Figueiredo from Instituto de Telecomunicações for his help and guidance in developing the Bayesian framework, especially, the application of the MDL principle. I would also like to acknowledge other members of my thesis committee (Prof. Stockman, Prof. Weng, and Prof. Koul) for their guidance and suggestions through the course of my research. Finally, I would like to thank the

members of the PRIP lab and from my friends at Michigan State University for their help and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information is inherently multimodal. Humans can efficiently and effectively process information simultaneously in multiple dimensions. These multiple media modalities/types, that aid effective communication, can be characterized into speech, audio, image, video, and textual data. Recently, there has been a significant amount of interest in digital storage, retrieval, and transmission of multimedia data. This is mainly driven by technological breakthroughs (such as powerful processors, high-speed networking, high-capacity storage devices, improvements in compression algorithms, and advances in processing of audio, speech, image, and video signals) which allow us to digitize, store, and transmit audiovisual content in a very cost effective and efficient manner. Multimedia systems are now not only economically feasible, but also highly desirable.

A number of commercial organizations have large image and video collections of programs, news segments, sporting events, paintings, and artifacts that are being digitized for convenient on-line access. The advent of digital photography and dig-

ital video further allows more and more people to have their personal collection of photographs and other audiovisual content available on the Internet. These digital databases are not a dream of the future, but have become a reality. Organizing these digital libraries into a small number of *categories* and providing effective indexing is imperative for accessing, browsing, and retrieving useful data in "real-time".

Due to the huge amount of potentially interesting documents available over the Internet, searching for relevant information has become very difficult. According to the Gilder technology report [1] (Aug, 1999) and the Netsizer review [2] (Jan 21, 2000) there exist over 70 million hosts on the Internet with over 4,000 TeraBytes of information. Images and video are a major source of these data. Current solutions for searching this humungous amount of data primarily deal with textual information; many text-based search engines such as *Yahoo*$^{(TM)}$ [3], *Altavista*$^{(TM)}$ [4], *Lycos*$^{(TM)}$ [5], *Excite*$^{(TM)}$ [6], *Google*$^{(TM)}$ [7] etc. are available on the World Wide Web. These search engines are among the most visited sites on the web, indicating a huge demand for efficient search tools on the Internet. However, identifying "keywords" for audiovisual content is a difficult problem as no generally recognized description of this material exists. For example, it is currently not possible to automatically search for a particular picture, say, "Arnold Schwarzenegger on a motorbike in Terminator", or a "hilarious ten minute video scene". As another example, a basketball coach might be interested in querying a video database to retrieve a particular type of play for developing a defensive strategy for his team. Can a basketball video be automatically segmented into meaningful "plays" (e.g., defensive and offensive strategies, foul plays, dunks, etc.) to support such queries? These numerous applications (and also

other applications in stock video, satellite imaging, medical imaging, education and distance learning, etc.) have identified the need of a solution to the problem of efficiently searching for various types of multimedia material of interest to the user. *The goal of this thesis is to extract low-level image features for semantic categorization of image and video content which will lead to an efficient retrieval.*

1.1 Text-based Retrieval Systems

Traditionally, textual features, such as filenames, caption and keywords have been used to annotate and retrieve image databases. However, there are several problems associated with text-based queries. First of all, human intervention is required to describe and tag the contents of the images in terms of a selected set of captions and keywords. In most of the images there are several objects that could be referenced, each having its own set of attributes. Further, we need to express the spatial relationships among the various objects in an image to understand its content. As the size of the image databases grows, the use of keywords becomes not only complex but also inadequate to represent the image content. The keywords are inherently subjective and not unique. Often, the preselected keywords in a given application are context dependent and do not allow for any unanticipated search. If the image database is to be shared globally then the linguistic barriers will make the use of keywords ineffective. Another problem with this approach is the inadequacy of commonly used textual descriptions of attributes such as color, shape, texture, layout, and sketch.

Consider the picture of a picnic scene shown in Figure 1.1. Various objects that

are associated with the picture which may later be used for querying, could be the individuals in the picture, the clothes they are wearing, their gender, the color of their clothes, the background in the image, the activity of the people in the image (standing, sitting, laughing, eating, etc.), the spatial relationships between the various people, and so on. There doesn't seem to be a standard textual representation of all these attributes. Later, while querying the database, a user can think of a number of other textual attributes to describe this picture, such as, "Show me a picture where Joe Smith is wearing a t-shirt with a PRIP logo", which had not been tagged with the images in the database. A larger number of attributes of keywords is typically needed to represent an image.

Figure 1.1: A photograph of a picnic scene.

For manual image retrievals, we generally do not store textual descriptions for various images, but have a general notion of what an image contains. One of the main goals in content-based retrieval research is to automatically extract features that aid in representing and retrieving multimedia data. For example, face detection followed by face recognition can be used to extract information of people in a scene (such

as in Figure 1.1). Current computer vision techniques are not able to automatically extract and identify objects in an image. State-of-the-art face recognition systems can detect and recognize only frontal faces in images [8]. Similarly, text detectors can be used to locate regions of text in images, provided the individual characters are significantly large and have a high contrast. An OCR can then be used to identify the text. Thus, queries like "Show me a picture where Joe Smith is wearing a t-shirt with a PRIP logo" can be retrieved based on automatic face and text recognition (which are currently hot topics of research).

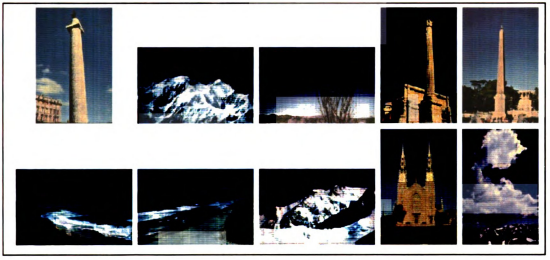
1.2 Content-Based Image Retrieval Systems

Due to the limitations of the traditional text-based systems in providing indices to digital libraries, there has been an intense activity in developing multimedia retrieval methods based on automatically extracting features from the visual content. Various systems have been developed for content-based image and video retrieval, the most prominent of these being QBIC [9], Photobook [10], FourEyes [11], SWIM [12], Virage [13], ViBE [14], VideoQ [15], Visualseek [16], Netra [17], and MARS [18]. These systems follow the paradigm of representing image and video content using a set of low-level attributes, such as color, texture, shape, layout, and global motion which are archived along with the multimedia database. A retrieval is performed by matching the features of a query image or video clip with features of the database documents. Users typically do not think in terms of low-level features while querying digital databases, i.e., user queries are typically based on semantics (e.g., "show me a

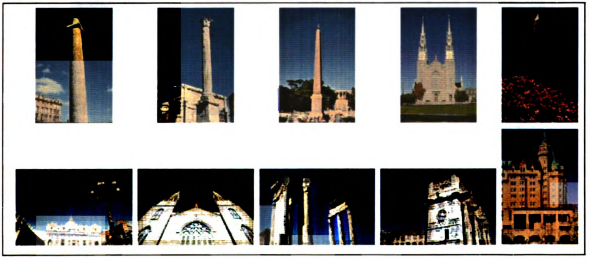
sunset image") and not on low-level image features (e.g., "show me a predominantly red and orange image"). As a result, most of these multimedia retrieval systems have poor performance for specific queries. For example, Figure 1.2(b) shows the top-10 retrieved results (based on color histogram features) for the query in Figure 1.2(a) on a database of 2,145 images of city and landscape scenes. While the query image has a specific monument (a tower here), some of the retrieved images in Figure 1.2(b) include scenes of mountains and coasts. Thus, there is a need for a higher level of abstraction (a semantic level) to aid in effective and efficient search of the multimedia data. A successful grouping of the database images into semantically meaningful classes [19] can greatly enhance the performance of a content-based image retrieval system. Figure 1.2(c) shows the top-10 retrieved results (again based on color histogram features) on a database of 760 *city* images for the same query. These results show that filtering out landscape images from the image database prior to querying can improve the retrieval results. Similarly, Figures 1.3(b) and (c) show top-10 retrieved results (based on color histogram features) for a landscape query in Figure 1.3(a) on a database of 2,145 city and landscape images and 1,386 landscape images, respectively. Again, it can be seen that filtering out the city images from the database prior to querying improves the retrieval results in Figure 1.3(c).

The limitations of current "content-based" image and video retrieval systems have led to a need to model human perception of image content to improve the accuracy and efficiency of retrievals. One method to decode human perception is through the use of a relevance feedback mechanism [20], where the user and the system can interact with each other to improve the retrieval performance. Although, the use of relevance

(a)



(b)



(c)

Figure 1.2: Color-based retrieval results; (a) query image; (b) top-10 retrieved images from 2,145 city and landscape images; (c) top-10 retrieved images from 760 city images; filtering out landscape images prior to querying improves the retrieval results.

(a)

(b)

(c)

Figure 1.3: Color-based retrieval results; (a) query image; (b) top-10 retrieved images from 2,145 city and landscape images; (c) top-10 retrieved images from 1,386 landscape images; filtering out city images prior to querying improves the retrieval results.

feedback is important to improve retrieval results, there is still a need for extracting high-level semantic information from multimedia data. Various psychological studies and experiments [21, 22, 19] have confirmed this, leading to a renewed interest in extracting semantic (or pseudo-semantic) information from image and video data [11, 23, 24, 25, 26, 19, 27, 28, 29, 30]. Most of the current systems manually insert these semantic tags into image databases. Queries combining both semantic tags (keywords) as well as low-level image features are then used for efficient retrieval. To highlight this issue, we present retrieval results using the $QBIC^{(TM)}$ image retrieval engine [31]. Figures 1.4(a) and (b) show the retrieval results for the query image (top left image in each figure) using color and texture features, respectively. Adding semantic tags like the keyword "people" improves the retrieval results as seen in Figures 1.5(a) and (b). Automatic face detection techniques [8] can be used to locate the presence of people in an image which can then be tagged along with the image. As another example, Figure 1.6 shows the retrieval result using *QBIC* for a *sunset* query image (top left image in the figure) using color histogram based search. When the keyword "sunset" is added to the query, the retrieval results become more meaningful as seen in Figure 1.7. Manually tagging each database image is not only tedious, but is highly subjective. For example, the author feels that the third and fourth images in the top row and the first image in the second row in Figure 1.6 are sunset images. However, since these were not tagged with the "sunset" keyword in the database, they were not retrieved in Figure 1.7. Thus, there is a need for an automatic and objective means of assigning semantic indices to images in a database.

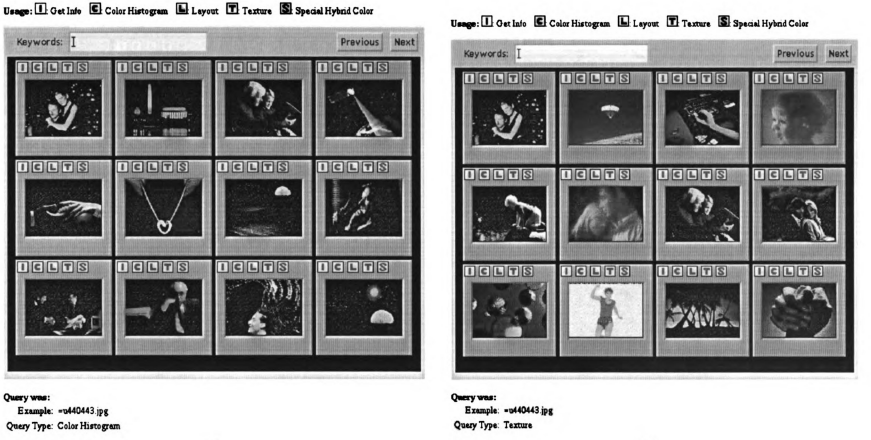
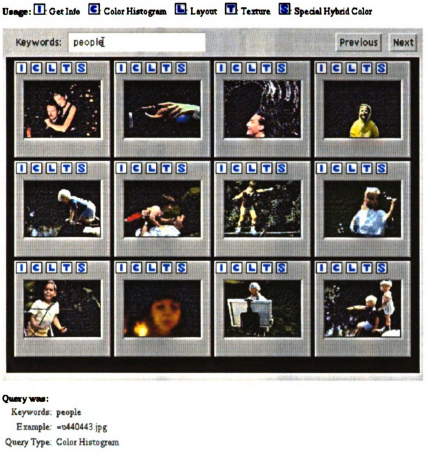
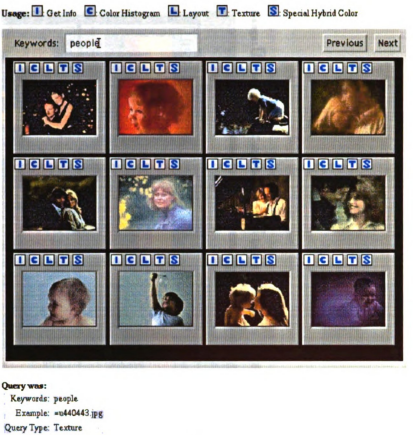


Figure 1.4: Retrieval result of $QBIC^{(TM)}$ image retrieval engine on a query image (top left image) using (a) color histogram and (b) texture features.



(a)



(b)

Figure 1.5: Retrieval result of $QBIC^{(TM)}$ image retrieval engine on a query image (top left image) using the keyword, *people*, along with (a) color histogram and (b) texture features; use of a keyword (*people*) improves the retrieval result as compared to Figures 1.4(a) and (b).

Figure 1.6: Retrieval result of $QBIC^{(TM)}$ image retrieval engine on a sunset query image (top left image) using color histogram features.

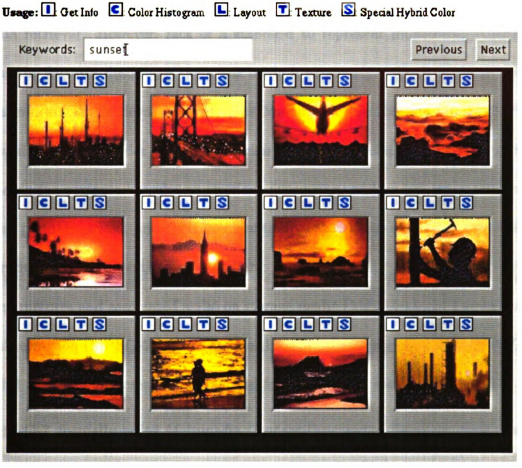


Figure 1.7: Retrieval result of $QBIC^{(TM)}$ image retrieval engine on a sunset query image (top left image) using the keyword, *sunset*, along with color histogram features; use of a keyword (*sunset*) improves the retrieval result as compared to Figure 1.6.

1.3 Image Storage Standards

It is generally agreed that a combination of textual features, low-level content-based features (color, texture, etc.), and semantic categories is necessary for efficient multimedia retrieval. However, lack of a standard to represent multimedia content as well as the features describing them has made it extremely difficult to design multimedia information retrieval systems. The Moving Picture Experts Group (MPEG) is a working group under ISO/IEC in charge of the development of international standards for compression, decompression, processing, and coded representation of moving pictures, audio, and their combination. The various standards that MPEG has produced (or is currently producing) are: (i) a standard for storage and retrieval of moving pictures and associated audio on storage media (MPEG-1); (ii) a standard for digital television (MPEG-2); (iii) a standard for multimedia applications (MPEG-4); and (iv) a content representation standard for information search (MPEG-7).

MPEG-7, formally called *Multimedia Content Description Interface*, aims to standardize the description of various types of multimedia information, associated with the content itself, to allow fast and efficient searching of material that is of interest to the user [32, 33, 34, 35]. According to MPEG-7, multimedia data should be appended by a standardized set of words, descriptions, or features representing a rich concept, that can be related to several levels of abstraction, offering the possibility of different levels of discrimination of image and video. This implies that the same image/video can be described using different types of features, tuned to application of interest. For example, descriptions for video may include a lower abstraction level of color,

texture, shape, layout, and motion features, whereas semantic information about the scene could be coded at the highest level of abstraction. For the picnic image shown in Figure 1.1, MPEG-7 descriptors could be defined as follows:

- Low-Level Features: local and global color, texture, and shape features.
- Semantic Descriptors: Picnic scene, names of people present, type of clothes they are wearing, activity of people in the image (standing, eating), location where picture was taken (X park), time the picture was taken, name of the photographer, occasion for the picnic, etc.

While the low-level features can be automatically extracted, they do not carry sufficient information for efficient and accurate retrieval. Hence, there is a need to index the multimedia data with semantic descriptors of the scene. However, not all of the high-level descriptors, which carry rich information about the scene, can be automatically extracted. For example, from the image alone, the name of the photographer or the location where the image was taken cannot be identified. On the other hand, face and text detectors can automatically identify some of the high-level descriptors that can be tagged with the above image.

Figure 1.8 shows the MPEG-7 processing chain. It includes feature extraction (analysis), the description itself, and the search engine (application). Automatic extraction of image features (or descriptors) will be extremely useful to generate MPEG-7 descriptions. However, with the current state-of-the-art computer vision algorithms, automatic extraction of features is not always possible, especially, at the higher semantic level of abstraction. This is the reason why the scope of MPEG-

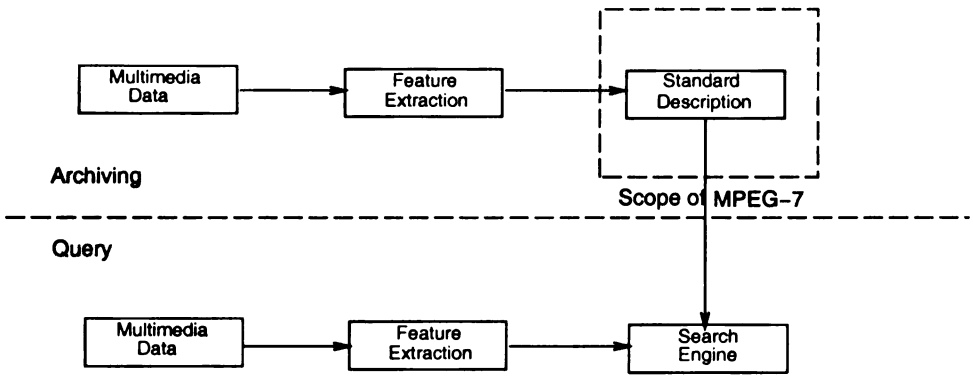


Figure 1.8: MPEG-7 processing chain.

7 does not include the feature extraction process. Rather, MPEG-7 assumes that these descriptions are already available (either automatically extracted or manually annotated). Regardless of how features are extracted from multimedia data, they need to be efficient and effective for content-based retrieval and browsing. The goal of this thesis is to address this issue of automatic assignment of specific high-level semantic tags to images and video frames.

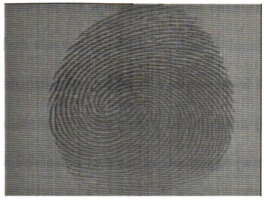
1.4 Indexing and Classification

As discussed above, there is a need for automatic conceptual indexing and categorization of images to enhance the performance of content-based image retrieval systems. Semantic image classification can also be used for *content-based image processing* in smart printers and cameras. Efficient filtering and printing techniques can be employed if the semantic class of an image is known a priori. For example, indoor images are usually close-ups. This information can be used by a smart printer to print the details in an indoor image. However, this rather difficult problem (semantic image classification) has not been adequately addressed in current image database systems.

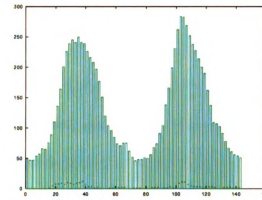
The main problem is that only low-level features can be reliably extracted from images as opposed to higher level features such as the nature of objects present in the scene and their inter-relationships. For example, color histograms can be easily computed for any color image, but presence/absence of sky, trees, buildings, furniture, people, etc., cannot be reliably extracted from general images. One attempt to solve this problem is the hierarchical indexing scheme proposed by Zhang and Zhong [36, 37], which uses a Self-Organization Map (SOM) to perform clustering based on color and texture features. This indexing scheme was further applied in [38] to create a texture thesaurus for indexing a database of large aerial photographs. However, the success of such clustering-based indexing schemes is often limited, largely due to the low-level feature-based representation of image content. For example, Figures 1.9(a)-(d) show two images (a fingerprint and a landscape image) and their corresponding edge direction coherence feature vectors (these features are defined in [19]). Although, these two images denote two very different concepts, their edge direction features are highly similar; the Euclidean distance between the corresponding histograms is only 0.0147 (distances are in the range [0,1]). This shows the limitations of low-level features in capturing semantic content in an image. Yet, as we shall show later, the same edge direction features have sufficient discrimination power for *city vs. landscape* classification. In other words, specific low-level features can be used in constrained environments to discriminate between certain conceptual image classes.

To achieve the goal of automatic categorization and indexing of images in a large database, we need to develop robust schemes to identify salient image features that capture a certain aspect of semantic content of these images. This necessitates an

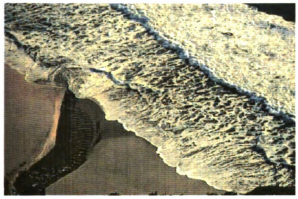
initial specification/definition of pattern classes, so that the database images can be organized in a *supervised* fashion.



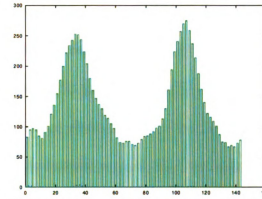
(a) Fingerprint Image



(b) Edge Direction Coherence Feature Vector



(c) Landscape Image



(d) Edge Direction Coherence Feature Vector

Figure 1.9: Edge direction coherence vector features for (a) fingerprint and (c) landscape image; The distance, $d(b, d)$, between these two histograms is 0.0147.

1.5 Proposed Research

Psychophysical and psychological studies have shown that scene identification in humans can proceed, in certain cases, without any kind of object identification [39, 40, 41]. Biederman [39, 40] proposed that an arrangement of volumetric primitives (geons), each representing a prominent object in the scene, may allow rapid

scene identification independently of local object identification. Schyns and Oliva [41] demonstrated that scenes can be identified from low-spatial-frequency images that preserve the spatial relations between large-scale structures in the scene, but which lack the visual detail to identify local objects. In addition, they reported that scene identification when it was presented for a very short duration (40-50 ms) was based more on low-frequency information (global image features) than on high-frequency information (edges) in the image (while both low-frequency and high-frequency features are extracted from the image in the short time duration, low-frequency features were weighted more). Human subjects, when presented with low-frequency representation of an image for a very short duration (40 ms), were able to identify the scene type (city scene, room, etc.) with approximately 85% accuracy. These results suggest the possibility of coarse scene identification from global low-level features before the identity of objects is established. The results further suggest that by attending first to the coarse scale (low-frequency features), the visual system can get a quick and rough estimate of the input to activate plausible scene schemas in the memory, while attending to fine information (high-frequency features) allows refinement, or refutation, of the raw estimate. Thus, for an unknown scene that needs a fast categorization, more salient - but uncertain - low-frequency (coarse) information may be more efficient for an initial estimate of the scene's identity. To quote Navon [42], "forest-before-trees is a better strategy than trees-before-forest". Based on the initial guess, verification can then proceed using more accurate high-frequency features.

The goal of this thesis is to concentrate on the extraction of content-based low-level features and their application in developing semantic categorization of image

and video content to aid in effective and efficient image and video data browsing and retrieval. We do not attempt to understand multimedia content, but we address the issue of making a rough estimate of the scene identity in terms of a number of semantic classes. For example, the picnic scene in Figure 1.1 can be tagged as follows: *outdoor scene, has vegetation (trees and grass), has man-made structures, does not have sky, has people (six), and has text (PRIP).* These semantic tags are then used as indices into the database. Users can now formulate their query in terms of these semantic tags.

Rather than first generating a multi-class classification, we believe that it may be more feasible to perform multiple two-class classifications based on features which have high discriminability for the particular two classes. Thus, we propose to develop multiple binary classifiers for image data which are later combined in a hierarchical fashion to generate semantic indices.

Figure 1.10 shows the proposed system block diagram. The system is divided into archiving and retrieval stages. During archiving, low-level features representing attributes such as color, texture, shape, motion, etc., are extracted from the multimedia data and archived in the database. These low-level features are also used to extract specific high-level scene descriptions for the multimedia data which are then stored along with the low-level features. Figure 1.11 shows our proposed system for building semantic indices into multimedia data using visual cues. An input image is fed to two levels of processing: (i) scene identification based on global image features; and (ii) object detection based on local image features. The *solid* lines show the flow of control in the system. Initially, global low-level features are used for scene classification.

As an example, we show how images can be classified into a hierarchy of semantic classes. Images are first classified into *indoor* or *outdoor* classes. Outdoor images are further classified into *city* or *landscape* classes. A subset of landscape images is further classified into *sunset*, *forest*, and *mountain* classes. The above hierarchy was identified based on experiments with human subjects on a small database of 171 images [19] and is briefly described in Chapter 4.1. In the retrieval and browsing mode, the low-level features are used along with the high-level scene description (semantic tags) to constrain the search space and provide more efficient and accurate retrieval results.

The scene classification information is also used with local image features to detect the presence of specific objects (sky, vegetation, people, etc.). The *dashed* lines in Figure 1.11 show the desired flow of control, but these paths have not currently been implemented in our system. The two modules for scene classification and object detection may in fact interact and coordinate to generate semantic tags. However, we currently rely on the scene information (presumed more reliable) to direct the object detection module and not the other way around. These modules can also send feedback to the feature extraction module if the low-level features being used do not have sufficient discriminatory information for classification. Finally, both the scene classification and object detection modules output semantic tags which are used to generate indices into the image database.

The classification problems formulated above are addressed in this thesis using Bayesian learning and inference mechanism. The probabilistic models required for the Bayesian approach are estimated during the training phase; in particular, the

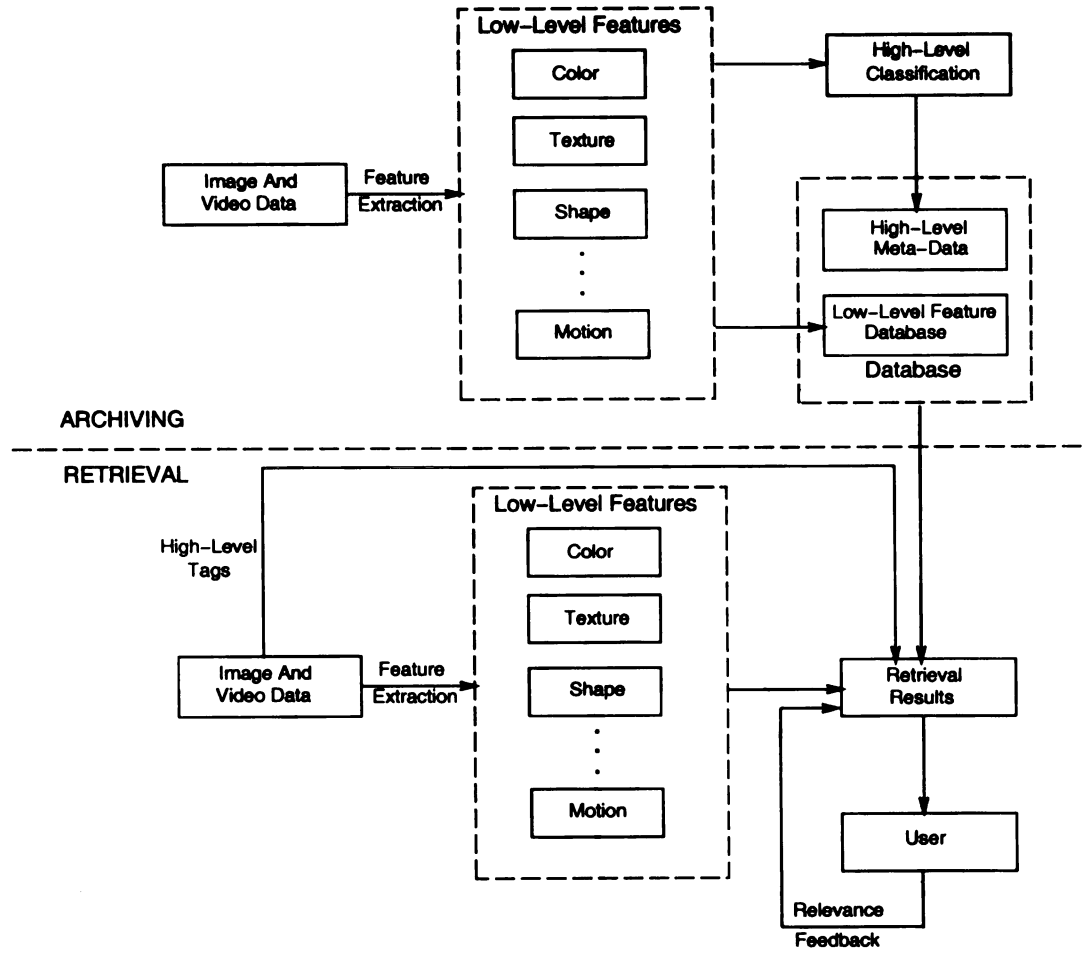


Figure 1.10: Proposed image retrieval system.

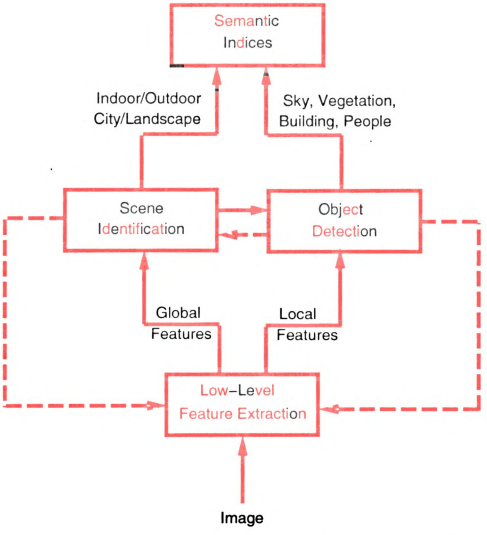


Figure 1.11: Proposed system for automatically building semantic indices in image databases.

class-conditional probability density functions of the observed features are estimated under a Learning Vector Quantization (LVQ) framework [43, 25, 44]. An MDL-type principle [45, 28] is used to determine the optimal size of codebook vectors from the training samples for the various classifiers. The *maximum a posteriori* (MAP) criterion is used to define the Bayesian classifiers.

1.6 Issues in Semantic Classification

A major drawback with content-based retrieval is that images that do seem to be perceptually similar need not be exactly similar in appearance. In fact, two images that describe the same semantic scene (city scene or landscape scene) need not have a high pixel-based correlation value. For example, Figure 1.12 shows four images that belong to the *city* class. These images differ considerably in their pixel-wise correlation value. We (humans) can identify these images as belonging to the city class but it is extremely difficult for a computer vision system to identify and group such images into the same class. As another example, Figure 1.13 shows four landscape images. It can again be seen that automatically developing models for the landscape image class is a very difficult problem. As a final example, Figure 1.14 presents four images belonging to the mountain image class. Although these images are perceptually similar, they differ markedly in their appearance. Given the current state-of-the-art in computer vision algorithms, it is not possible to reliably identify various objects (such as sky, mountains, buildings, etc.) in an arbitrary image. However, we will show in this thesis that constrained problems such as discriminating between global scenes such as city and landscape images, or indoor and outdoor images, can be addressed using specific low-level features. Furthermore, we show that the image classification information can be used to extract objects that have a high probability of occurrence in that image. For example, we demonstrate results of extracting regions of sky and vegetation in outdoor images.

Training the classifiers is another major issue in content-based image classification

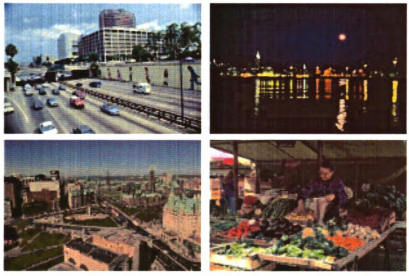


Figure 1.12: A subset of city images.

and retrieval. Humans do not assign a single class label to an image and many of the semantic classes overlap. For example, Figures 1.15(a) and (b) show two images belonging to both city and landscape (sunset) image classes. Modeling this fuzziness in an automatic classifier is a very difficult problem. In order to simplify the problem at hand, we do not consider fuzzy classification of images. Rather, we assume that every image belongs to only one class. This assumption simplifies the generation of training samples. Each image is assigned to the class that it most resembles (as determined by the system designer). Both the images in Figure 1.15(a) and (b) were assigned to the landscape class. Our classifiers then output the final classification result for an input image by assigning it a probability (confidence) of belonging to each of the given classes (which can be used as a fuzzy membership value) and choosing the class with the maximum a posteriori class probability. The images in Figures 1.15(a) and (b) were assigned to the landscape class with a probability of 0.98 and 0.01, respectively, using our city vs. landscape classifier. The result suggests

Figure 1.13: A subset of landscape images.

that Figure 1.15(b) would not be retrieved in case of a query on landscape images, even though the user may be querying for a sunset image.

1.7 Thesis Outline

The thesis is organized as follows. Chapter 2 presents a brief literature survey on image and video database retrieval systems. In Chapter 3, we discuss the Bayesian framework for image classification and present an introduction to Vector Quantization (VQ) and density estimation along with a detailed description of the MDL principle for selecting an optimal codebook size. Chapter 4 discusses the hierarchical classification of vacation images along with the implementation issues and choice of feature sets for the various binary classifiers. Chapter 5 presents another application of the above Bayesian methodology for automatic orientation detection of digitally scanned images. Chapter 6 discusses methods to improve classifier robustness. In Chapter

Figure 1.14: A subset of images from the mountain image class.

7, we develop classifiers to detect specific objects (sky, vegetation, people) in an image. We discuss the efficiency of a retrieval system based on the above classification scheme on a database of approximately 24,000 images in Chapter 8. Chapter 9 finally concludes the thesis and presents directions for future work.

(a)

(b)

Figure 1.15: Fuzzy membership: Two images belonging to both the city and landscape (sunset) image classes; a user assigned these images to the landscape class.

Chapter 2

Literature Review

A number of image and video database retrieval systems that perform retrieval based on low-level image features have been reported in the literature. In the case of image databases, a feature vector which describes various visual cues, such as shape, texture, or color is computed for each image in the database. Given a query image, its feature vector is calculated and those images which are most similar to this query based on an appropriate distance measure in the feature space are retrieved. The feature selection methods proposed in the literature can be broadly classified into two categories on the basis of the approach used for extracting the image attributes [46]. The *spatial information preserving* methods derive features that preserve the spatial information in the image. It is possible to reconstruct the image on the basis of these feature sets. Representative techniques in this category include polygonal approximation of the object of interest, physics-based modeling, and principal component analysis (PCA). The *non-spatial information preserving* methods extract statistical features that are used to discriminate objects of interest. These include features based on

color histograms, color moments, and statistical shape and texture features such as edge direction histograms, Tamura texture features, wavelet-based texture features, etc. In the following section, we briefly review the color-, shape-, and texture-based image retrieval methods.

2.1 Image Features

Most image retrieval systems use shape, texture, and color to represent an image and retrieval is based on the similarity of features derived from these cues. Although, color seems to be a highly reliable attribute for image retrieval, situations where color information is not present in the images require the use of shape and/or texture attributes for image retrieval. Moreover, retrieval based on a single image attribute might lack sufficient discriminatory information warranting a need for the use of multiple low-level image attributes. For example, color-based approaches cannot distinguish between a red apple and a red Ferrari. Additional shape information can very easily distinguish these two objects. We now briefly describe some of the image attributes and the associated low-level features that have been reported in the literature.

2.1.1 Color

Most of the recent work on color feature extraction has concentrated on color histograms. Some of the earlier works include color indexing using histogram intersection [47, 48] and reference color methods [49]. The 3-D histogram intersection

technique described in [47] uses $16 \times 16 \times 16$ bins and the resulting matching is relatively fast. Reference color method [49] improves the performance further by using fewer reference color bins. Color histograms are generally invariant to translation and rotation of the images and normalizing the histograms leads to scale invariance. However, color histograms do not incorporate spatial adjacency of pixels in the image and may lead to inaccuracies in the retrieval. Other schemes such as PCA on color features [10] maintains the spatial adjacencies. More recently, local color histograms and local color moments [50, 19, 28] have been used to capture some spatial color information. A further histogram refinement method that captures spatial coherence in each color bin (color coherence vector) has been proposed in [51]. A color coherence vector is a color histogram refinement scheme that divides each bin into coherent and non-coherent pixels [51]. A pixel in a bin is said to be coherent if it is part of a large similarly-colored region. This concept was further extended to local histograms in [19].

Another issue in color feature extraction is its representation. While the *RGB* color space has been most commonly used, it does not model the human perception of color. *HSV* and *LUV* color spaces better model human color perception. A comparison of color features and color spaces for image indexing and retrieval can be found in [52]. The authors report that while no single color feature or color space is best, the use of color moment and color histogram features in the *LUV* and *HSV* color spaces yielded better retrieval results than in the *RGB* color space.

2.1.2 Shape

Although humans can effectively use color to differentiate among natural objects, many artificial (man made) objects cannot be distinguished on the basis of color alone. Moreover, humans when presented with binary or grayscale images can easily distinguish among these. Various schemes have been proposed in the literature for shape-based retrieval. These include polygonal approximation of the shape [48]; shape matching using relaxation techniques [53], which uses relaxation methods to find acceptable combinations of matches between pairs of angles on two shapes; image representation on the basis of strings [54, 55], which represent the shape of objects as strings and consider string matching techniques for retrieval; comparing images using the Hausdorff distance [56], which measures the extent to which each point of the stored database image lies near some point of the query and vice versa; experiments in point matching techniques [57], which extract a set of distinctive local features from the query and the model, and then match the resulting point patterns; image registration by matching relational structures [58], which uses relational structures to represent images; shape matching based on chord distributions [59], which uses chord length distribution for image matching; image representation using Codons [60], which uses continuous curve segments in an image that are separated by concave cusps to represent the object shape; matching objects using Fourier descriptors [61]; and object matching using invariant moments [62].

The above techniques rely on a single concise feature to describe the shape. A major limitation of using a single shape model in image database retrieval is that

it might not be possible to extract the corresponding features in a given application domain. Moreover, shape-based representation schemes are not generally invariant to large variations of image size, position, and orientation. In order to incorporate invariance to rigid motions (rotation and translation), these methods need to be applied for all possible rotations and translations, thereby, reducing the speed of the retrievals. Vailaya et al. [63] describe a hierarchical scheme where a combination of shape filters can be quickly applied in the first stage with a more detailed matching in the second stage for efficient shape-based retrieval of trademark images. Edge direction histograms and moment invariant shape features act as efficient pre-filters for shape matching. A more computationally intensive matching stage using deformable templates then orders the filtered images according to their similarity to the query trademark.

2.1.3 Texture

Variations of image intensities that form certain repeated patterns are called visual texture [64]. These patterns can be the result of physical properties of the object surface (roughness, peakedness), or be the result of reflectance differences such as the color on a surface. Humans can very easily recognize a texture, yet it is very difficult to define it. Texture analysis is an important and useful area of study in computer vision. Most natural surfaces exhibit texture and it may be useful to extract texture features for querying. For example, images of wood, grass, etc. can be easily classified based on the texture rather than shape or color.

Texture models developed in the literature can be divided into the following four classes [64]. *Statistical methods* define texture in terms of the spatial distribution of grey values. These include the use of co-occurrence matrices [65] and autocorrelation features (extracting repetitive nature of placement of texture elements). *Geometric methods* are characterized as being composed of "texture elements" or primitives. These include Voronoi tessellation features [66] and structural methods that extract the texture primitives [67]. *Model based methods* assume an underlying image model to describe and synthesize texture. These include the use of random field models [68], fractals [69], and SAR texture models [70]. *Signal processing methods* use frequency analysis of the image to classify texture. The schemes include the use of spatial domain [71] and Fourier domain [72] filtering, and the use of Gabor filters and wavelet models [73, 38]. A number of research studies have shown that the Gabor and MSAR (multiresolution simultaneous autoregressive) texture models seem to outperform other texture models in content-based retrieval and indexing [38, 11].

2.1.4 Summary of Low-Level Image Features

Table 2.1 briefly describes the various low-level image features that have been widely used for content-based image retrieval along with their advantages and limitations. In terms of color, the most widely used features are color histograms [47, 48], color moments [50, 28], and color coherence vectors [51, 19]. These features describe the global properties in an image and can be easily extracted from the image. A major limitation is their inability to appropriately represent local spatial information or ob-

jects in an image. Shape features more aptly describe objects in an image. Commonly used shape and contour features are polygonal approximation of the shape [48], shape representation via invariant moments [62, 63], and shape representation via Fourier descriptors [61]. Shape features provide a higher level of abstraction in terms of shape of objects in an image. However, these require good segmentation algorithms to extract objects of interest from an image. Since objects in an image can occur at any scale, orientation, or translation, matching based on shape features is more expensive than based on the color features. Texture features provide an intermediate level of abstraction in an image. Low-level texture features can provide both global and local information, but are hard to define. Commonly used features include the use of co-occurrence matrices [65], multi-resolution simultaneous auto regressive (MSAR) texture features [70], and Gabor filter models [73, 38]. Like color features, texture features are extracted automatically from an image. Moreover, they are limited in their ability in describing the semantic content in the image. Another limitation of texture features is the high computational complexity of matching based on these features.

2.1.5 Image Segmentation

The above defined features can be extracted either globally or locally (from sub-blocks in the image) from an image. However, they are more effective if they are extracted from individual homogeneous regions (segments) in an image. Image segmentation thus plays an important role in image retrieval. Segmenting images into "meaning-

Table 2.1: Qualitative attributes of various low-level image features.

Image Attributes	Image Features	Advantages	Limitations
Color	Histograms, Moments and Coherence Vectors	Can be easily extracted automatically, computationally less expensive, represent global image properties	Inability to appropriately represent local spatial information (object-level), limited description of image semantics
Shape	Polygon Approximation, Invariant Moments, and Fourier Descriptors	Provide a higher level of abstraction in terms of shape of local objects, allow object-level queries	Cannot be automatically extracted (require good segmentation algorithms to extract objects of interest), computationally expensive to provide invariance to rigid object motion
Texture	Co-occurrence Matrices, MSAR Features, and Gabor Filter Features	Can be automatically extracted from an image, provide both global and local information	Can be computationally expensive (feature extraction and matching), hard to define, limited description of image semantics

ful" parts is a very difficult problem. Many techniques have been reported in the literature for image segmentation [74, 66, 73, 70, 75, 76, 77, 78]. A main limitation of these image segmentation methods is that the resulting homogeneous regions may not represent any "meaningful" object of interest. In fact, most semantic objects have heterogeneous regions. In this thesis, we will also address the issue of texture representation of semantic concepts such as sky and vegetation, without attempting image segmentation.

2.1.6 Combining Multiple Features

Retrieval based on a single image attribute lacks sufficient discriminatory information and might not be able to accommodate large scale and orientation changes. For large databases, we need to extract multiple features for querying the database. A number of studies have been carried out which combine the various features for efficient and effective querying by image content. These systems are briefly described in Section 2.3.

2.2 Feature Extraction from Video

The paradigm of using visual cues has also been extended for video database retrieval by several researchers [79, 37, 80, 81]. A video clip is first segmented into shots (shot detection). Each shot is then represented in terms of a number of (generally few) key frames (key frame extraction) [82, 83, 84, 85]. Features describing the color, texture, and shape content in the key frames along with features describing motion

vectors over a set of frames near the key frames are then used for retrieval. Key frame-based representation does not take into consideration important objects (e.g., tracking basketball players) in the video. The object-based approach to video segmentation and representation represents shots in terms of the objects (key objects) present [86, 87, 15, 88]. Automatic extraction of key objects is in itself a very challenging problem. Object-based representation is thus currently restricted to specific types of videos where extracting objects is relatively simple, e.g., news video where anchor person is an important object, specific sport videos such as tennis, where segmentation of moving players is relatively simple, etc. More recently, a different approach to video database browsing and retrieval based on a systematic integration of all the available media such as video, audio, and close-caption text has been proposed [13, 89, 90]. Although, it seems logical to use multimedia for understanding video, much research is needed in order to understand and analyze each of the individual cues.

While there has been a substantial progress in video database retrieval based on shot detection, retrieval based on a *video clip* (say, a few seconds of video) is only now receiving some attention [91, 92, 93]. A *clip* can be described as a set of shots describing a particular event; for example, a dialogue clip between two people may include interspersed single shots of the individuals involved as well as combined shots showing the two individuals. As another example, a free-throw segment in a basketball video might consist of a close-up of the player, followed by a side view of the court during the free-throw. These shots inherently carry more semantic information than a single shot. Querying on the basis of clips as a unit, attempts to improve the speed of retrievals by avoiding initial segmentation of video into shots and identification of

key frames. We next briefly describe some of the current image and video database retrieval systems.

2.3 Image and Video Database Retrieval Systems

A number of general purpose image and video retrieval systems have been developed. These systems attempt to combine multiple low-level features based on shape, color, texture, and motion cues for retrieval.

- QBIC: The QBIC system [94, 9, 95] has developed methods to query large on-line digital libraries using image and video content as the basis of the query. In other words, the system allows users to search through databases consisting of very large numbers of images using sketches, layout or structural descriptions, texture, color, and sample images. QBIC techniques serve as database filters and reduce the search complexity for the user. These techniques limit the content-based features to those parameters that can be easily extracted, such as color distribution, texture, global shape of an image, and layout. The system offers a user a virtually unlimited set of unanticipated queries thus allowing for general purpose applications rather than catering to a particular application. Color- and texture-based queries are allowed for both images and objects, whereas shape-based queries are allowed only for individual objects and layout-based queries are allowed only for the entire image.

- Photobook: *Photobook* and *FourEyes* [10, 11] provide a set of interactive tools for browsing and searching image sequences. The features used for querying

can be based on both text annotations and image content. The key idea behind the system is *semantics-preserving image compression*, which reduces images to a small set of perceptually significant coefficients. These features describe the shape and texture of the images in the database. Photobook uses multiple image features for querying general purpose image databases. The user is given the choice to select features based on the appearance, shape, and texture to browse through large databases. These features can be used in any combination with textual features to improve the efficiency and accuracy of the retrievals.

- Virage: Similar to QBIC, Virage [13, 96] supports visual queries based on arbitrary combinations of color, composition (color layout), texture, and structure (object boundary information). The users can adjust the weights associated with the atomic features according to their own emphasis. Virage also provides general purpose features (as described above) and problem specific features (e.g., features for face recognition, medical applications, etc.).

- VisualSEEk and WebSEEk: VisualSEEk and WebSEEk [16, 97] are text- and image feature-based search engines. They allow spatial relationship query of image regions as well as visual feature extraction from compressed domain. The visual features used are color sets and wavelet transform based texture features. Binary tree based indexing algorithms are used to speed up the retrieval process.

- Netra: Netra [17, 38] is a prototype image retrieval system using color, texture, shape, and spatial location information in segmented image regions to search and retrieve similar regions from the database images. The main features of

Netra are a set of Gabor filter based texture features [38] and an edge flow based image segmentation scheme [77].

- MARS: MARS [18, 20] uses integration of database management systems (DBMS) and information retrieval (IR), integration of indexing and retrieval tasks, and integration of computer and human expertise for image retrieval. The main goal of MARS is not to find the single best feature representation, but rather on how to organize various visual features into a meaningful retrieval architecture which can dynamically adapt to different applications and different users. MARS formally proposes a relevance feedback architecture to decode human perceptual models.

- ViBE: Video Browsing Environment (ViBE) [14] allows a user to organize large video sequences. A video sequence is initially segmented into shots using the Generalized Trace obtained from DC-sequence of the compressed data stream. Each shot is represented by a hierarchical tree structure of key frames. These shots are also automatically classified into pre-determined pseudo-semantic classes. A similarity pyramid data structure is used to present results to the user. ViBE also uses relevance feedback to decode human models of semantic classes.

- VideoQ: VideoQ [15, 98, 99] presents a real-time, interactive system on the Web for video browsing and retrieval, based on the visual paradigm. Spatio-temporal attributes are the main focus of VideoQ for video retrieval. The system automatically segments and tracks objects and regions of interest based

on color, texture, shape, and motion features. VideoQ allows users to query based on animated sketch using multiple objects and their trajectories.

- STAR: System for Trademark Archiving and Retrieval [49] uses a combination of color and shape features for retrieval purposes. The color of an image is represented in terms of the R, G, and B components, whereas the shape is represented in terms of a combination of outline-based features (sketch of the images) and region-based features (objects in an image). The features used to describe both the color and shape in an image are non-information preserving or ambiguous in nature. Although these features cannot be used to reconstruct the image, they are useful as approximate indicators of shape.

- Other: Vailaya et al. [63, 100] proposed a method for trademark image database retrieval based on object shape information. This system achieves both the desired efficiency and accuracy using a two-stage hierarchy: in the first stage, simple and easily computable shape features are used to quickly browse through the database to generate a moderate number of plausible retrievals when a query is presented; in the second stage, the candidates from the first stage are screened using a deformable template matching process to discard spurious matches. Retrieval results show that the top most image retrieved by the system agrees with that obtained by human subjects, but there are significant differences between the ranking of the top 10 images retrieved by the system and the ranking of those selected by the human subjects. This demonstrates the need for developing shape features that are better able to capture human perceptual similarity of

shapes. An improved heuristic has been suggested for more accurate retrievals. The proposed scheme matches filled-in query images against filled-in images from the database, thus using only the gross details in the image. Experiments have shown that matching on the filled-in database retrieves more images that have similar content to the query image (as judged by users).

While a number of systems have been developed for image indexing and retrieval, there is a lack of a proper methodology to evaluate their performance. No standard datasets and sets of queries exist that can provide comparison between the existing approaches. Moreover, the semantic nature of the queries confounds the issue. Identifying ideal retrieval results for a query image is not only subjective, but also time consuming. Therefore, most of the above systems do not report quantitative analysis of their performance.

Most of these above systems and much of the past research has concentrated on the low-level feature extraction stage. Although, these features can be extracted automatically, they have their limitations for content-based image and video retrieval. These features are hard-coded into the system and are usually application specific. For different applications, the features and the associated weights have to be specifically tweaked for efficient performance. A typical user of these systems does not have the basic knowledge of feature extraction and thus, is unable to use the system effectively. In fact, users have to be trained well enough before they can effectively use the system. Due to the presence of a large variety of images (natural scenes, man made objects, etc.), the user needs to use effective features that are most expressive in

identifying the class of the query object and retrieve stored images from that class. For example, while querying for scenes containing buildings, it is more meaningful to query on the basis of texture than on the basis of color, but while trying to identify images of plants and grass, it may be more meaningful to query on the basis of green color and texture. Since the user has to specify a combination of features to use for querying, the retrieval results critically depend on the ability of the user to identify expressive features for the query. This has led to an interest in automatically selecting image features for retrieval purposes and automatically extracting certain high-level semantics from image and video data.

2.4 Incorporating Learning

A general purpose image database system should be able to automatically select salient image features for retrieval purposes [101]. Picard and Minka [23] describe an interactive learning system using a society of models. Instead of requiring universal similarity measures or manual selection of relevant features, this approach provides a learning algorithm for selecting and combining groupings of the data, where these groupings are generated by highly specialized and context-dependent features. The selection process is guided by a rich user interaction where the user generates both positive and negative retrieval examples (*relevance feedback*). A greedy strategy is used to select a combination of existing groupings from the set of all possible groupings. These modified groupings are generated based on the user interactions and over a period of time, these replace the initial groupings that have a very low weightage.

Thus, the system improves on its performance over a period of time through user interaction and feedback. For a formal and comprehensive work on use of relevance feedback for image retrieval, the reader can refer to [20].

2.5 Identifying Semantic Categories

Another major challenge in content-based retrieval is that users typically think in terms of semantic concepts. Unless these semantic concepts are identified in the image and video data, retrieval cannot be efficient and effective (one of the goals of MPEG-7). This has led to some recent research in extracting conceptual information from multimedia data. Table B.1 in Appendix B briefly tabulates results of various image classification systems reported in the literature.

A number of attempts have been made to understand high-level semantics from images using low-level features. Yiu [102] and Szummer and Picard [24] propose algorithms for indoor-outdoor scene classification. Yiu reports results on a database of 500 images and uses color and dominant directions to do the classification. Szummer and Picard show results on a larger database of $1,343$ images. They propose a combination of color and texture (MSAR - multiresolution, simultaneous autoregressive model [70]) features on 4×4 blocks of images to do the classification. These systems report classification accuracies of approximately 90%.

Forsyth et al. [103] use specialized grouping heuristics to classify coherent regions in an image under increasingly stringent conditions to recognize objects in the image. They demonstrate recognition of trees by fusing texture and geometric properties,

and learn blob-like landscape concepts using grouped features. However, they do not report the efficiency of classification or the accuracy of the retrievals on a large database based on classification of regions into blobs.

Yu and Wolf [104] use one-dimensional hidden Markov models along horizontal and vertical blocks of images to do scene classification. Quantized color histogram vectors are used as features for the image sub-blocks and the hidden Markov models are used to learn statistical templates from examples. The scheme suffers from the drawback that the one-dimensional model does not capture spatial relationships well.

Gorkani and Picard [105] have proposed the use of dominant orientations found using a multiscale steerable pyramid in 4×4 sub-blocks of 98 images to discriminate between city/suburb scenes from photos of landscape scenes. They classify an image as a city scene if a majority of sub-blocks have a dominant vertical orientation or a mix of vertical and horizontal orientations. Their database consisted of vacation photographs provided by British Telecom (BT). Their system misclassified 7 out of the 98 database images.

Torralba and Oliva [29] propose the use of discriminant structural templates for organizing scenes along various semantic axes. They classify the global scene represented by an image along two axes; (i) *degree of naturalness* which represents artificial vs. natural images; and (ii) *degree of openness* which represents panoramic views (e.g., coast, beach, long distant city shots) vs. closed environments (such as forest scenes, or close-up of tall buildings). A supervised learning stage using linear discriminant analysis (LDA) is used to generate the decision boundaries along the various semantic axes. The classification is further based on Gabor texture features extracted either

from the entire image or from sub-blocks in the image. The authors report how different semantic image classes such as beach, city centers, skyscrapers, etc., appear along these semantic axes.

Ratan et al. [30] have developed a multiple-instance learning scheme to model ambiguity in supervised learning examples in natural scenes. Each image can represent multiple concepts. To model these ambiguities, each image is modeled as a bag of instances (sub-blocks in the image). A bag (image) is labeled as a positive example of a concept if there exist some instances representing the concept (a concept could be a car, a waterfall scene, etc.). A bag is labeled as a negative example if there exist no instances representing the desired concept. Using a small collection of positive and negative examples, the authors propose to learn the concept (in terms of relevant sub-blocks) and use it to retrieve images containing a similar concept from large databases.

2.6 Discussion

Image and video features that can be reliably extracted are low-level features. Systems have been developed that use multiple low-level features for retrieval and browsing. Users typically think in terms of high-level semantic concepts when querying and browsing multimedia databases. There is thus, a need to extract these concepts and provide annotations for the multimedia data. These automatically extracted annotations can then be coded along with the data as proposed in MPEG-7.

Extracting high-level conceptual information from low-level image features is a

challenging problem. The concepts that users are interested in depend on the intended application and the current user needs. Automatically, extracting all such relevant information may not be possible. However, we shall show in this thesis that certain semantic concepts can be identified in a constrained environment.

Chapter 3

Bayesian Learning Framework

Our goal is to provide semantic indices into image databases to aid in effective retrieval of multimedia data. We approach the problem in a *data driven* fashion. Rather than build rules for extracting semantics from an image, we build binary classifiers based on a set of low-level features extracted from the image data. Conceptual classification is thus based on learning, in a *supervised* fashion, the distribution of these low-level features under the various classes of interest, from a rich source of training data.

A number of learning schemes have been reported in the literature that can be used for supervised classification. Due to its simplicity, ease of use, and ability to combine different sources of information, we use a Bayesian learning and inference framework for our binary classifiers. Bayesian theory provides a formal (probabilistic) framework for image classification problems. It requires that all assumptions be explicitly specified to build models (observation model, prior, loss function) which are then used to derive an "optimal" decision/classification rule. Optimality here means that, under the assumed models, there does not exist any other classification

rule which has a lower expected loss. The Bayesian paradigm has been successfully adopted in a number of image analysis and computer vision (both low and high level) problems, such as restoration, segmentation, and classification (see [106, 107] and the references therein). However, its use in content-based retrieval from image databases is just being realized [25].

3.1 Basic Elements

The Bayesian framework requires that all the entities involved in decision making be adequately formalized:

- Each observed image $\boldsymbol{x}$ belongs to a set $\mathcal{I}$ of possible images.
- The set $\mathcal{I}$ is assumed to be partitioned into K classes, $\Omega = \{\omega_1, \omega_2, ...\omega_K\}$; these classes are exhaustive and mutually exclusive, i.e., any image $\boldsymbol{x}$ from $\mathcal{I}$ belongs to one and only one class.
- Each observed image $\boldsymbol{x}$ is modeled as an observation of a random variable $\mathbf{X}$, whose class-conditional probability density function for class ω_i is written as $f_{\mathbf{X}}(\boldsymbol{x}|\omega_i)$.
- An *a priori* knowledge concerning the classes is expressed via a probability density function defined on the set of classes, $\{p(\omega_1), p(\omega_2), ..., p(\omega_K)\}$, with $\sum_{i=1}^{K} p(\omega_i) = 1$.
- A loss function, $\mathcal{L}(\omega, \hat{\omega}) : \Omega \times \Omega \rightarrow \mathcal{R}$, specifying the loss incurred when class $\hat{\omega}$ is chosen and the true class is ω. As is common in classification problems, we

adopt the "0/1" loss function; $\mathcal{L}(\omega, \omega) = 0$, and $\mathcal{L}(\omega, \hat{\omega}) = 1$, if $\omega \neq \hat{\omega}$.

- Finally, the solution of the classification problem is a decision rule $\delta(\boldsymbol{x}) : \mathcal{I} \rightarrow \Omega$ which maps any possible observed image into one of the available classes.

3.2 Image Features

In many image analysis problems, it is typical that the classification is based on, say, m features extracted from the observed image, rather than directly on the raw pixel values. Let $\boldsymbol{y} = \{y_1, y_2, \ldots, y_m\}$ denote the set of m features based on which the classification procedure must operate. As a result, the class-conditional density function can be written as

$$f_{\mathbf{X}}(\boldsymbol{x} \mid \omega) \equiv f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega). \tag{3.1}$$

The classification problem can be stated as: "given a set of observed features, $\boldsymbol{y}$, from an image $\boldsymbol{x}$, classify $\boldsymbol{x}$ into one of the K classes in Ω".

3.3 Classification Rule

In the Bayesian framework, all inferences are based on the *a posteriori* probability function, which is obtained by combining the class-conditional observation models with the *a priori* class probabilities (Bayes law):

$$p(\omega \mid \boldsymbol{y}) = \frac{f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega)\, p(\omega)}{f_{\mathbf{Y}}(\boldsymbol{y})}, \tag{3.2}$$

where the denominator, $f_{\mathbf{Y}}(\boldsymbol{y})$, in Eq. (3.2) is the unconditional (or marginal) probability density function of the observed features, which serves simply as a normalizing constant.

The "0/1" loss function leads to the most common criterion in Bayesian classification problems: choose the class whose *a posteriori* probability is maximum. This is known as the *maximum a posteriori* (MAP) criterion, and is given by

$$\hat{\omega} = arg \max_{\omega \in \Omega} \{p(\omega \mid \boldsymbol{y})\} = \arg \max_{\omega \in \Omega} \{f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega)\, p(\omega)\}\,. \tag{3.3}$$

In addition to reporting the MAP classification of a given image, say ω_k, the Bayesian approach also assigns a degree of confidence to that classification, which is proportional to $p(\omega_k \mid \boldsymbol{y})$. We next describe a procedure to estimate the class-conditional density functions.

3.4 Density Estimation Using Vector Quantization

The performance of the Bayes classifier clearly depends on the ability of the feature set $\boldsymbol{y}$ to discriminate among the various classes. Moreover, since the class-conditional densities have to be estimated from training data, the accuracy of these estimates is also critical. Choosing the right set of features for a given classification problem is a difficult problem which requires expert knowledge about the problem domain. We concentrate instead on estimating the class-conditional densities for which we adopt a Vector Quantization (VQ) based approach [44].

3.4.1 Vector Quantization

Vector Quantization (as its name implies) is a compression/quantization technique that is applied to vectors rather than scalars. Just like scalar measurements can be quantized by rounding off or setting thresholds, VQ quantizes a group of measurements (components of a feature vector) together. Thus, VQ takes as input a p-dimensional vector and quantizes it into a p-dimensional *reproduction vector*. A VQ can be specified by a set of reproduction vectors and a rule for mapping input vectors to the reproduction vectors.

In the compression and communication applications, a Vector Quantizer is described as a combination of an encoder and a decoder. A p-dimensional VQ consists of two mappings: an encoder γ which maps the input alphabet ($\mathbf{A}$) to the channel symbol set ($\mathbf{M}$), and a decoder β which maps the channel symbol set ($\mathbf{M}$) to the output alphabet ($\hat{\mathbf{A}}$), i.e., $\gamma(\boldsymbol{y}) : \mathbf{A} \rightarrow \mathbf{M}$ and $\beta(\boldsymbol{v}) : \mathbf{M} \rightarrow \hat{\mathbf{A}}$. A distortion measure $\mathcal{D}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ specifies the cost associated with quantization, where $\hat{\boldsymbol{y}} = \beta(\gamma(\boldsymbol{y}))$. Usually, an optimal quantizer minimizes the average distortion under a size constraint on $\mathbf{M}$. The generalized Lloyd algorithm for vector quantization uses the mean square error (MSE) criterion for distortion and is equivalent to a K-means clustering algorithm [108], where K is the size of the output alphabet, $\hat{\mathbf{A}} : \{\hat{\boldsymbol{y}}_i, i = 1, \ldots, K\}$. An input vector $\boldsymbol{y} \in \mathbf{A}$ is quantized into one of the K output vectors $\hat{\boldsymbol{y}}_i$, also referred to as *codebook vectors*, such that

$$\mathcal{D}(\boldsymbol{y}, \hat{\boldsymbol{y}}_i) \leq \mathcal{D}(\boldsymbol{y}, \hat{\boldsymbol{y}}_j), \ \forall \ 1 \leq j \leq K. \tag{3.4}$$

These codebook vectors define a partition of the p-dimensional feature space, according to Eq. (3.4), into the so-called Voronoi cells, $\{S_i,\ i = 1, 2, \ldots, K\}$. Figure 3.1 shows an example of such a 2-D Voronoi tessellation where the $\hat{\boldsymbol{y}}_i$ are shown as square dots. As the data points get closer, the convex cells around the points become more compact. According to Eq. (3.4), an input vector is assigned the codebook vector of the cell it falls into. A comprehensive study of VQ, choice of distortion measures, and use of VQ in classification and compression (along with further references) can be found in [109, 44].

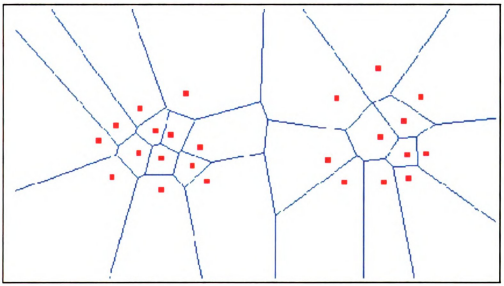


Figure 3.1: Voronoi Tessellation for 2-D data points.

3.4.2 VQ as a Density Estimator

Vector quantization provides an efficient tool for density estimation [44]. Consider n training samples from a class ω. In order to estimate the class-conditional density of the feature vector $\boldsymbol{y}$ given the class ω, i.e., $f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega)$, a vector quantizer is used to extract q (with $q < n$, hopefully $q \ll n$) codebook vectors, $\boldsymbol{v_j}$ $(1 \leq j \leq q)$, from the

n training samples. It has been shown (see [44]) that in the so-called high-resolution approximation (i.e., for sufficiently small Voronoi cells), the class-conditional density can be approximated as a piecewise-constant function over each cell S_j, with value

$$f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega) \approx \frac{m_j}{Vol(S_j)}, \tag{3.5}$$

where m_j and $Vol(S_j)$ are the frequency of training samples falling into cell S_j and the volume of the cell S_j, respectively. This approximation fails if the Voronoi cells are not sufficiently small, as is the case when the dimensionality of the feature vector $\boldsymbol{y}$ is large. The class-conditional densities can then be approximated using a kernel-based approach [44, 25], approximating the density by a mixture of Gaussians, each centered at a codebook vector. In most VQ algorithms, the codebook vectors are iteratively selected by minimizing the MSE (mean square error) which is the sum of the Euclidean distance of each training sample from its closest codebook vector. Hence, an identity covariance matrix can be assumed for the Gaussian components used to represent the densities [25], resulting in the following (approximate) class-conditional densities:

$$f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega) \propto \sum_{j=1}^{q} m_j * \exp(-\|\boldsymbol{y} - \boldsymbol{v_j}\|^2/2). \tag{3.6}$$

A more comprehensive approach would be to use the Mahalanobis distance [108] in estimating the codebook vectors; but, if feature dimensionality is high and the number of training samples is small, the estimated covariance matrices are likely to

be singular.

3.4.3 Selecting Codebook Size

A key issue in using vector quantization for density representation is the choice of the codebook size (value of q). It is clear that, given a training set, the VQ-approximated likelihood (probability) of the training set will keep increasing as the dimensionality of the codebook grows; in the limit, we would have a code vector for each training sample, with the corresponding probability equal to one. To address this issue, we adopt the *minimum description length* (MDL) principle [45]. MDL is an information-theoretic criterion which has been used for model selection in several problems in computer vision and image processing (see [110], and references therein). We start by noting that the VQ learning algorithm basically looks for the *maximum likelihood* estimates of the parameters of the mixture in Eq. (3.6). The first key observation behind MDL is that finding an ML estimate is equivalent to finding the Shannon code for which the observations have the shortest code-length [45]; this is so because Shannon's optimal code-length[1], for some set of observations, $\mathcal{Y} : \{\boldsymbol{y}(1), \ldots, \boldsymbol{y}(n)\}$, obeying the joint probability density function $f(\mathcal{Y}|\boldsymbol{\theta}_{(q)})$, is simply [111, 110]

$$L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) = -\log f(\mathcal{Y}|\omega, \boldsymbol{\theta}_{(q)}). \tag{3.7}$$

[1] In bits or ***nats***, if base-2 or natural logarithms are used, respectively [111].

Under the assumption of independent samples, $\boldsymbol{y}(j)$ $(1 \leq j \leq n)$, the joint likelihood in Eq. (3.7) can be written as

$$L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) = -\sum_{j=1}^{n} \log f(\boldsymbol{y}(j)|\omega, \boldsymbol{\theta}_{(q)}), \tag{3.8}$$

where $\boldsymbol{\theta}_{(q)}$ contains the codebook vectors, $\{\boldsymbol{v_j} : 1 \leq j \leq q\}$, and the weights m_j. The second fundamental fact is that the parameters themselves are also part of the code, in the following sense: a code word representing $\mathcal{Y}$ can not be decoded by itself; only a full knowledge of $f(\mathcal{Y}|\boldsymbol{\theta}_{(q)})$ (i.e., of its parameters) allows reconstructing the code and respective decoder. Accordingly, the MDL criterion states that the description code-length to be minimized by the estimate must include not only the data code-length but also the code-lengths of the parameters. The resulting criterion for the choice of q (codebook size) is then

$$\hat{q} = \arg\min_{q} \left\{ L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) + L(\boldsymbol{\theta}_{(q)}) \right\}. \tag{3.9}$$

Finally, concerning the parameter description length, $L(\boldsymbol{\theta}_{(q)})$, the common choice is $L(\boldsymbol{\theta}_{(q)}) = (\zeta(q)/2)\log n$, where n is the sample size and $\zeta(q) = \{q + q \dim(\boldsymbol{y})\}$ is the number of real-valued parameters needed to specify a q^{th}-order model and dim() represents the dimension of the feature space [45]. This is an asymptotically optimal choice, which is only valid when all the parameters depend on all the data, which is not the case in the present problem. The weights m_j are, in fact, estimated from all the data; however, each $\boldsymbol{v_j}$ is estimated from the m_j samples that fall in the associated

cell. Accordingly, we obtain the following *modified* MDL (MMDL) criterion,

$$\widehat{q} = \arg\min_{q} \left\{ L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) + \frac{q}{2}\log n + \frac{\dim(\boldsymbol{y})}{2}\sum_{j=1}^{q}\log(m_j\, n) \right\}, \tag{3.10}$$

where the first term is the negative log-likelihood of the observations, the second one accounts for the weights m_j, while the third one corresponds to the codebook vectors themselves.

To better understand the MDL principle, consider the following 2-class mixture densities:

- Class 1 (ω_1): Bimodal bivariate Gaussian with $\mu_1 = [1\ 1]^T$, $\mu_2 = [4\ 4]^T$, and $\Sigma_1 = \Sigma_2 = I$.

- Class 2 (ω_2): Bimodal bivariate Gaussian with $\mu_1 = [4\ 1]^T$, $\mu_2 = [1\ 4]^T$, and $\Sigma_1 = \Sigma_2 = I$.

The goal of the MDL criterion is to select a model that is both accurate (is able to describe/represent the training data well, Eq. (3.7)) and simple (Occam's razor: simpler the model, better the generalization capability). The data code-length and the parameter code-length (Eq. (3.9)) represent the accuracy and the simplicity of a model, respectively. Figures 3.2(a)-(d) show 200 patterns per class (100 patterns per mixture) for the above mentioned 2-class mixture problem. Codebook vectors extracted from the training data are represented by $*$. Figures 3.3(a)-(c) show the plots for the data, parameter, and combined code-lengths for varying codebook size. While Figure 3.3(a) shows how the data code-length reduces with increase in code-

book size, Figure 3.3(b) shows how the parameter code-length increases. The MDL criterion, Figure 3.3(c) selects the optimal codebook size as that which minimizes the sum of the data and parameter code-lengths. In this case, the codebook size of 4 is the optimal choice which agrees with the data generation model. Alternate unsupervised schemes to estimate the mixture desities have also been reported in the literature [112]. However, much research is required to make these robust under high dimensional feature vectors.

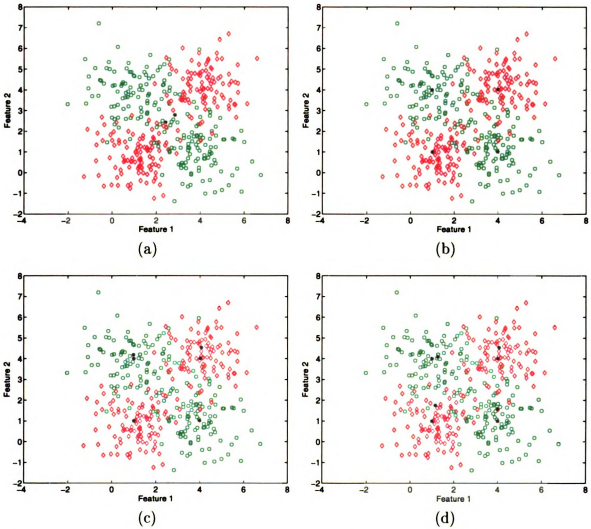
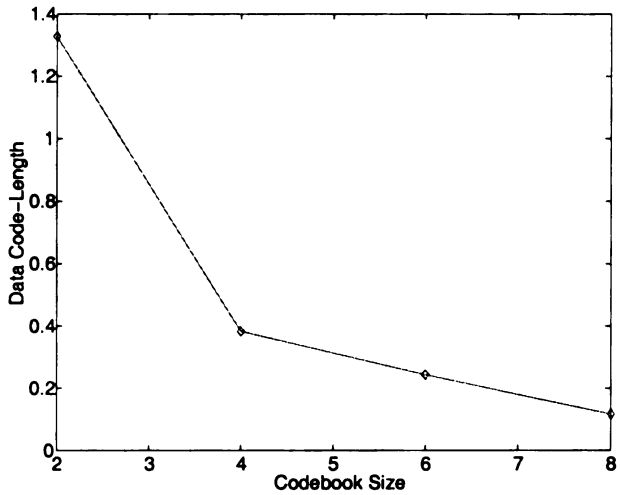
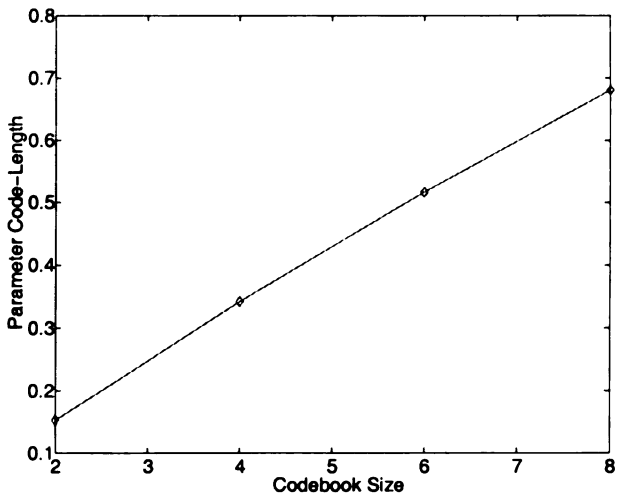


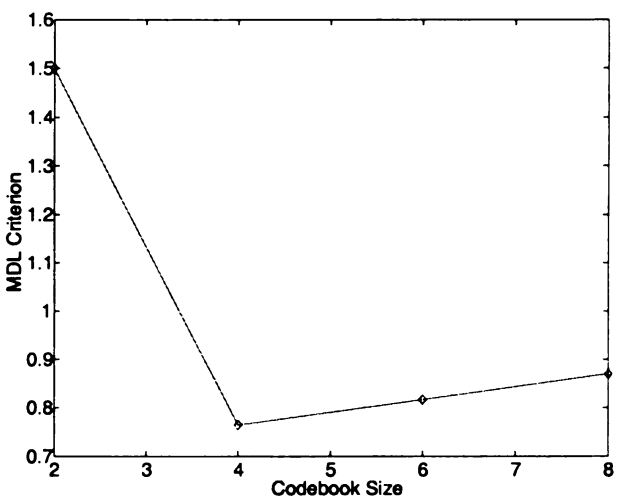
Figure 3.2: Codebook vectors (q) extracted from 2-D mixture of Gaussian data: (a) $q = 2$; (b) $q = 4$; (c) $q = 6$; (d) $q = 8$.



(a)



(b)



(c)

Figure 3.3: MDL Criterion: (a) data code-length; (b) parameter code-length; (c) MDL criterion is the combined code-length (data code-length + parameter code-length).

3.5 Classification Algorithm

We now present an algorithm for the Bayesian classification procedure.

1. Collect a set of images from the various classes of interest and randomly divide them into independent training and test sets. More comprehensive the training and test sets, more robust is the classifier.

2. Identify a set of salient features for the given classification problem. A feature is said to be salient if it has a high discrimination ability. In general, identifying a salient set of features for classification is a subjective process and cannot be easily automated.

3. Extract features from the training set.

4. Use LVQ_PAK [113] Vector Quantization package to extract a set of q, $q_{min} \leq q \leq q_{max}$, codebook vectors per class (q_{min} and q_{max} are empirically set to 5 and 100, respectively; more complex the classification problem, larger is the value of q_{max}).

5. For a given q, identify the weights of each Gaussian in the mixture from the training samples (determined by the number of training samples assigned to each codebook vector).

6. Determine the optimal codebook size by the MMDL principle (Section 3.4.3).

7. Compute the class-conditional probability density as defined in Equation (3.6).

In order to classify a test pattern, extract the features and compute the a posteriori class probability as defined in Eq. (3.2). The MAP classification rule is then used to classify the test pattern.

3.6 Discussion

We present a formal probabilistic framework for image classification problems. The classifiers are built in a supervised fashion under a Bayesian learning and inference framework. Vector Quantization is used to estimate the class-conditional probability densities of the observed features. An MDL-type principle is used to determine the optimal size of codebook vectors from the training samples for the various classifiers. The MAP criterion is used for the final classification of test images. The Bayesian approach has the following advantages: (i) a small number of codebook vectors represent a particular class of images, thereby greatly reducing the number of comparisons necessary for each classification; (ii) it naturally allows for the integration of multiple features through the class-conditional densities; and (iii) it not only provides a classification rule, but also assigns a degree of confidence in the classification which may be used to build a reject option into the classifiers.

Chapter 4

Hierarchical Classification of Images

What do users typically want to search for in an image database? Users either have an a priori idea of what they are looking for (e.g., all images containing a friend's face) or they have a rather vague remembrance of the images they would like to look at (e.g., scenic photographs for planning a vacation) [81]. In the second type of queries, there is a need for a classification of database images based on some semantic categories. The user can then browse through the images belonging to the desired class.

The Bayesian paradigm was applied to obtain a hierarchical classification of images. Images were first classified into indoor and outdoor classes. Outdoor images were then classified into city and landscape classes. Finally, a subset of landscape images was classified into sunset, forest, and mountain classes. We next describe a small-scale experiment conducted on human subjects that formed the basis for identifying the semantic classes used in our hierarchy [19].

4.1 Identifying Semantic Image Classes

The first step in image classification is to identify meaningful image categories that are of interest to users and which can be automatically identified by simple and efficient pattern recognition techniques. For this purpose, we conducted an experiment with 8 human subjects to classify a set of 171 images (images are shown in Appendix A). We asked these subjects to group the 171 color images (including the 98 images used by Gorkani and Picard [105]) into meaningful categories; no a priori categories were specified or made available to the users. The goal of this experiment was to identify semantic and abstract classes that humans assign to these images. The subjects were given no explicit criteria for judging the similarity and they could create any number of categories with any number of images per category. The subjects were also asked to explain the semantic meanings of the groups that they formed and explain the reasons for placing each image into a particular group. The subjects were given unlimited time to complete the task, except that they had to do so in one sitting without consulting anyone. In general, the subjects took between 1-2 hours to complete the task.

The experimental data indicated that the number of categories selected by the eight subjects varied from 7 to 17. Some of the categories formed by various subjects included buildings, streets, cities, bridges, monuments, people, natural scenes, mountains, farms, country side, forests, sunset/sunrise scenes, etc. Of these numerous categories, there were certain categories which were coherent in their semantic descriptions. For example, the class of buildings, streets, and monuments, used by some subjects can be grouped under the *city class* which was created by another

subject. Similarly, the categories such as mountains, forests, farms, and country side, can be grouped under a broader category, called *natural scenes.* Figures 4.1(a) and (b) show representative images from the landscape and city clusters identified by a human subject, respectively.

In order to generate a hierarchical grouping based on the categorization of the human subjects, we generated a dissimilarity measure between every pair of images. A 171×171 dissimilarity matrix was generated, where the dissimilarity value for every pair of images was assigned the number of subjects (out of 8) who *did not* group the pair into the same category (regardless of what the actual category meant for each subject). Note that the integer entries in the dissimilarity matrix vary from 0 to 8. Based on the dissimilarity matrix, we performed a complete-link hierarchical clustering of the 171 images. Note that clusters can be formed by cutting the resulting dendrogram at a specific level of dissimilarity. At the highest level of dissimilarity (by cutting the dendrogram at the dissimilarity level of 7; at level 8, all the images are in one cluster), 11 clusters were formed. The semantic labels that can be assigned to these clusters and the number of images in each cluster are as follows: (i) forests and farmland (37 images), (ii) natural scenery and mountains (19 images), (iii) beach and water scenes (6 images), (iv) pathways (roads and streams, 9 images), (v) sunset/sunrise shots (21 images), (vi) city shots (38 images), (vii) bridges and city scenes with water (6 images), (viii) monuments (9 images), (ix) towers and pictures of Washington DC (5 images), (x) a mixed class of city and natural scenes (20 images; 9 images are long distance city shots, while 11 images are long distance shots of fields), and (xi) a face image (1 image). We refer to the mixed category of long distance city

(a)

(b)

Figure 4.1: Representative images of (a) landscape and (b) city clusters identified by a user.

and landscape shots as the miscellaneous class.

We organized the 11 categories identified above into a hierarchy as shown in Figure 4.2. The first four classes (forests, natural scenery, beach scenes, and pathways) can be grouped into a single class, labeled natural scenes (71 images). The clusters of city shots, monuments, and shots of Washington DC can be grouped into the category of city scenes (58 images). At the next level, the sunrise/sunset group can be classified into the category of landscapes (92 images). Finally, the landscapes, the city scenes, the miscellaneous class, and the lone face image are grouped together into one class (all 171 images). Hence, this rather small database can be considered to have two major classes (city shots and landscapes) at the highest level of grouping which accounts for 150 out of the 171 images.

In order to verify that the above hierarchy is reasonable, we used a multidimensional scaling algorithm to construct a 3-D pattern matrix for the 171 images from the 171×171 dissimilarity matrix. We then applied a K-means clustering algorithm to partition the 3-D data. In our experiments, we used $K = 2$ and $K = 4$. Our goal was to verify if the main clusters in the data agreed with the hierarchy shown in Figure 4.2. Setting $K = 2$, we obtained two clusters of 62 and 109 images, respectively. The first cluster consisted of predominantly city images, while the second cluster consisted of landscape images. Figures 4.3 and 4.4 show images from the city and landscape classes, respectively. Note that the face image was misclassified into the landscape image class. The following clusters were obtained with $K = 4$; (i) city scenes (70 images), (ii) sunrise/sunset images (21 images), (iii) forest and farmland scenes and pathways (49 images), and mountain and coast scenes (31 images). In

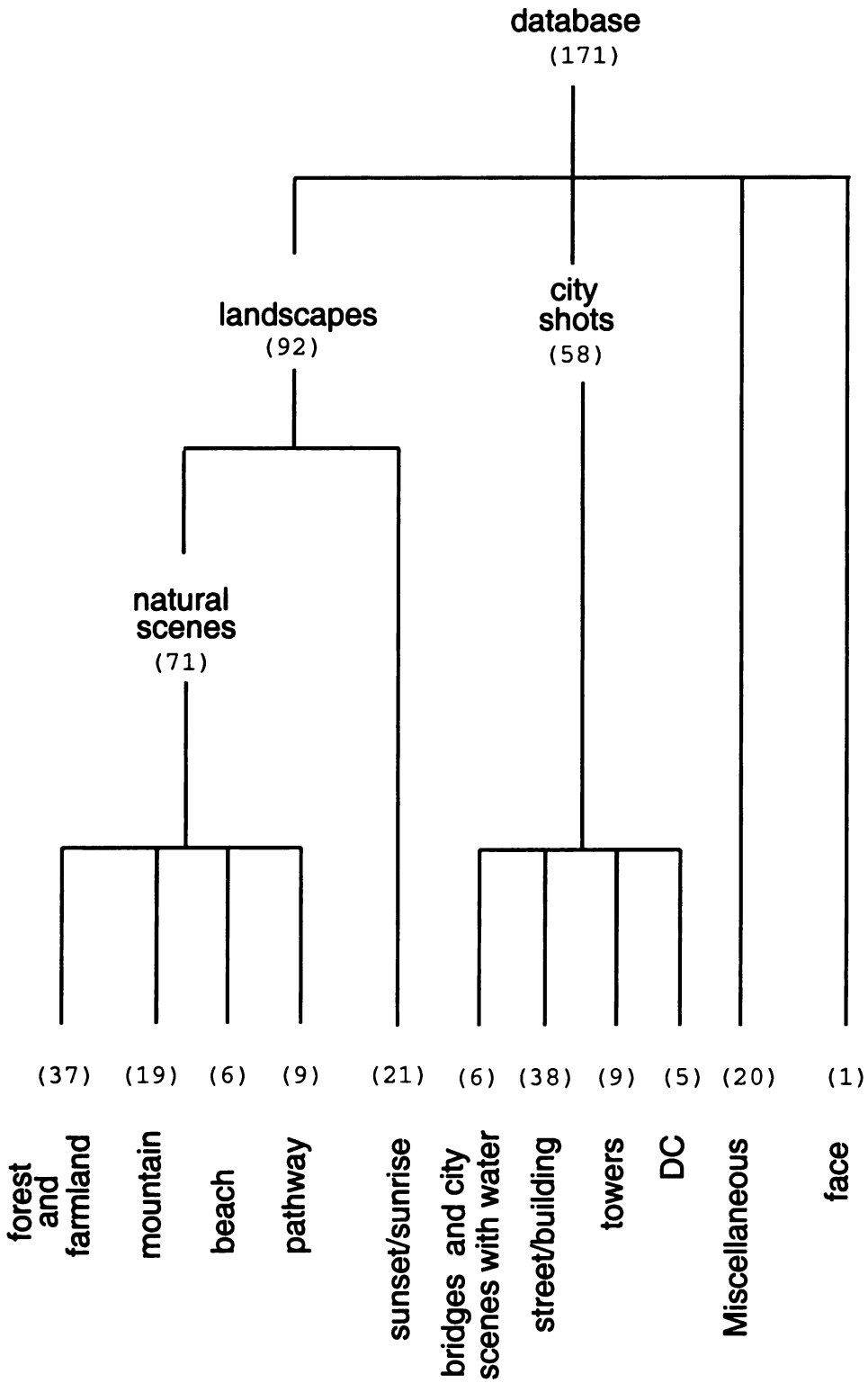


Figure 4.2: A hierarchical organization of the 11 categories obtained from the dendrogram generated by complete-link clustering of 171 images using the dissimilarity matrix provided by 8 users; the number of images in each category is indicated in the parenthesis.

order to view the clusters in the 3-D feature space, we generate plots of the 171 patterns along two axes at a time. Figures 4.5(a)-(c) show the three plots of the 171 patterns in 2-D feature space. The labels of various clusters are marked in the plots. The plots show that the sunrise/sunset image class is very well separated from the other three classes. The mountain image class is well separated from the forest class (and the city class is well separated from the other classes) in two of the three plots. Figure 4.5(a) shows an additional class of water scenes which consists of long distance city and landscape images that contain water bodies (sea and ocean). This class of images is not as well separated in the other two feature subspaces. These experiments verify that our database of 171 images consists of two main classes: *city* and *landscape*. Moreover, the landscape images can be broadly classified into three major categories, i.e., sunrise/sunset images, forest and farmland images, and mountain and coast images, respectively.

Based on the above experiments, we simplified the classification hierarchy as shown in Figure 4.6. The solid lines show the classification problems addressed in this paper. At the highest level, images can be divided into *indoor*, *outdoor*, and *other* images. Outdoor images can then be further classified into *city*, *landscape*, and *other* classes. Landscape images can then be dichotomized into *sunset*, *mountain*, *forest*, and *other* classes. While the above hierarchy is not in itself complete (a user may be interested in querying the database for images captured in the evening - (day/night classification), images containing faces (face vs. non-face classification), or images containing text (text vs. non-text classification)), it is a reasonable approach to simplify the image retrieval problem.

Figure 4.3: Images from the *city* class.

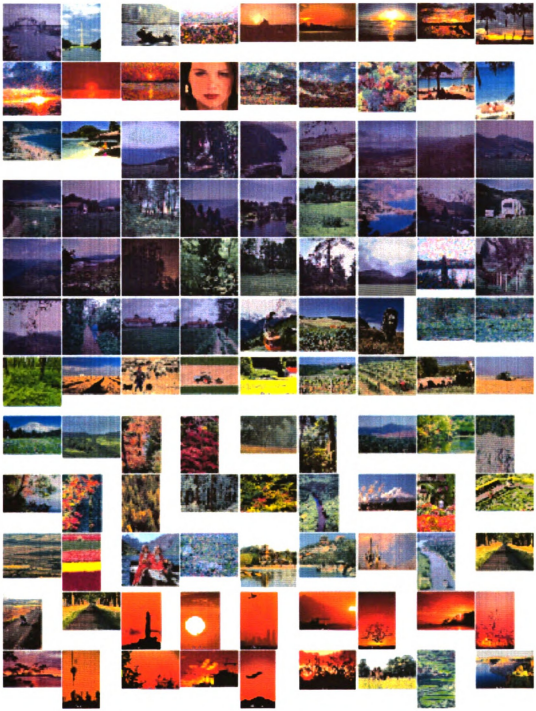
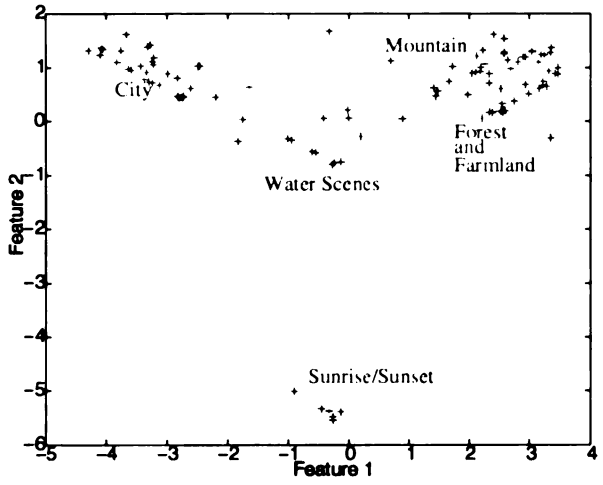
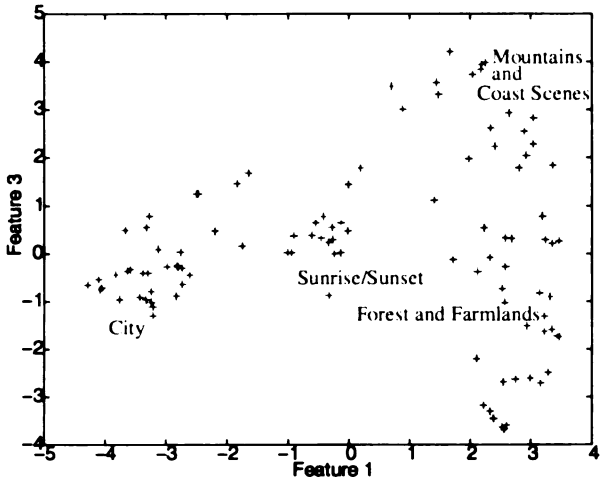


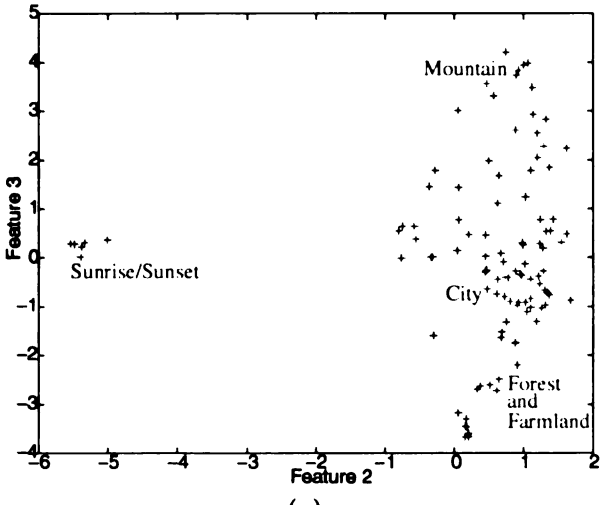
Figure 4.4: Images from the *landscape* class.



(a)



(b)



(c)

Figure 4.5: 2-D plots of the 3-D 171 patterns along two dimensions; (a) along dimensions 1 and 2; (b) along dimensions 1 and 3; and (c) along dimensions 2 and 3.

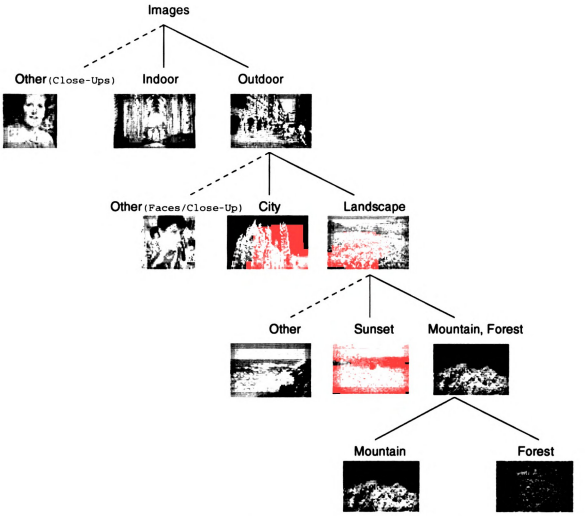


Figure 4.6: Simplified hierarchy of images; solid lines show the classification problems addressed in this thesis.

The indoor vs. outdoor classification problem can be stated as follows: Given an image, classify it as either an indoor or an outdoor image. Most of the images can indeed be classified into one of these two classes. Exceptions include close-up shots, pictures of a window or door, etc. Outdoor images can be further classified into city vs. landscape images [19, 27] which can be posed as follows: Given an outdoor image, classify it as either a city or a landscape image. City images can be characterized by the presence of man-made objects and structures such as buildings, cars, roads,

etc. Natural images, on the other hand, lack these structures. A subset of landscape images can be further classified into one of the sunset, forest, and mountain classes. Sunset images can be characterized by saturated colors (red, orange, or yellow), forest images have predominantly green color distribution due to the presence of dense trees and foliage, and mountain images can be characterized by long distance shots of mountains (either snow covered, or barren plateaus). We assume that the input image does belong to one of the classes under consideration. This restriction is imposed, because automatically rejecting images that do not belong to the specific classes (such as city or landscape) based on low-level image features alone is in itself a very difficult problem (see Figure 1.9).

4.2 Implementation Issues

Experiments were conducted on two databases of 5,081 (indoor vs. outdoor classification) and 2,716 (city vs. landscape classification and a further classification of landscape images) images. The two databases, henceforth referred to as database D1 and database D2, have 866 images in common, thus the entire database contains 6,931 distinct images. These images were collected from various sources (Corel stock photo library, scanned personal photographs, key frames from digitized video of television serials, images captured using a digital camera, and images downloaded from the web) and are of varying sizes (from 150×150 to 750×750). The color images are represented by 24-bits per pixel and stored in JPEG format. The ground truth for all the images was assigned by a single subject. Images in this dissertation are

presented in color. We next describe the low-level features that were extracted from the images.

4.3 Image Features

We extract global image features that capture the general essence of an attribute in the image. These include: local color histograms and coherence vectors [19], spatial color moments [28], edge direction histograms and coherence vectors [19], DCT coefficient features [19], and sub-block MSAR texture features [70, 24]. We next describe these features in detail.

4.3.1 Color Histograms

We define color features in terms of a histogram in the quantized HSV color space. It was reported in [52] that color histograms in the HSV color space yielded better results during image retrieval than color histograms in other color spaces. The RGB space is uniformly sampled into $16 \times 16 \times 16$ bins which are represented by the center color and transformed to the HSV color space. The $4,096$ colors in the HSV space are then clustered into 64 colors using a K-means clustering algorithm [108]. A look-up table is generated to map every RGB value to a bin based on the cluster membership. We extract 5 local color histograms for every image (top left quarter, top right quarter, bottom left quarter, bottom right quarter, and central quarter) to incorporate spatial information (320 features per image). A color similarity matrix, $A(i, j)$, (generated from the distances between the 64 color centers) is used to smooth

the histograms as follows:

$$H^{'}(i) = \sum_{j=1}^{64} H(j) * A(i,j),$$

where H is the original histogram and $H^{'}$ is the smoothed histogram. The smoothing spreads the value of a bin into bins of similar color.

4.3.2 Color Coherence Vector

A color coherence vector is a color histogram refinement scheme that divides each bin into coherent and non-coherent pixels [51]. A pixel in a bin is said to be coherent if it is part of a large similarly-colored region. An 8-neighbor connected component analysis is used to extract connected regions of the same color. Pixels in regions whose size exceeds a threshold (1% of image size) are counted as coherent pixels, and those from smaller regions count towards non-coherent regions. We extended the color coherency concept to our local histograms in the quantized HSV space leading to a 640-dimensional feature vector. Figures 4.7(a)-(h) show a city and landscape image and their color histogram and color coherence vector features, respectively. Long distance landscape (mountains, sunset/sunrise, etc) shots tend to have more coherent pixels (large regions of similar color, see Figures 4.7 (f) and (h)), whereas city scenes and some landscape scenes (forest images) tend to have an equal fraction of coherent and non-coherent pixels (see Figures 4.7 (e) and (g)).

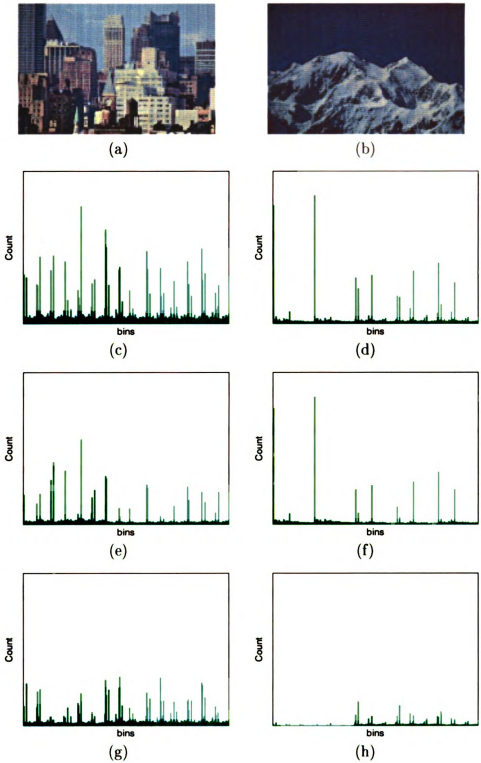


Figure 4.7: Color-based features for (a) city and (b) landscape image; (c) and (d) show the color histogram features for (a) and (b); (e) and (f) show the coherent color bins for (a) and (b); (g) and (h) show the non-coherent color bins for (a) and (b).

4.3.3 Spatial Color Moment Feature Vector

First- and second-order moments in the *LUV* color space were used as color features (it was pointed out in [52] that moments in the *LUV* color space yielded better results during image retrieval than moments in other color spaces). The image was divided into 10^2 sub-blocks and six features (3 each for mean and standard deviation) were extracted from each sub-block (600-dimensional feature vector). The features thus capture the low frequency spatial color information in the image. Figures 4.8(a)-(d) show a typical indoor and outdoor image and their spatial color moment features in the *LUV* color space. Indoor images tend to have more uniform illumination because they are close-up shots. Outdoor images, on the other hand, tend to have more varied illumination and chrominance changes. These effects are captured by the spatial color moments, with more variation in the values for typical outdoor images. Note that each feature vector is composed of the mean and the variance for the L, U, and V components of every block.

4.3.4 Edge Direction Histograms

We define texture in an image in terms of an edge direction histogram [114]. The Canny edge detector is used to extract the edges in an image. We have modified the edge direction histograms defined in [114] by adding an extra bin to measure the frequency of non-edge pixels in the image. A total of 73 bins are used to represent the edge direction histogram; the first 72 bins are used to represent edge directions quantized at 5° intervals and the last bin represents a count of the number of pixels that

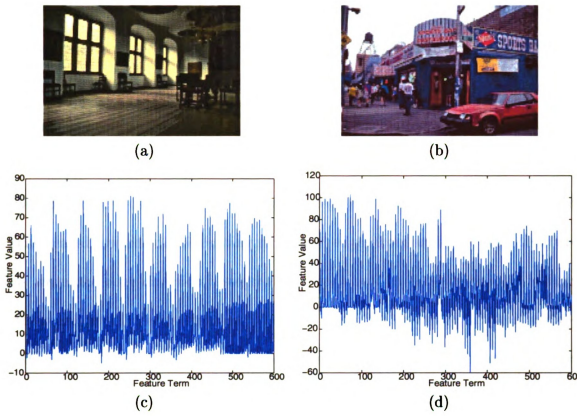


Figure 4.8: Spatial color moment features for a typical (a) indoor and (b) outdoor image; (c) and (d) show the spatial color moment features, where the x-axis represents the feature terms and the y-axis represents the feature values, in the *LUV* color space over 10×10 sub-blocks of an image.

didn't contribute to an edge. To compensate for different image sizes, we normalize the histograms as follows:

$$H(i) = H(i)/n_e, \quad i \in [0, \ldots, 71]; \qquad H(72) = H(72)/n_p,$$

where $H(i)$ is the count in bin i of the edge direction histogram, n_e is the total number of edge points detected in the image, and n_p is the total number of pixels in the image.

4.3.5 Edge Direction Coherence Vector

The coherency concept of color histograms [51] was extended to edge direction histograms in [19]. An edge direction coherence vector stores the number of coherent versus non-coherent edge pixels with the same directions (within a quantization of 5°). A threshold (0.1% of image size) on the size of every connected component of edges in a given direction is used to decide whether the region is coherent or not. This feature is thus geared towards discriminating structured edges from arbitrary edge distributions when the edge direction histograms are matched. A 145-dimensional feature vector represents the edge direction coherence vector for an image. Figures 4.9(a)-(h) show a city and landscape image and their edge direction histograms and edge direction coherence vector features, respectively. As can be seen in Figures 4.9 (e) and (f), the city images tend to produce coherent edge pixels in the vertical direction, where as most edges in landscape images are non-coherent. The coherent edge pixels in the vertical direction in the city images (and the lack of these in landscape images) increase the discriminating power of the edge direction coherent vectors over the edge direction histograms for the two classes under consideration.

4.3.6 DCT Coefficients

Color and texture features are also described in terms of the Discrete Cosine Transform (DCT) coefficients of an image. Since every image in the database is represented in the JPEG compressed format, extracting DCT features is relatively efficient. Central moments of second- and third-order [115, 100] of the DCT coefficients were used

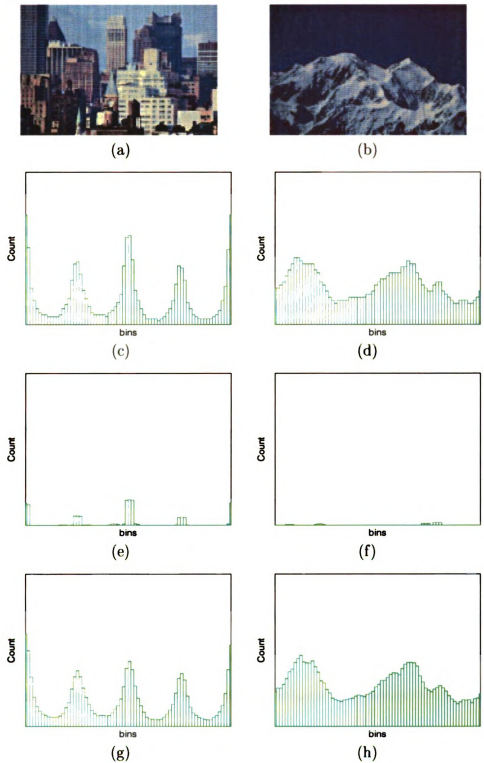


Figure 4.9: Edge direction-based features for (a) city and (b) landscape image; (c) and (d) show the edge direction histogram features for (a) and (b); (e) and (f) show the coherent edge direction bins for (a) and (b); (g) and (h) show the non-coherent edge direction bins for (a) and (b).

to represent an image. In particular, 4 moments for each of the 17 DCT coefficients (first 9 coefficients in the intensity field and the first 4 DCT coefficients in the two chrominance fields) were used as the features. The resulting 68-dimensional feature vectors were then normalized to zero mean and unit variance. The DCT coefficients store the coded version of *LUV* color features in 8×8 pixel blocks. Hence, these features are similar to the spatial color moment features. The difference is that the DCT features capture more of the global distribution (moments of DCT coefficients over the entire image) rather than local sub-block distributions captured by the spatial color moment features described above (spatial color moments in image sub-blocks).

4.3.7 MSAR Texture Features

The multiresolution simultaneous autoregressive model for texture features (MSAR [70]) have been shown to be among the best texture features on the Brodatz album [11]. The model constructs the best linear predictor of a pixel based on a non-causal neighborhood. The features used to describe various textures are the weights associated with the predictor. Szummer et al. [24] used three different neighborhoods at scales of 2, 3, and 4 to yield a 15-dimensional feature vector. As in [24], every image was divided into 4×4 sub-blocks and MSAR features were extracted from each sub-block. Figures 4.10(a)-(d) show two images and their corresponding sub-block MSAR texture features. Homogeneous image regions yield texture features with very low values.

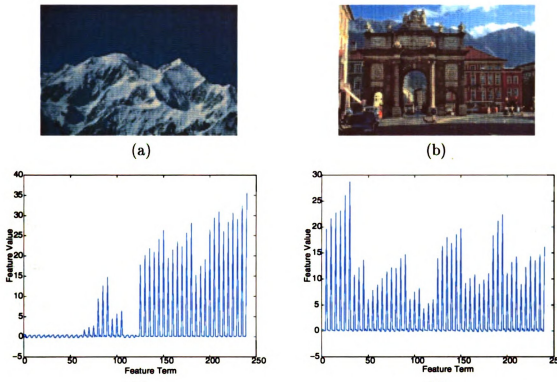


Figure 4.10: MSAR texture features; (a) and (b) show two database images; (c) and (d) show the sub-block MSAR texture features; homogeneous image regions yield very low values of texture feature.

4.3.8 Feature Saliency

The accuracy of a classifier depends on the underlying representation of the images. The more discriminative the features, better is the classification accuracy. Therefore, we need to develop robust schemes to identify salient image features that have sufficient discriminative power for each of the above classification problems.

A given feature is said to have a large discrimination power if its intra-class (within-class) variance is small and the inter-class (between-class) variance is large. We thus empirically constructed distributions for the intra- and inter-class distances for the given set of features for each classification problem. We present the results on

city vs. landscape image classification problem using database $D2$. The intra-class distribution was generated by computing the distances between every pair of images in the same class. The inter-class distribution was generated by computing the distance of every image in the city class to every image in the landscape class. Based on the overlap of the intra- and inter-class distance distributions, we can identify the more discriminative features for the given classification problem.

The Euclidean distance metric was used to generate the intra-class and the inter-class distance distributions for the database images. Figures 4.11(a)-(d) show the intra- and inter-class distributions for four different feature sets, namely, color histogram, DCT moment, edge direction histogram, and edge direction coherence vector features. The color coherence vectors and spatial color moments yield similar inter-class and intra-class distributions as the color histograms. The large overlap between the two distributions based on color histograms and DCT coefficients shows that these features are not suited for the city vs. landscape classification problem. On the other hand, the distributions based on the edge direction coherence vectors have a substantially smaller overlap (especially, at lower distance values). These results suggest that the use of a small set of near neighbors based on edge direction features is better in discriminating between the two classes than the color histograms and DCT coefficient features. In order to verify the claim, we used Sammon's non-linear projection algorithm to plot the high-dimensional patterns in two dimensions. As can be seen in Figures 4.12(a) and (b), the edge direction coherence vectors do a better job in discriminating between the given two classes (edge direction coherence vectors yield a piecewise linear boundary).

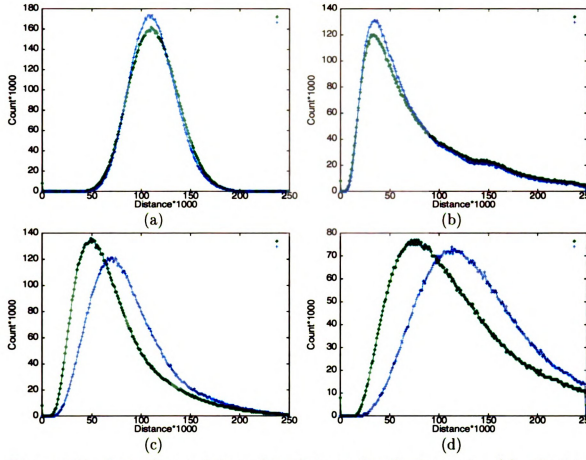
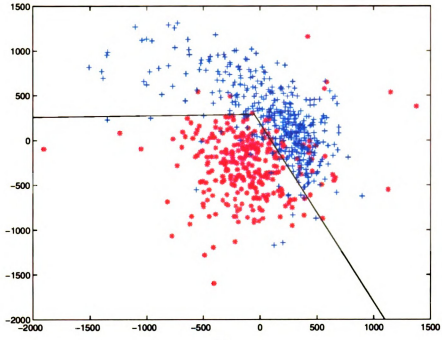
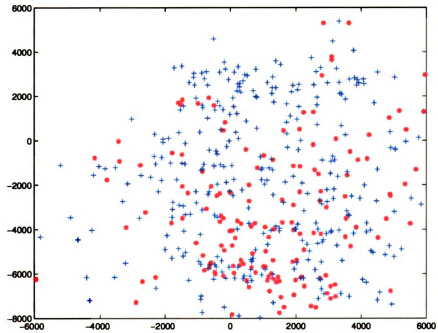


Figure 4.11: Intra-class and inter-class distance distributions using (a) color histograms; (b) DCT moments; (c) edge direction histograms; (d) edge direction coherence vectors; solid-line represents the intra-class distance, while the dotted-line represents the inter-class distance; x-axis represents inter-image distance; y-axis represents the frequency.

Man-made objects in the city images usually have strong vertical and horizontal edges, whereas non-city scenes tend to have edges randomly distributed in various directions. A feature based on the distribution of edge directions can discriminate between the two categories of images. On the other hand, color features would not have sufficient discriminatory power as man-made objects have arbitrary color distributions (two buildings need not have the same color). We feel that color features may be better suited for further classification of landscapes (natural scenes) where colors are relatively constant (grass has a yellowish-green hue, sky has a blue hue, etc).



(a)



(b)

Figure 4.12: 2-D projections of (a) edge direction coherence vector features and (b) color coherence vector features; $*$ represents the landscape patterns and $+$ represents the city patterns; only a subset of $2,716$ patterns has been plotted here for clarity of display.

We use a similar approach to identify appropriate features for the other classifiers. For example, outdoor images tend to have uniformity in spatial color distributions, where the sky is on top and is typically blue in color. Indoor images tend to have more varied color distributions and have more uniform lighting (most are close up shots). Thus, it seems logical to use spatial color distribution as a feature for discriminating between indoor and outdoor images. On the other hand, shape and texture features may not be useful, since objects with similar shapes and textures (people, furniture, plants, edges due to walls, etc.) can be present in both indoor and outdoor images. Therefore, we use spatial color information features that represent these qualitative attributes of indoor and outdoor classes.

In the case of classification of landscape images into sub-categories, such as sunset, forest, and mountain, global color distributions seem to adequately describe these sub-classes. Sunset pictures typically have highly saturated colors (mostly yellow and red); mountain images tend to have a sky in the background (typically blue); and forest images tend to have more greenish distributions (presence of dense foliage). Based on the above observations, we use color features (histograms and coherence vectors) in the HSV color space for further classification of landscape images [19]. Table 4.1 briefly describes the qualitative attributes of the various classes and the features used to represent them.

Table 4.1: Qualitative attributes of classification problems and the associated low-level features.

Classification Problem	Qualitative Attributes	Low-level Features
Indoor vs. Outdoor	spatial color and intensity distributions	10×10 sub-block color moments in LUV space
City vs. Landscape	distribution of edges	edge direction histograms and coherence vectors
Sunset vs. Forest vs. Mountain	global color distributions and saturation values	color histograms and coherence vectors in HSV space

4.4 Vector Quantization

A number of experiments were conducted to study the robustness and limitations of the VQ-based Bayes classifier for the classification problems mentioned in Table 4.1. For every pattern class, half of the database images were used to train the VQ for each of the image features. The MMDL principle described in Section 3.4.3 was used to determine the codebook size from the training samples for the various classifiers. We present the results of applying the MMDL principle to the indoor vs. outdoor classifier for the spatial color moment features and the city vs. landscape classifier based on the edge direction coherence vector features.

Figures 4.13(a)-(c) show the plots of the criterion function $\{L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) + L(\boldsymbol{\theta}_{(q)})\}$ which needs to be minimized (Eq. (3.9)) vs. the codebook size, q, for the spatial color moment features for (a) indoor, (b) outdoor, and (c) both the classes. As can be seen in the figures, $q = 10$ minimizes the criterion in Eq. (3.9) for the indoor class, while $q = 15$ minimizes the criterion for the outdoor class. Combining the two classes yields $q = 30$ as the optimal number of codebooks for the indoor vs. outdoor classifier based

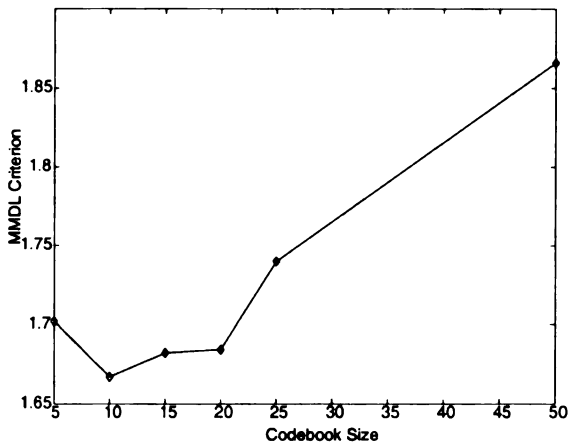
on the training samples. Using the results, 15 codebook vectors were extracted for both indoor and outdoor classes. Similar plots for the edge direction coherence vector are shown in Figures 4.14(a)-(c). Note that now, $q = 15$ minimizes the criterion in Eq. (3.9) for the city class, whereas $q = 20$ is optimal for the landscape class. For the combination of the two classes, $q = 40$ is the optimal size of the codebook based on the training samples. Based on the above analysis, 40 codebook vectors (20 per class) were selected for the city vs. landscape image classifier. For a further classification of landscape images, a codebook of 5 vectors was selected for each class. The codebook vectors for each class were then stored as prototypes for the class. Table 4.2 shows the number and dimensionality of the codebook vectors for the various classification problems.

Table 4.2: Codebook vectors used for the various classifiers.

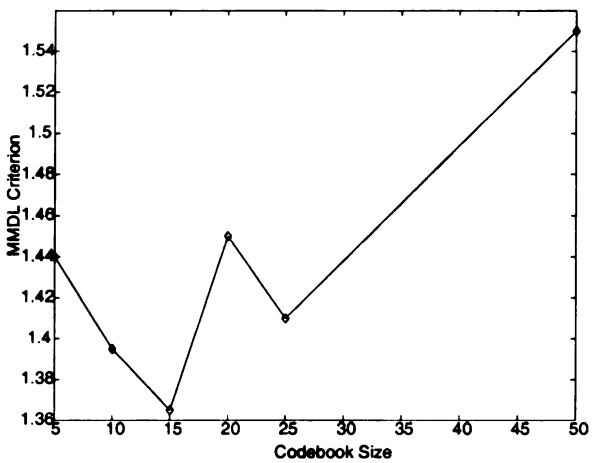
Classification Problem	# of Codebook Vectors / Class	Feature Dimensionality
Indoor/Outdoor	15	600
City/Landscape	20	145
Sunset/Forest/Mountain	5	640

4.5 Experimental Results

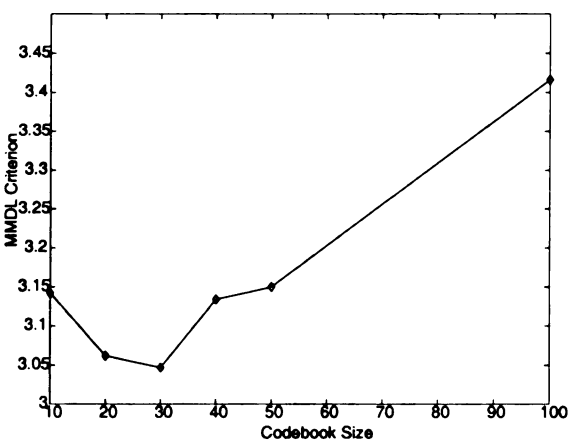
We present classification accuracies on a set of independent test patterns (hold-out error) as well as on the training patterns (re-substitution error).



(a)

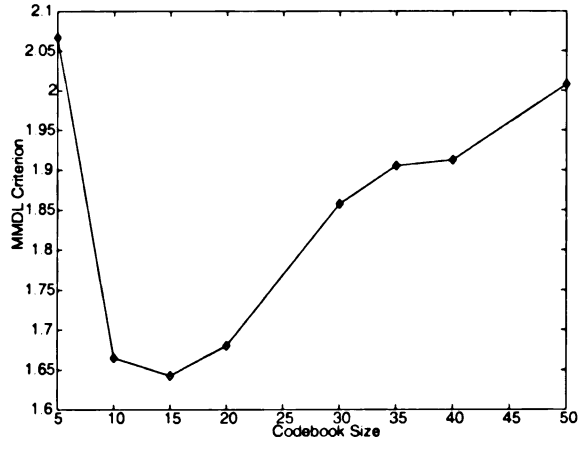


(b)

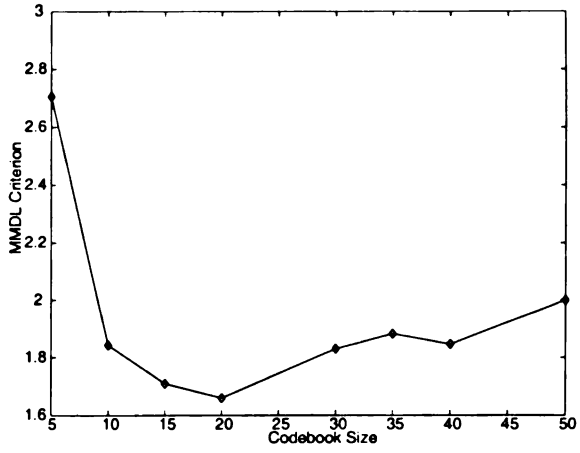


(c)

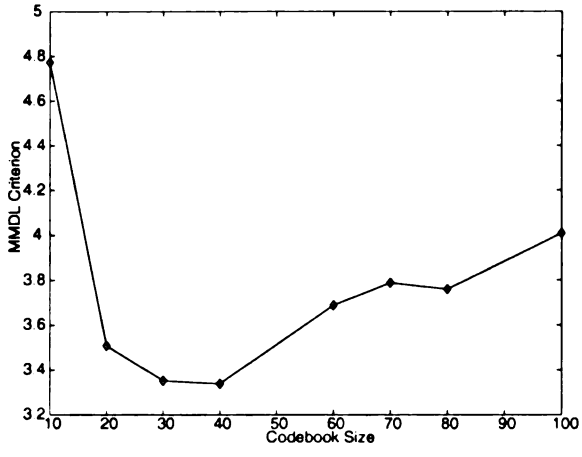
Figure 4.13: Determining the codebook size for spatial color moment features for the indoor vs. outdoor classification problem; (a) indoor class; (b) outdoor class; (c) indoor and outdoor classes combined.



(a)



(b)



(c)

Figure 4.14: Determining the codebook size for edge direction coherence vectors; (a) city class; (b) landscape class; (c) city and landscape classes combined.

4.5.1 Indoor Vs. Outdoor Classification

Database D1 was used to train the indoor vs. outdoor classifier. This database consisted of 2,470 indoor and 2,611 outdoor images. Apart from the color moment features, we also used the sub-block MSAR texture features [24], the edge direction features, DCT features, and the color histogram features for the classification. MSAR features yielded an accuracy of around 75% on the test set; edge direction histogram, DCT, and color histogram features yielded an accuracy of around 60%; and the color moment features yielded a much higher accuracy of around 90%. These results show that the spatial color distribution (which captures intensity/illumination changes) is more suited for indoor vs. outdoor classification. A combination of color and texture features did not yield a better accuracy than the color moment features alone. Table 4.3 shows the classification results for the color moment features for the indoor vs. outdoor classification problem. The classifier performed with an accuracy of 94.2% and 88.2% on the training set and an independent test set (Test1 in Table 4.3), respectively. On a different test set (Test2 in Table 4.3) containing 1,850 images from database D2, the classifier performed with an accuracy of 89.5%. The classification accuracy of 90.5% was obtained on the entire database of 6,931 images. Szummer et al. [24] use a K-Nearest Neighbor classifier and leave-one-out criterion to report classification accuracies, for the indoor vs. outdoor classification problem, of approximately 90% on a database containing 1,324 images. Thus, our classifier's performance is comparable to those reported in the literature. A major advantage of the Bayesian classifier is its efficiency due to the small number of codebook vectors

needed to represent the training data.

Figures 4.15 and 4.16 show a representative subset of the misclassified indoor and outdoor images. Presence of bright spots either from some light source or from sunshine through windows and doors seems to be a main cause of misclassification of indoor images. The main reasons for the misclassification of outdoor images are as follows: (i) uniform lighting on the image mostly as a result of a close-up shot and (ii) low contrast images (some of the indoor images used in the training set were low contrast digital images and hence, most low contrast outdoor images were classified as indoor scenes). The above experimental results show that spatial color distribution captured in the sub-block color moment features has sufficient discrimination power for the indoor vs. outdoor classification problem. Moreover, since the training and test set error rates are not very different, the classifier has not been overly trained.

Table 4.3: Classification accuracies (in %) for indoor vs. outdoor classification problem using color moment features; Test1 and Test2 are two independent test sets.

Test Data	Database Size	Accuracy (%)
Training Set	2,541	94.2
Test1	2,540	88.2
Test2	1,850	89.5
Entire Database	6,931	90.5

4.5.2 City Vs. Landscape Classification

We now address the city vs. landscape classification problem and a further classification of landscape images into sunset, forest, and mountain classes using the Bayesian framework. Table 4.4 shows the classification results for the city vs. landscape classifi-

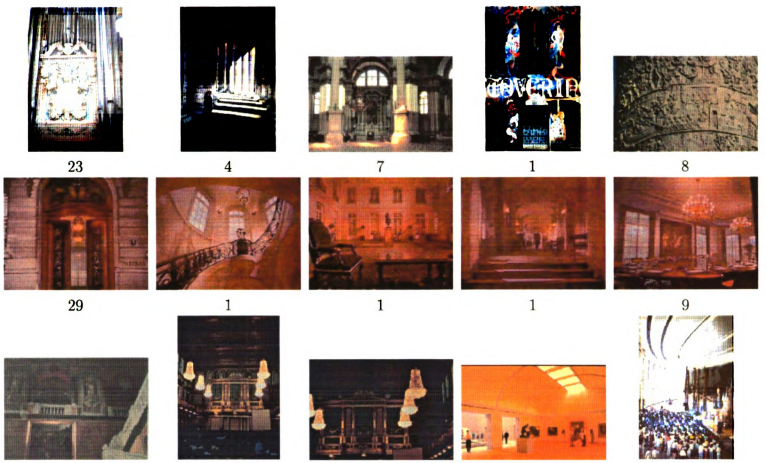


Figure 4.15: A subset of the indoor images that were misclassified using the color moment features; the corresponding confidence values (in %) associated with the true class are presented.

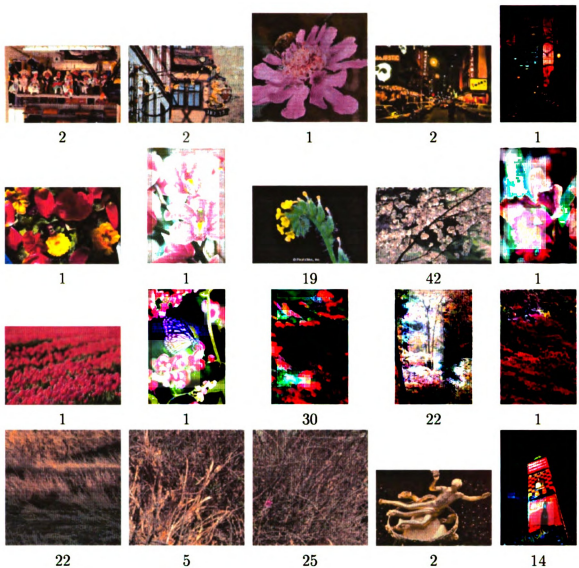


Figure 4.16: A subset of the outdoor images that were misclassified using the color moment features; the corresponding confidence values (in %) associated with the true class are presented.

cation problem using database D2. Edge direction coherence vector provides the best accuracy of 97.0% for the training data and 92.9% for the test data. A total of 126 images were misclassified (a classification accuracy of 95.3%) when the edge direction coherence vector was combined with the color histogram feature vector. Figures 4.17 and 4.18 show representative subsets of the misclassified city and landscape images, respectively. Most of the misclassifications for city images could be attributed to the following reasons: (i) long distance city shots at night (which made it difficult to extract the edges), (ii) top view of city images (lack of vertical edges in the images), (iii) highly textured buildings, and (iv) trees obstructing the buildings. Most of the misclassified landscape images had strong vertical edges from tree trunks, close-ups of stems, fences, etc., that led to their assignment to the city class.

Table 4.4: Classification accuracies (in %) for the city vs. landscape classification problem; the features are abbreviated as follows: edge direction histogram (EDH), edge direction coherence vector (EDCV), color histogram (CH), and color coherence vector (CCV); accuracies in **bold** represent the best classification accuracies using individual features and a combination of features on the entire database.

Test Data	EDH	EDCV	CH	CCV	EDH & CH	EDH & CCV	EDCV & CH	EDCV & CCV
Training Set	94.7	97.0	83.7	83.5	94.8	95.4	96.4	96.9
Test Set	92.0	92.9	75.4	76.0	92.5	92.8	93.4	93.8
Entire Database	93.4	**95.0**	79.6	79.8	93.7	94.1	94.9	**95.3**

We also evaluated the classification accuracy using the edge direction coherence vector on an independent test set of 568 outdoor images from database D1. A total of 1,177 images of the 4,181 outdoor images in database D1 contained close ups of human faces. We removed these images for city vs. landscape classification. Recent advances in face detection algorithms show that faces can be detected rather reli-

ably [8]. Of the remaining images, we extracted 568 test images that were not part of database D2. The edge direction features yielded an accuracy of 90.0% with 57 misclassifications out of the 568 images. Combining color histogram features with the edge direction coherence vector features reduced the misclassification in the above experiment to 56, again showing that edge direction features have enough discriminative power for the city vs. landscape classification problem.

4.5.3 Further Classification of Landscape Images

A subset of 528 landscape images from database D2 was classified into the sunset, mountain, and forest classes. While our limited experiments on human subjects [19] revealed classes such as sunset/sunrise, forest and farmland, mountain, pathway, water scene, etc., these classes were not consistent among the subjects in terms of the actual labeling of the images. We found it extremely difficult to generate a semantic partitioning of landscape images. We thus restricted classification of landscape images into three classes that could be more unambiguously distinguished, namely, sunset, forest, and mountain classes. Of these 528 images, a human subject labeled 177, 196, and 155 images as belonging to the forest, mountain, and sunset classes, respectively. A 2-stage classifier was constructed. First, we classify an image into either sunset or the forest & mountain class. The above hierarchy was based on the categories identified by the human subjects as shown in Figure 4.6(a), where the sunset cluster seemed to be more compact and separated from the other categories in the landscape class. We next address the forest vs. mountain classification prob-

Figure 4.17: A subset of the misclassified city images; the corresponding confidence values (in %) associated with the true class using a combination of edge direction coherence vector and color histogram features.

Figure 4.18: A subset of the misclassified landscape images; the corresponding confidence values (in %) associated with the true class using a combination of edge direction coherence vector and color histogram features.

lem. Table 4.5 shows the results for the classification of landscape images into sunset vs. forest & mountain classes. The color coherence vector provides the best accuracy of 99.2% for the training data and 93.9% for the test data. Color features do much better than the edge direction features here, since color distributions remain more or less constant for natural images (blue sky, green grass, trees, plants, etc). A total of 18 images were misclassified (a classification accuracy of 96.6%) when the color coherence vector feature was used. We find that combining feature vectors does not improve the classification accuracy. This shows that color coherence vector has sufficient discrimination ability for the given classification problem.

Table 4.5: Classification accuracies (in %) for the sunset vs. forest & mountain classification; accuracies in **bold** represent the best classification accuracies using individual features and a combination of features on the entire database; SPM represents the spatial color moment features.

Test Data	EDH	EDCV	CH	CCV	SPM	EDH & CH	EDH & CCV	EDCV & CH	EDCV & CCV
Training Set	88.3	88.3	96.2	99.2	98.9	95.9	96.6	95.5	97.0
Test Set	86.3	89.0	89.7	93.9	93.9	90.1	95.4	90.5	95.1
Entire Database	87.4	88.7	93.0	**96.6**	96.4	93.0	96.0	93.0	**96.1**

Table 4.6 shows the classification results for the individual features for forest vs. mountain classes (373 images in the database). Spatial color moment features provide the best accuracy of 98.4% for the training data and 93.6% for the test data. A total of 15 images were misclassified (a classification accuracy of 96.0%) when the spatial color moment features were used. Again, the combinations of features did not perform better than the color features, showing that these features are quite adequate for this classification problem. Note that the spatial color moment features and the

color coherence vector features yield similar accuracies for the classification of landscape images. However, the database of 528 images is very small to identify the best color feature for the classification of landscape images. Using color coherence vector features adds to the complexity of the classifiers (feature extraction and additional storage).

Table 4.6: Classification accuracies (in %) for the forest vs. mountain classification; accuracies in **bold** represent the best classification accuracies using individual features and a combination of features on the entire database.

Test Data	EDH	EDCV	CH	CCV	SPM	EDH & CH	EDH & CCV	EDCV & CH	EDCV & CCV
Training Set	83.4	78.1	92.0	98.9	98.9	94.1	98.4	93.6	98.4
Test Set	87.1	77.2	91.4	91.9	93.6	93.0	92.5	93.5	91.9
Entire Database	85.3	77.7	91.7	95.5	**96.0**	93.6	**95.5**	93.6	95.2

4.6 Discussion

Content-based indexing and retrieval has emerged as an important area in computer vision and multimedia computing. User queries are typically based on semantics and not on low-level image features. Providing semantic indices into large databases is a challenging problem. Psychological and psychophysical studies have shown that scene identification in humans can occur, under certain conditions, without any kind of object recognition form global low-level features. We have experimentally shown that certain semantic categories can be learnt using specific low-level image features under the constraint that the test images do belong to one of the pattern classes under consideration. Specifically, we have developed a hierarchical Bayes classifier

for classifying images. At the top level, images are classified into indoor and outdoor categories. The outdoor images are then classified into city and landscape classes (we assume a face detector is available to separates close-up images of people in outdoor images into the "other" category) and finally, a subset of landscape images are classified into the sunset, forest, and mountain class. Classifications based on local color moments, color histograms, color coherence vectors, edge direction histograms, and edge direction coherence vectors as features show promising results.

Chapter 5

Automatic Detection of Image Orientation

All image management systems require information about the true image *orientation.* When a user scans a picture, she expects its image to be displayed in its correct orientation, regardless of the orientation in which the photograph was placed on the scanner. Moreover, people usually take photographs at a number of orientations, but they expect the resulting images from scanned negatives to be displayed and printed in their correct orientations. Thus, an image management system is expected to correctly orient the input images. Currently, the above operation of orientation detection is performed manually.

5.1 Problem Definition

We define the image orientation detection problem as follows: "Given an image, determine its *correct* orientation". The correct orientation of an image is defined as the orientation in which the scene (captured by the image) originally occurred. Due to the rotation of the camera while taking a picture or incorrect placement of the picture on a scanner, a digital image can be mis-oriented. We assume that the input image is restricted to only four possible rotations that are multiples of 90°, i.e., a photograph which is scanned can differ from its correct orientation by 0° (no rotation), 90°, 180°, or 270°. This is reasonable since photographs placed on a scanner are usually aligned with horizontal or vertical boundaries of the scanner plate. Thus, the orientation detection problem can be represented as a four-class classification problem with ω_1, ω_2, ω_3, and ω_4 representing the four possible orientations of 0°, 90°, 180°, and 270°, respectively. Note that an image in an arbitrary orientation can easily be rotated into one of the above four orientations by aligning the image boundaries horizontally and vertically. Figure 5.1(a) shows an image in four possible orientations and Figure 5.1(b) shows the true orientation detected by our algorithm for the four input images in Figure 5.1(a).

Automatic image orientation detection is a very difficult problem. Humans use object recognition and contextual information to identify the correct orientation of an image. Since information regarding the presence of semantic objects such as sky, grass, house, people, furniture, etc. and their inter-relationships cannot be reliably extracted from general images, we rely on low-level visual features (e.g., spatial color

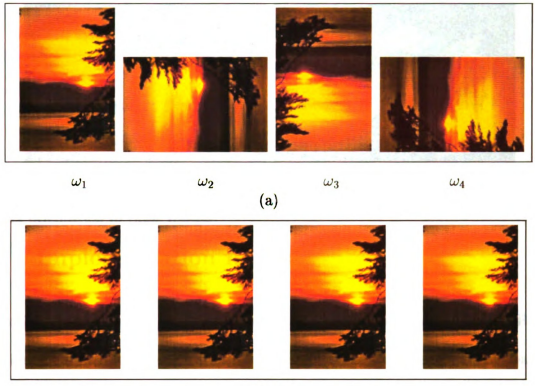


Figure 5.1: Orientation detection: (a) four possible orientations of interest; (b) correct orientations detected by our algorithm.

distributions, texture, etc.) for orientation detection. Figures 5.2(a) and (b) illustrate the difficulty in image orientation estimation, where the true orientation cannot be detected unless you first recognize the object present in the image (a seal and knowledge that head of the seal should be in the top portion of the image). Close-up images, low contrast images, or images of uniform or homogeneous texture (e.g., sunset/sunrise and indoor images) pose additional problems for robust orientation estimation.

(a) (b)

Figure 5.2: Object detection and contextual knowledge are required to detect the correct orientation in these images.

5.2 Implementation Issues

The orientation detection problem formulated above is again addressed using Bayes decision theory as described in Chapter 3. Each image is represented by a feature vector extracted from the image. The probabilistic models required for the Bayesian approach are estimated during the training phase, wherein, the class-conditional probability density functions of the observed feature vector are estimated under a Vector Quantization (VQ) framework.

Feature extraction is an important issue in classifier design. Since, global image features are not invariant to image rotation, we use local regional features for the orientation classification. An image is represented in terms of N^2 blocks ($N = 10$) and the features are extracted from these local regions. A number of features (e.g., color moments in the LUV color space [28]; color histograms in the HSV color space [19]; edge direction histograms [19]; and MSAR texture features [70], see also Section 4.3) were evaluated for their ability to discriminate between the four possible orientations of an image. Feature saliency was evaluated using a K-nearest neighbor classifier.

Preliminary results showed that spatial color moment features yielded a much higher accuracy for the four-class classification problem than the color histogram, edge direction histogram, and MSAR features. Certain image classes such as outdoor images, tend to have uniform spatial color distributions (sky is on top and typically blue in color, grass is typically green and towards the bottom of an image, etc.), while texture features have smaller variations with changes in image orientation. We thus, trained our Bayesian classifier using spatial color moment features.

The number of sub-blocks, N^2, was selected using a K-nearest neighbor classifier, on a database of 4,000 images, by varying N over a small number of values ($N = 3$, $N = 5$, and $N = 10$) (see Table 5.1). The experiments show that as the number of regions (N^2) was increased, the accuracies improved with the best accuracy of about 82% obtained with $N = 10$. We feel that as the number of regions is increased, the color moment features are better able to describe local regions. However, this trend would not follow for much larger values of N, since the local regions would become too small and the second-order color moments would not capture the variation in the local regions. Moreover, with increasing N, the complexity of the classifier increases. Hence, we used 600 spatial color moment features in 10^2 regions (3 mean and 3 variance values of L, U, and V components per region) for the orientation detection problem.

Table 5.1: Classification accuracies of a K-NN classifier using leave-one-out-method for the four-class orientation detection problem; accuracies are presented for the spatial color moment features under varying number of regions (N^2) per image.

Database Size	N	Acc (%)
4,000	3	77.1
4,000	5	77.6
4,000	10	81.8

5.3 Experimental Results

We have tested our orientation detection system on two independent databases of 16,392 ($D1$) and 1,509 ($D2$) images, respectively. Database $D1$ consists of professional quality images from Corel stock photo library. Database $D2$ consists of photographs (an amateur's personal collection) taken using a digital camera (mostly indoor images in poor lighting conditions). These images were used in experiments to determine scaling issues, i.e., robustness of a classifier trained and tested on different datasets. Table 5.2 shows the classification results using the LVQ-base Bayes classifier for the four-class orientation detection problem. When the classifier was tested on the training data, it yielded accuracies of over 96%. On an independent test data, however, the accuracy drops to approximately 87%. Training the classifier on images from database $D1$ and testing on images from database $D2$ yielded a rather poor classification accuracy of 77.3%. This is due to the fact that the images in database $D1$ have high contrast (professional quality images), whereas those in database $D2$ are captured under non-optimal conditions. On the other hand, when the classifier is trained on $D1 \cup D2$, it yielded classification accuracies of over 96% on the training data and over 87% on independent test data. Figure 5.3 shows a subset of the images

whose orientation was detected correctly. Since we do not pose any restrictions on the input images, nor do we use any contextual information (about image content), the above classification results are reasonable. Further, when both the databases are combined and used for training, the classifier reported an accuracy of over 96% (resubstitution case) on the training data (over 17,900 images). This shows that the classifier has a further capacity to learn.

Table 5.2: Experimental results for orientation detection on different databases.

Training Set (Database:Size)	Independent Test Set (Database:Size)	Accuracy (%)	Number of Misclassifications
$D1$: 8,000	$D1$: 8,392	87.2	1,071
$D2$: 755	$D2$: 754	87.3	97
$D1$: 16,392	$D2$: 1,509	77.3	342
$D1$+$D2$: 8,755	$D1$+$D2$: 9,146	87.7	1,123

Most of the errors were made on images with uniform or homogeneous texture (as in close-ups of buildings, faces, trees, plants, and animals, or indoor or underwater pictures, etc.), low contrast images (either due to poor lighting or sunset/sunrise images), and images of top-views (lack of discriminating illumination information across the image). Figure 5.4 shows a subset of the misclassified images.

5.4 Selecting Codebook Size

The above experiments were based on an empirical choice of codebook size. We next demonstrate how the MMDL principle described in Section 3.4.3 can be used to select the optimal codebook size for the orientation detection problem. The training set consisted of 8,755 images extracted randomly from both the databases (database

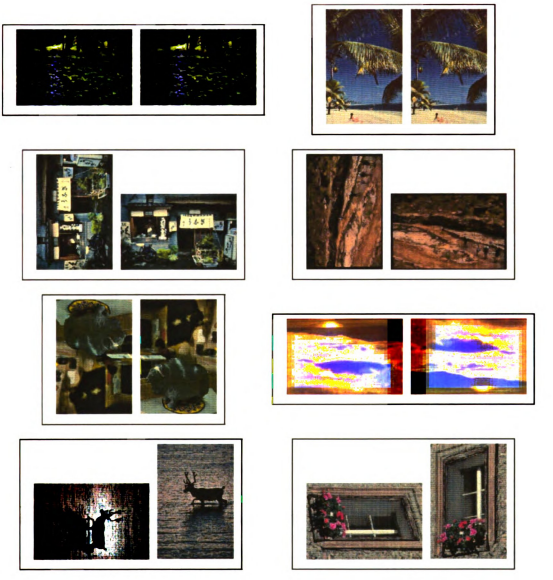
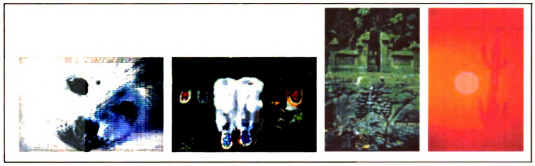
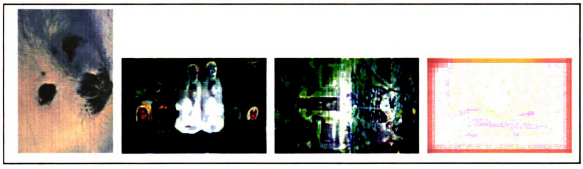


Figure 5.3: A subset of database images whose orientations were detected correctly; the first image in each block is the input and the second image in the block is the image with the correct orientation as determined by our algorithm.



(a)



(b)

Figure 5.4: A subset of images for which our orientation detection system fails: (a) input images; (b) detected orientations.

used in the last experiment in Table 5.2). Figures 5.5(a)-(e) show the plots of the criterion function $\{L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) + L(\boldsymbol{\theta}_{(q)})\}$ which needs to be minimized (Eq. (3.9)) vs. the codebook size, q, for each of the four classes (Figures 5.5(a)-(d)) and for the combination of the four classes (Figure 5.5(e)). Based on these plots, $q = 75$ codebook vectors were selected for each of the four classes. The new classifier yielded accuracies of 95% (438 misclassifications out of $8,755$ images) and 87.8% ($1,112$ misclassifications out of $9,146$ images) on the training set and the independent test set, respectively.

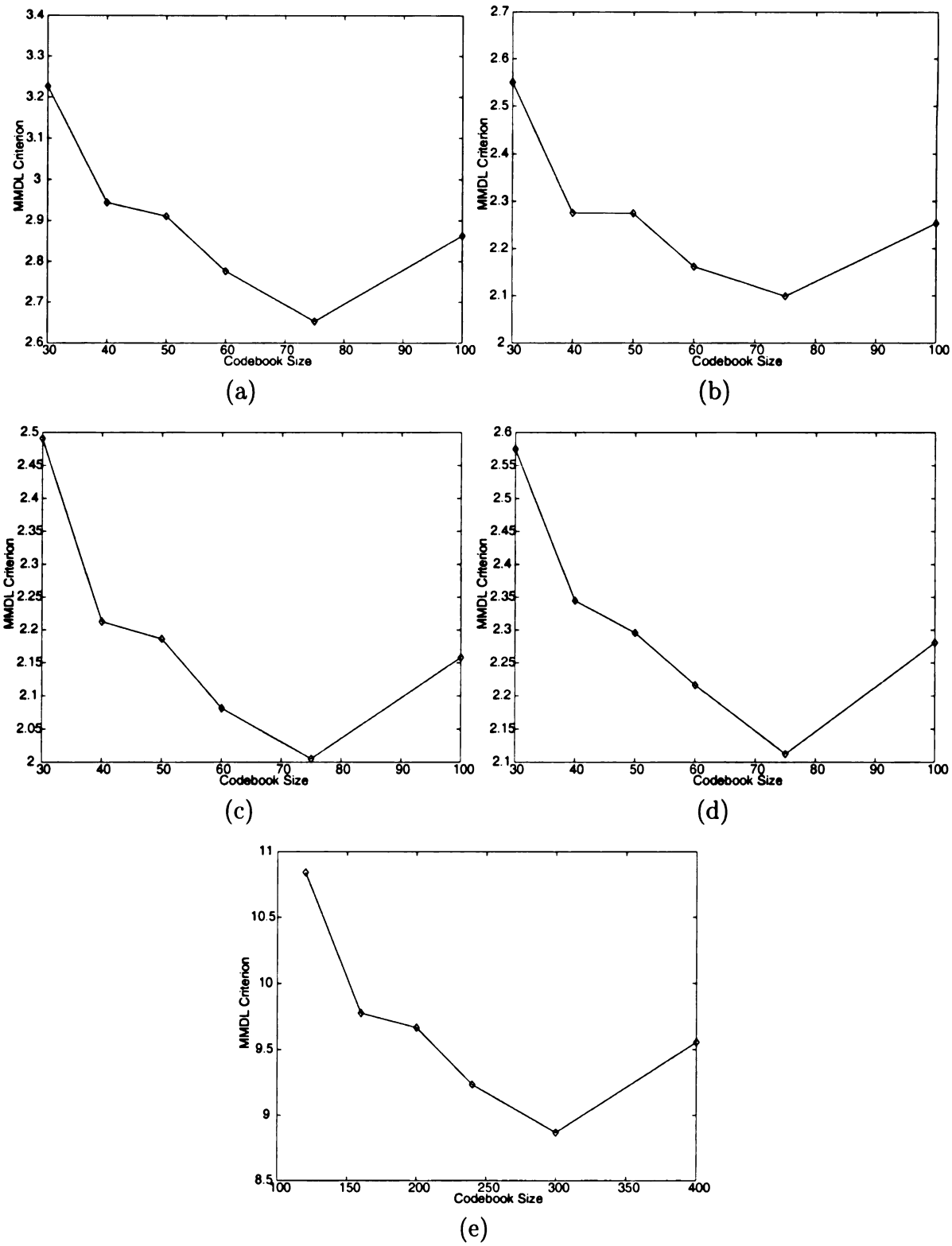


Figure 5.5: Determining the codebook size for spatial color moment features for the orientation detection problem; (a) class ω_1 (0°); (b) class ω_2 (90°); (c) class ω_3 (180°); (d) class ω_4 (270°); (e) all classes combined.

5.5 Discussion

Automatic image orientation detection in the absence of any contextual information is a very difficult problem. Using a Bayesian classifier, we have presented a *novel* solution to the above problem. We show how the spatial color moment features can capture information about the image orientation for most of the images in our database. The classifier probably captures the difference in illumination and color across an image (illumination is usually from the top) to make a decision. This premise is strengthened by the fact that most misclassifications are on images with low-contrast and images representing a top view of a scene. A limitation of the above Bayesian paradigm is its dependence on the training samples and the low-level features used to represent the images. In the next Chapter, we address the issue of improving the robustness of the semantic classifiers.

Chapter 6

Designing Robust Classifiers

We have shown in the previous chapters how the problem of semantic image classification can be addressed using a Bayesian classification framework from global low-level features. In this chapter, we propose schemes to make the classifiers more robust. In particular, we address the following problems:

- Feature Selection: We investigate how feature selection methods can improve the classification accuracy (Section 6.1).

- Incremental Learning: We propose a scheme to *incrementally* train the classifiers to incorporate additional training samples as they become available (Section 6.2).

- Reject Option: We develop a method to locally adapt the thresholds for rejecting images for which the corresponding image features do not have sufficient discriminatory information (Section 6.3).

- Combining Multiple Classifiers: We show how combining multiple classifiers using bagging techniques can improve classification accuracies (Section 6.4).

- Using Different Similarity Measures: We investigate the effect of using different similarity measures on classifier performance (Section 6.5).

- Using Different Classification Schemes: We have primarily used a Bayesian classifier for semantic classification of images. There has been a significant interest in recently proposed Support Vector Machines (SVM). In Section 6.6, we have compared the performance of SVM with Bayesian classification for the man-made vs. natural image classification problem.

6.1 Feature Selection

It has been observed that the performance of a classifier trained on a finite number of samples initially improves, attains a maximum, and then starts to deteriorate as more and more features are added. This phenomenon is called the *curse of dimensionality* [116]. Can the classification accuracy be improved using feature subset selection methods, i.e., using only a subset of the features from a high-dimensional feature vector? Selecting the optimal feature subset is in itself a problem of exponential time complexity and various heuristics have been proposed to yield approximate solutions.

The problem of automatic feature subset selection is defined as follows: given a set of features, select a subset that performs the best for a chosen classifier. In other words, the goal is to find the optimal subset of features from a high-dimensional

feature set under a given classification system. A robust procedure for feature subset selection would not only reduce the complexity of the classifier, but it may also improve the classification accuracies by eliminating redundant or highly correlated features.

A large number of algorithms have been proposed for feature subset selection. Jain and Zongker [117] illustrate the merits of various feature subset selection methods. While the branch-and-bound algorithm proposed by Narendra and Fukunaga [118] is an "optimal" procedure, it requires the feature selection criterion function to be monotone, i.e., the addition of new features to a feature subset should never decrease the value of the criterion function. The above requirement may not be true in small sample size situations. Due to this, it is desirable to use approximate solutions that are fast and also guarantee a near optimal solution. Pudil et al. [119] demonstrate that floating search methods show great promise where the branch-and-bound method can not be used. These results were also confirmed by Jain and Zongker [117]. Therefore, we tested the Sequential Floating Forward Selection (SFFS) method for feature subset selection. For details on SFFS algorithm, readers are requested to refer to Pudil et al. [119].

We have also applied a simple heuristic procedure based on clustering the features in order to remove redundant features [120]. A K-means clustering algorithm [108] was used to cluster the features. Note that normally, we cluster the patterns or the samples. The feature components assigned to each cluster are then averaged to form the new feature. Thus, the number of clusters determines the number of features in the subset. Although, this method does not guarantee an optimal solution for

feature subset selection, it does attempt to eliminate highly correlated features in high-dimensional feature vectors. We refer to this method as the Feature Cluster (FC) method.

6.1.1 Experiments using SFFS

We have experimented with feature subset selection on the *indoor vs. outdoor* classifier using the implementation of SFFS algorithm provided by Jain and Zongker [117]. We found the algorithm to be very slow over the entire training set of 2,541 training samples from database $D1$. We hence, sub-sampled 700 samples each from the training and test set for the feature subset selection process. The results of the feature subset selection using SFFS can be summarized as follows.

- The SFFS algorithm is extremely slow. It took 12 days on a Sun Ultra 2 Model 2300 (dual 300MHz processors) processor with 512 MB memory to select up to 67 features from the 600-dimensional feature vector.

- The best accuracy of 87% on the independent test set of 700 samples was provided by a feature subset of 52 components compared to the accuracy of 88.2% using all the 600 features.

- Training a new classifier on the 52 features selected by SFFS using the 2,541 samples from the training set of database $D1$ yielded an accuracy of 82.2% on the independent test set of 2,540 samples. The lower accuracy on a larger test set is in agreement with the results shown by Jain and Zongker [117] on the

pitfalls of using feature subset selection on sparse data in a high dimensional space.

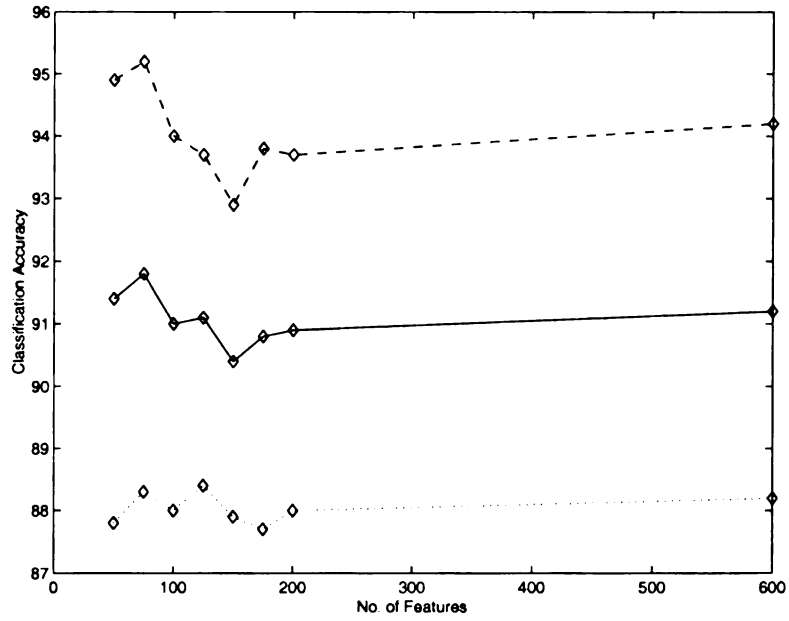
6.1.2 Experiments using FC

In order to remove correlated features, we also clustered the features. The spatial color moment features used in the *indoor vs. outdoor* and *orientation detection* classification problems (feature size of 600) were clustered to generate new feature vectors of sizes 50, 75, 100, 125, 150, 175, and 200. The Euclidean distance measure was used as the dissimilarity metric for clustering. The feature components assigned to each cluster were averaged to define a new feature. This clustering approach was much faster than the SFFS method taking only a few seconds on a training set size of 8,755 samples for the orientation detection problem. The classification accuracies for the two classifiers for various features are shown in Figures 6.1(a) and (b). A codebook size of 30 (the optimal size for the spatial color moments features) was used for all the features for the indoor vs. outdoor image classifier. Best classification accuracy of 91.8% on the entire database of 5,081 images (95.2% on the training set and 88.3% on an independent test set of 2,540 images) was obtained for the indoor vs. outdoor classifier when it was trained on the feature vector having 75 components. For the orientation detection problem, a codebook size of 100 was used for all the features. Best classification accuracy of 92.7% on the entire database of 17,901 images (97.8% on the training set and 87.9% on an independent test set of 9,146 images) was obtained when the classifier was trained on the feature vector with 150 components. On identifying

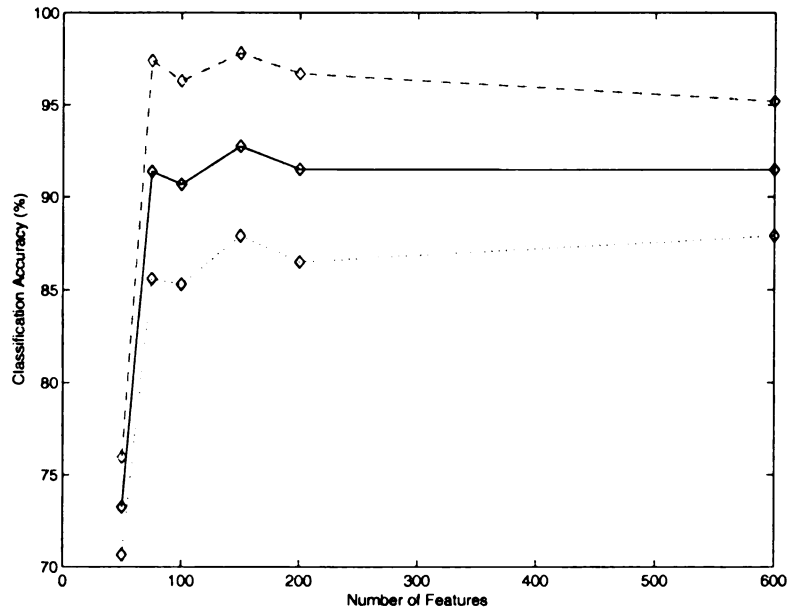
the feature components that were clustered together, we found that all groupings were formed within features of neighboring image regions. These preliminary results show that clustering the features (linear combination of feature components) is more efficient and accurate than the SFFS feature subset selection method for very high-dimensional feature vectors.

We applied the MMDL principle (described in Section 3.4.3) to generate the optimal codebook size for the new feature set. The MMDL criterion selected $q = 50$ (25 codebook vectors per class) as the optimal number of codebooks for the indoor vs. outdoor classifier and $q = 200$ (50 codebook vectors per class) for the orientation detection classifier based on the training samples. Figures 6.2(a)-(c) show the plots of $L(\mathcal{Y}|\boldsymbol{\theta}_{(q)}) + L(\boldsymbol{\theta}_{(q)})$ (criterion to be minimized in Eq. (3.9)) vs. the codebook size, q, for this new feature set of 75 components (for the indoor vs. outdoor classifier). As can be seen in the figures, $q = 20$ minimizes the criterion in Eq. (3.9) for the indoor class, while $q = 25$ minimizes the criterion for the outdoor class. Combining the two classes (Figure 6.2(c)) yields $q = 50$ as the optimal number of codebooks for the indoor vs. outdoor classifier based on the $2,541$ training samples.

Table 6.1 shows the classification accuracies for the two classifiers trained on these new features compared against the classification accuracies of the classifier trained on all the 600 spatial color moment features. The FC method for feature selection improved the classifier performance from 91.2% to 92.4% for the indoor vs. outdoor classification problem (on a database of $5,081$ images), while reducing the feature vector dimensionality from 600 components to 75 components. For the orientation detection problem, the FC method for feature selection improved the classifier perfor-



(a)



(b)

Figure 6.1: Classification accuracies for (a) indoor vs. outdoor classifier and (b) orientation detection classifier, trained on varying sized feature vectors generated by clustering (FC method) the 600-dimensional spatial color moment features; *dashed line*, *dotted line*, and *solid line* represent the classification accuracies on the training set, test set, and the entire database, respectively.

mance from 92.0% to 93.1% on a database of 17,901 images. Note that the low-level spatial color moment features used for these classification problems are extracted over 10×10 sub-blocks in the image. Usually, neighboring sub-blocks in an image have similar features as various objects span multiple sub-blocks (e.g., sky, walls, roads, forest, etc., may span a number of sub-blocks in an image). Thus, feature selection based on a combination of features (feature clustering) is more effective than selecting a subset of features (SFFS). We feel that other linear and non-linear techniques for feature extraction such as PCA, Discriminant Analysis, and Sammon's non-linear projection algorithms may be as effective as feature clustering in reducing the feature dimensionality.

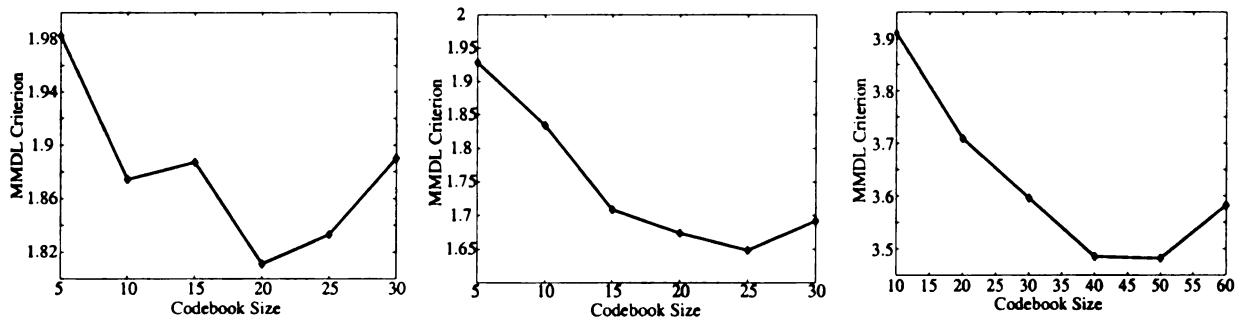


Figure 6.2: Determining codebook size for the 75-dimensional feature vector (generated using the FC method) for the indoor vs. outdoor classification problem; (a) indoor class; (b) outdoor class; (c) indoor and outdoor classes combined.

Table 6.1: Classification accuracies for the indoor vs. outdoor (I/O) and orientation detection (OD) image classification problems.

Classification Problem	New Feature Size	Accuracy (%)	Old Feature Size	Accuracy (%)
I/O	75	92.4	600	91.2
OD	150	93.1	600	92.0

6.1.3 Weighting Features

An appropriate weighting of the feature vectors can also increase the accuracy of the classifiers. However, automatically selecting the optimal weights for individual feature components is an unsolved problem. We experimentally show how scaling the feature components to a common scale (the spatial color moment features have different scales for the L, U, and V components) improves the classification accuracy for the orientation detection classifier. The 150 feature components selected by the feature selection algorithm for the orientation detection problem were normalized to the same scale[1] as follows. Let y_i represent the ith feature component of a feature vector $\boldsymbol{y}$. Let min_i and max_i represent the range of values for the ith feature component over all samples. Then the scaled feature component, y_i', can be defined as

$$y_i' = \frac{y_i - min_i}{max_i - min_i}.$$

Table 6.2 shows the classification accuracy for the orientation detection problem using the scaled features. In this case, scaling improves the classification accuracy by over 3% on the independent test data. Automatically extracting the optimal weight vector for a given feature vector is a promising direction for future research [121, 122].

[1] We acknowledge Wei-Shiuan Huang and Changjiang Yang of PRIP lab, Department of Computer Science and Engineering, Michigan State University for their suggestion to normalize the feature components to the same scale.

Table 6.2: Classification accuracies for the orientation detection classifier using unscaled (150-dimensional feature vector extracted in Table 6.1) and scaled features; the training and test set sizes are 8,755 and 9,146, respectively.

Feature	Accuracy on Training Set (%)	Accuracy on on Test Set (%)
Unscaled	97.1	89.4
Scaled	97.3	92.5

6.2 Incremental Learning

It is well known that the classification performance depends on the training set size; more comprehensive a training set, better is the classification performance. Table 6.3 compares the classification accuracies of the *indoor* vs. *outdoor* image classifier (trained on spatial color moment features) with increasing training set sizes. As expected, increasing the training set size improves the classification accuracy. When we trained the vector quantizer with all the available 5,081 images using the color moment features, a classification accuracy of 95.7% (resubstitution case) was obtained compared to 88.2% using the holdout method. This shows that the classifier still has the capacity to learn provided additional training samples are available. The above observations illustrate the need for an *incremental* learning paradigm for the Bayesian classifiers.

Collecting a large and representative training set is expensive, time consuming, and sometimes not feasible. Therefore, it is not realistic to assume that a classifier can be initially trained on a *comprehensive* training set. Rather, it is desirable to incorporate learning techniques in designing a classifier [123]. Over time, as additional training samples become available, the classifier should have the capability to adapt

Table 6.3: Effect of increasing the size of training data on classification accuracy for the indoor vs. outdoor classifier; Test and training sets were different.

Training Set Size	Ind. Test Set Size	Accuracy (%)
700	2,540	75.3
1,418	2,540	79.8
1,768	2,540	86.0
2,192	2,540	86.4
2,541	2,540	88.2

to the new data while retaining the discriminating information which it has already learnt. At the same time, the training set size can become extremely large over time and it may not be feasible to store all of the previously learnt data. Therefore, instead of re-training the classifier on the entire training set every time new samples are collected, it is more desirable to *incrementally* update the classifier. For the Bayesian classifier used here, the training set is represented in terms of the codebook vectors ($\boldsymbol{v_j}$). Incremental learning involves updating these codebook vectors as new training data become available.

6.2.1 Updating the Classifier

One method to re-train the classifier is to simply train it with the new data, i.e., start with the previously learnt codebook vectors and update them using the new data. This straight-forward method, however, suffers from the drawback that it does not assign an appropriate weight to the previously learnt information. In other words, if a classifier was trained on a large number of samples (say, a few hundred) and then a small number of new samples (say, 10 samples that are mostly outliers) are used

to incrementally train the classifier using the above learning paradigm, the new data will unduly influence the current value of the codebook vectors. Learning with these outliers will in fact lead to un-learning of the distribution based on previous samples. Table 6.4 demonstrates the results of updating the indoor vs. outdoor classifier using this scheme. The indoor vs. outdoor classifier was initially trained on $1,418$ images and yielded an accuracy of 79.8% on an independent test data of $2,540$ images. The classifier was then incrementally trained with an additional 350 images. As can be seen, the performance of the updated classifier deteriorates on the independent test data to 63.7%. When the classifier was further trained on an additional 773 samples using the naive approach, the accuracy on the independent test set slightly improves to 72.5%. Note that when the all training data were used ($1,418 + 350 + 773 = 2,541$ samples), the accuracy on the independent test set was 88.2% (Table 6.3). These results demonstrate that any robust incremental learning scheme must assign an appropriate weight to the already learnt distribution (codebook vectors).

Table 6.4: A naive approach to incremental learning using newly acquired data (350 and 773 images) to improve the performance of a classifier that was previously trained on $1,418$ images; accuracies are presented on an independent test set of $2,540$ images. The initial classification accuracy of 79.8% dropped using this approach.

Additional Training Samples	Accuracy on Test set (%)
350	63.7
773	72.5

6.2.2 Proposed Learning Scheme

The proposed learning scheme estimates or tries to generate the original training samples from the codebook vectors and augments these estimated samples to the new training set. The combined training set is then used along with the current codebook vectors to determine the new set of codebook vectors. This method differs from traditional bootstrapping methods [108] (which assume that the original training samples are available for sampling with replacement) in that the original training samples are not available. In our case, the original training samples need to be generated based on the proportion of these samples assigned to each codebook vector (m_j), and the codebook vectors themselves. Figure 6.3 illustrates this learning paradigm for a synthetic classification problem where 2-D samples are generated from two i.i.d. Gaussian distributions with mean vectors $[1,1]^T$ and $[2,1]^T$, respectively, and identity covariance matrices. We see that as the classifier is incrementally trained with additional data, the new codebook vectors approach the true mean vectors.

We have studied the following methods to generate training samples from a codebook vector:

- *Case* 1: Using duplicates of the codebook vectors as the samples.

- *Case* 2: Generating the samples from an i.i.d. multivariate normal distribution with identity covariance matrix centered at the codebook vectors.

- *Case* 3: Same as *Case* 2, except that we use a diagonal co-variance matrix instead of the identity covariance matrix. The diagonal elements correspond

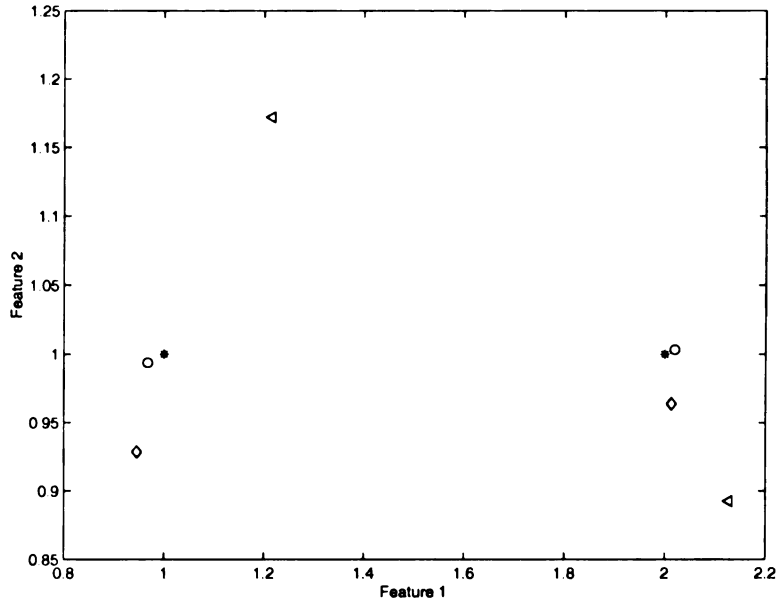


Figure 6.3: Incremental learning paradigm applied to a 2-class 2-dimensional synthetic classification problem; ($*$) represents the true means of the underlying densities; ($\triangleleft$) represents the initial codebook vectors determined from 100 training samples per class; ($\diamond$) represents the codebook vectors trained using an additional 400 samples from each class; and ($\circ$) represents the codebook vectors trained with an additional 500 samples from each class.

to the individual variances of features of the training samples assigned to the respective codebook vector.

- *Case* 4: Generating the samples from an i.i.d. multivariate normal distribution with identity covariance matrix centered at the mean vector of the training patterns assigned to a codebook vector. Note that each codebook vector need not be the mean of the training samples assigned to it, as both positive and negative examples influence the codebook vectors.

- *Case* 5: Same as *Case* 4, except that we use a diagonal co-variance matrix instead of the identity covariance matrix. The diagonal elements correspond to the individual variances of features of the training samples assigned to the

respective codebook vector.

The last four methods do not enforce the condition that the generated samples be closest to the codebook vector they are estimated from. Note that the number of samples generated from each codebook vector is the same as the number of original training samples assigned to that codebook vector. If we had chosen to use the EM algorithm to estimate the mixing parameters of the class-conditional densities, instead of LVQ, then, incremental learning could be achieved by using an on-line version of EM [124].

6.2.3 Experimental Results

We have tested the proposed incremental learning paradigm on the Bayesian classifier for indoor vs. outdoor, city vs. landscape, and orientation detection image classification problems. Initially, half the images from the database were used to train a classifier. The classifier was then incrementally trained using the remaining images. The performance of the classifier was then compared to that of a classifier trained on the entire set of database images (non-incremental learning). Table 6.5 shows the classification accuracies for the various classifiers with and without incremental learning. The best classification accuracies achieved for each of the classifiers were 95.9% for the city vs. landscape classifier (on 2,716 images), 94.6% for the indoor vs. outdoor classifier (on 5,081 images) and 95.3% for the orientation detection classifier (on 17,901 images). When each of the classifiers was trained using the entire training database, the accuracies achieved were 97.0%, 95.7%, and 95.2%, respectively. These

results show that a classifier trained incrementally is able to achieve almost similar accuracies as one trained with all the data. We feel that the reduction in classification accuracies of around 1% for the indoor vs. outdoor and city vs. landscape classifiers is mainly due to our inability to reliably generate representative training samples from the codebook vectors. We further see that all the five methods used to estimate the training samples perform equally well. Since *Case* 1 requires the least additional storage (only one real number denoting the total number of training samples used to train the classifier so far), it seems to give the best accuracy vs. storage trade-off.

Table 6.5: Classification accuracies (in %) with and without incremental learning.

Incremental Learning Method	City vs. Landscape	Indoor vs. Outdoor	Orientation Detection
Case 1	95.9	94.1	95.3
Case 2	95.8	94.3	95.3
Case 3	95.8	94.5	94.9
Case 4	95.8	94.3	95.1
Case 5	95.9	94.6	94.2
Non-Incremental	97.0	95.7	95.2

6.3 Reject Option

It is well known that introducing a reject option in a classification problem can result in a reduction in the error rate. The most widely used reject criteria for Bayesian classifiers are the *distance reject option* and the *ambiguity reject option* [125, 126, 127, 128]. The *distance reject option* addresses the problem of outlier detection, i.e., a test pattern is rejected if its distance to the nearest training pattern (or the stored prototypes) is larger than a threshold. The *ambiguity reject option* rejects those test

patterns which lie near the class boundaries and for which the classifier assigns a low confidence value. In this paper, we address the problem of improving the classification accuracy of a VQ-based supervised Bayesian classifier by developing a reject option. We show how a simple, yet novel scheme for local adaptation of rejection criteria can improve the error vs. reject characteristics of our classifier over those achieved by using the commonly adopted global thresholds for ambiguity and outlier rejection.

6.3.1 Rejection Scheme

The goal of rejection is to improve the classification accuracy by rejecting outliers and patterns for which the classifier has a low confidence. The outlier and ambiguity reject options can be formalized for the Bayesian classifiers as follows. Let $p(\omega_i \mid \boldsymbol{y})$ represent the posterior probability of class ω_i given $\boldsymbol{y}$. Then pattern vector $\boldsymbol{y}$ is classified into class ω_i if the following conditions are satisfied: (i) $p(\omega_i \mid \boldsymbol{y}) \geq p(\omega_j \mid \boldsymbol{y})$, $\forall j \neq i$; (ii) $p(\omega_i \mid \boldsymbol{y}) \geq t_1(\boldsymbol{y})$; and (iii) $\frac{p(\omega_i|\boldsymbol{y})}{p(\omega_j|\boldsymbol{y})} \geq t_2(\boldsymbol{y})$, $\forall j \neq i$; where $t_1(.)$ and $t_2(.)$ represent the thresholds for outlier and ambiguity rejection, respectively. If either (ii) or (iii) is not satisfied, pattern $\boldsymbol{y}$ is rejected.

Defining Local Thresholds

Typically, the two threshold (t_1 and t_2) are selected globally, i.e., the thresholds are constant over the entire feature space. Here, we show how computing the above thresholds in local regions (around the test patterns) improves the error vs. reject characteristics of the VQ-based Bayesian classifier. The feature space is divided into regions and thresholds are selected for each region. The higher the classification

accuracy in a region, the lower is the threshold value. Note that the codebook vectors ($\boldsymbol{v_i}$) act as prototypes of the training samples and can be used to partition the feature space into various regions. We have used two schemes to partition the feature space. The first scheme uses the Voronoi tessellation based on the codebook vectors. Let $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$ represent the partition of the feature space $\mathcal{R}$, where $\mathcal{R}_i$ is defined as,

$$\mathcal{R}_i = \{\boldsymbol{y} \mid \|\boldsymbol{y} - \boldsymbol{v_i}\|^2 \leq \|\boldsymbol{y} - \boldsymbol{v_j}\|^2,\ i \neq j,\ 1 \leq j \leq q\}, \tag{6.1}$$

where q is the codebook size. The second scheme further partitions $\mathcal{R}_i$ based on the second closest codebook vector as follows:

$$\mathcal{R}_{i,j} = \{\boldsymbol{y} \mid \|\boldsymbol{y} - \boldsymbol{v_i}\|^2 \leq \|\boldsymbol{y} - \boldsymbol{v_j}\|^2 \leq \|\boldsymbol{y} - \boldsymbol{v_k}\|^2, \tag{6.2}$$
$$i \neq j \neq k,\ 1 \leq k \leq q\}.$$

Figure 6.4 shows the partitioning for a set of 4 codebook vectors, $\boldsymbol{v_i}$, $1 \leq i \leq 4$. The solid lines represent the partitions considering only the near neighbor. The dashed lines represent the partitions when two nearest neighbors are considered. The main goal of partitioning the high-dimensional feature space is to identify those regions for which the classifier has high accuracies. Test samples belonging to regions with low classification accuracy (either due to less discrimination power of the given features in that region, or non-availability of sufficient training samples from that region) are then marked for rejection. Finer the partitioning, the better the rejection scheme. However, more training samples are needed to robustly estimate the local thresholds for rejection in these fine regions. There is thus, a trade-off between the complexity

of the partitioning and the size of training set needed to reliably estimate the local thresholds.

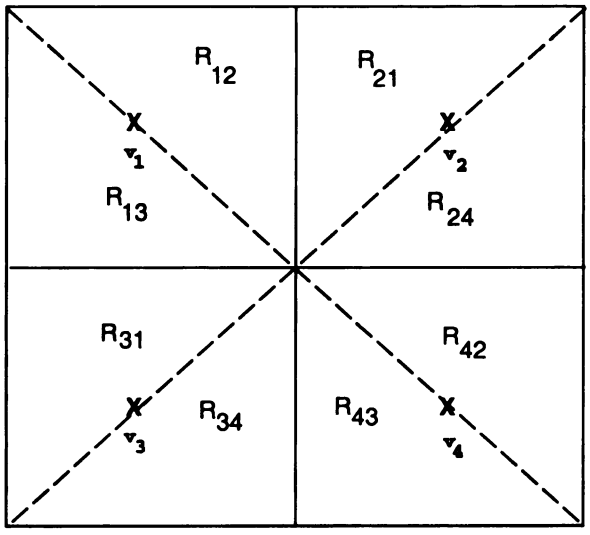


Figure 6.4: Partitioning the feature space based on nearest codebook vectors.

Threshold Selection

The outlier rejection threshold for region $\mathcal{R}_i$ is chosen as the minimum posterior probability of a correctly classified training sample falling in $\mathcal{R}_i$, i.e., $t_1(\boldsymbol{y}) = f_1(\mathcal{R}_i) = \min_{\boldsymbol{x} \in \mathcal{R}_i}(\max_j(p(\omega_j \mid \boldsymbol{x})))$, where $\boldsymbol{y} \in \mathcal{R}_i$ and $\forall \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{R}_i) \wedge (\boldsymbol{x}$ is correctly classified).

The ambiguity rejection threshold for a region is specified as follows. For every region $\mathcal{R}_i$, we define $g(\boldsymbol{y})$, the minimum ratio of the largest to the second largest posterior probability of a correctly classified pattern in $\mathcal{R}_i$, as

$$g(\boldsymbol{y}) = \min_{\boldsymbol{x} \in \mathcal{R}_i} (p(\omega_j \mid \boldsymbol{x}) / p(\omega_k \mid \boldsymbol{x})),$$

where $\boldsymbol{y} \in \mathcal{R}_i$, ω_j and ω_k are the classes with the largest and the second largest

posterior probability for sample $\boldsymbol{x}$, and $\forall \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{R}_i) \wedge (\boldsymbol{x}$ is correctly classified). Similarly, we define $h(\boldsymbol{y})$, the maximum ratio of the largest to the second largest posterior probability of a misclassified pattern in $\mathcal{R}_i$, as

$$h(\boldsymbol{y}) = \max_{\boldsymbol{x} \in \mathcal{R}_i} (p(\omega_j \mid \boldsymbol{x}) / p(\omega_k \mid \boldsymbol{x})),$$

where $\forall \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{R}_i) \wedge (\boldsymbol{x}$ is misclassified). Ambiguity rejection threshold is then defined as $t_2(\boldsymbol{y}) = f_2(\mathcal{R}_i) = \min(g(\boldsymbol{y}), h(\boldsymbol{y}))$.

In order to operate the classifier at different reject rates, the two thresholds are varied according to the following equation: $\Delta f_i(\mathcal{R}_j) \propto \frac{1}{r_j}$, where $i \in \{1, 2\}$ and $r_j = \frac{N_j^C + 0.5}{N_j^I + 0.5}$ is the ratio of correctly classified training samples (N_j^C) to the number of misclassified training samples (N_j^I) in region $\mathcal{R}_j$. This heuristic forces the local thresholds to increase at a higher rate in regions with a higher misclassification rate.

6.3.2 Experimental Results

We have tested our rejection paradigm on the man-made vs. natural (M/N), indoor vs. outdoor (I/O), and orientation detection (OD) image classification problems. These three classifiers were trained on high-dimensional feature vectors (145, 745, and 150 features, respectively) using Kohonen's LVQ program [113] and the optimal size of the codebook has been selected using a modified MDL principle The man-made vs. natural image classification is an extension of the *city vs. landscape* image classification with the inclusion of indoor images into the man-made class. Man-made class thus, consists of all the images that have man-made structures such as

buildings, roads, cars, furniture, etc., whereas natural images lack such structures. The indoor vs. outdoor classifier was trained using a weighted concatenation of the 600-dimensional spatial color moments and 145-dimensional edge direction coherence vectors.

We conducted three sets of experiments that differed in the method for threshold selection. The three rejection methods, called Method 1, Method 2, and Method 3, are defined as follows: (i) Use global threshold values; (ii) Local regions are chosen as Voronoi tessellations formed by the codebook vectors (Eq. 6.1); and (iii) Local regions are chosen on the basis of the two nearest codebook vectors (Eq. 6.2).

Table 6.6 presents the accuracy (in %) of the various classifiers on training and test data. The accuracies for the classifier with rejection are shown for Method 3 only. Figures 6.5(a)-(c) show the error vs. outlier, ambiguity, and both outlier and ambiguity reject curves, respectively, for the orientation detection classifier on the test set. The other classifiers show similar curves. The plots show how the locally adapted thresholds improve the error vs. reject characteristics of a classifier. Method 3 outperforms Method 1 (global) for all the three classifiers. On an average, Method 3 leads to an improvement of 3% in classification accuracy at a reject rate of 10% for the three image classification problems as against the classifier with no reject option (see Table 6.6). Figures 6.6(a)-(c) show a subset of images for which the classifier has a high confidence in classification. For these images, the extracted low-level features capture sufficient information to discern the true class. Figures 6.7(a)-(c) show a subset of images that were rejected at a reject rate of 10% by the three classifiers. Most of the images rejected by the man-made vs. natural classifier are images that contain

both man-made structures and natural scenery (either long distance city shots or city scenes with trees). The indoor vs. outdoor classifier has a low confidence for close-up images. These images do not present sufficient background information to discriminate between the two classes. The orientation detection classifier rejects images with uniform or homogeneous textures or close-up images. Color moment features do not capture sufficient information to discriminate between the four possible orientations for these images.

Table 6.6: Classification accuracies for image classification at various rejection levels (0%, 10%, 50%).

Problem	Training Set Size	Test Set Size	Accuracy with Rejection		
			0%	10%	50%
M/N	2,699	9,895	92.3	94.8	98.0
I/O	6,931	15,631	92.7	95.7	99.7
OD	8,755	17,901	93.0	96.3	99.1

6.4 Combining Multiple Classifiers

What is the best feature set and the best classifier for a given problem? This is a very difficult question which has evaded pattern recognition researchers for a long time. Instead of answering this question directly, researchers have shown that combining multiple classifiers can exploit the discrimination ability of individual feature sets and classifiers [129, 130, 131, 132, 133]. For a brief review on classifier combination, interested readers are referred to [134].

Bagging [135, 136] and boosting [137] are two commonly used methods of combining classifiers based on statistical re-sampling techniques. Bagging uses bootstrap

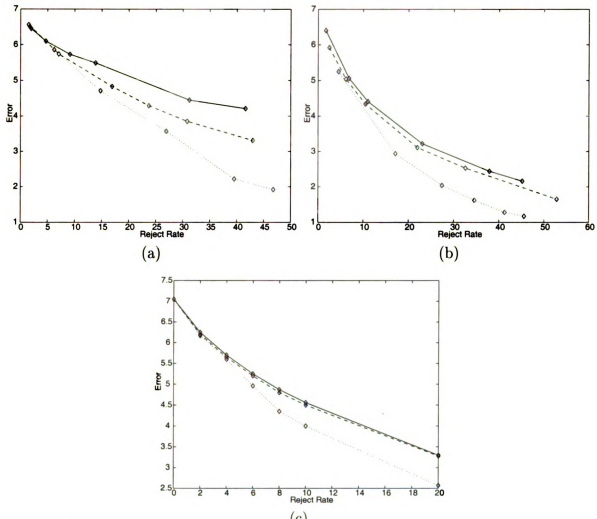
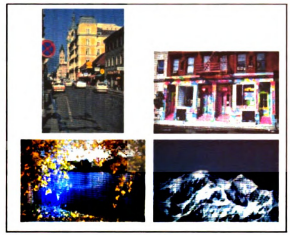
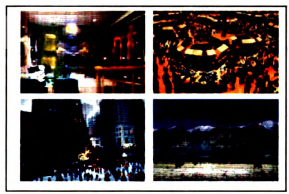


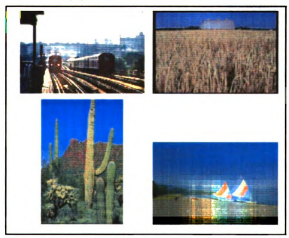
Figure 6.5: Error vs. reject curves for the orientation detection problem; (a) outlier rejection ; (b) ambiguity rejection; (c) combined outlier and ambiguity rejection; *solid*, *dashed*, and *dotted* lines represent Method 1, Method 2, and Method 3, respectively.



(a) Man-made vs. Natural

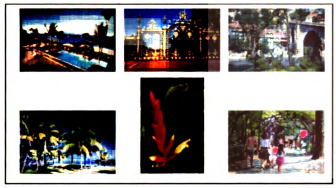


(b) Indoor vs. Outdoor



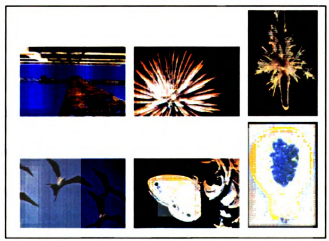
(c) Orientation Detection

Figure 6.6: Images classified with a high confidence value.



(a) Man-made vs. Natural

(b) Indoor vs. Outdoor



(c) Orientation Detection

Figure 6.7: Images rejected at 10% reject rate.

techniques (randomly draw n patterns with replacement from the original training data of size n) to generate a number of training sets. Different classifiers are then trained on these training sets. The final classification is based on the combined output (linear combination) of the various classifiers. Boosting is another statistical resampling technique where individual classifiers are trained hierarchically to learn more subtle regions of a classification problem. Each classifier in the hierarchy is trained on a training set that overemphasizes (assigns more weights to) the patterns that were misclassified in the earlier stages. A number of studies [135, 136, 133, 137, 138] have shown that bagging and boosting can improve the classification accuracy of "weak" classifiers (classifiers with near chance accuracies). However, if the classifier performance is good, bagging and boosting do not guarantee any improvement (in fact, the bagged or boosted classifier may perform worse than the original classifier). Specifically, Mao [133] showed that boosting can in fact reduce the classification accuracy for robust and efficient classifiers under a reject option. Mao argued that for robust classifiers, the misclassifications are mostly due to the lack of sufficient discrimination ability of the underlying features used for classification. Under these circumstances, boosting does not improve classification accuracies and using additional features with higher discrimination ability for these misclassified patterns is a more practical approach. For these reasons, we present results of combining classifiers using the bagging ensemble only.

We have developed bagged classifiers (we empirically set the number of classifiers in the ensemble to 10) for the man-made vs. natural and the indoor vs. outdoor image classification problems. The bagging algorithm is described as follows.

- Generate bootstrapped sets (here ten) of training data for the classification problem at hand.

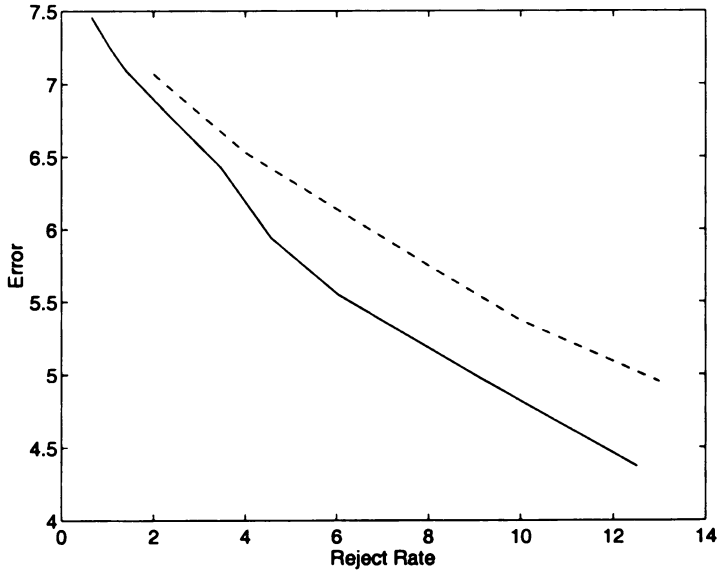
- Train different classifiers independently on the ten training sets.

- The output of the individual classifiers is linearly combined to yield the bagged classifier.

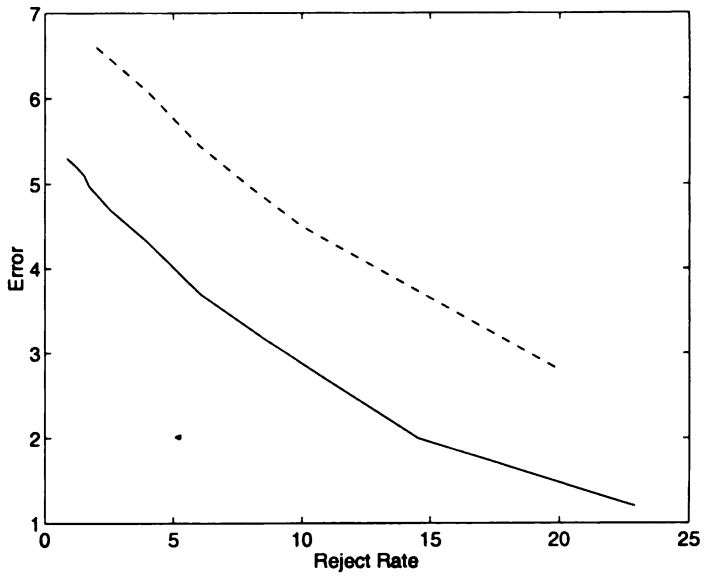
Figures 6.8(a) and (b) show the classifier accuracies for the bagged classifiers versus the original (single best) classifiers. The results show that bagging improves the classification accuracies for both the classifiers. For the man-made vs. natural image classifier, bagging improves classification accuracies by 0.6% at 10% rejection. In the case of indoor vs. outdoor image classifier, improvement by the bagging ensemble is much more significant. The bagged classifier improves the accuracy by 2% at 10% rejection. Although bagging shows improvement in classification accuracy, a thorough analysis is required to quantify the benefit. A number of research issues that need to be addressed are: (i) automatic selection of number of classifiers in the bagged ensemble; (ii) methods of combining the outputs of individual classifiers; and (iii) increase in the complexity of classification.

6.5 Different Similarity Metrics

We have shown in Section 6.1.3 how feature weighting can affect classifier performance. Another factor that affects classifier accuracy is the similarity measure used. In this Section, we analyze how changing the distance metric from Euclidean distance



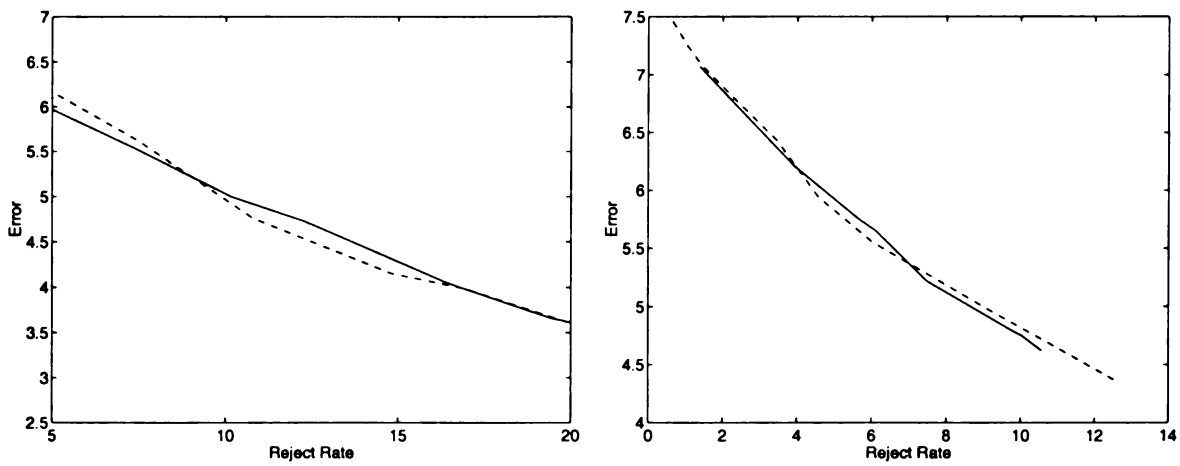
(a)



(b)

Figure 6.8: Error vs. reject curves for bagged classifiers (*solid line*) as against single best classifier (*dashed line*); (a) man-made vs. natural; (b) indoor vs. outdoor.

to City-block affects the classification accuracy of the man-made vs. natural image classification problem. Figure 6.9(a) compares the classification accuracies for the man-made vs. natural image classifier under Euclidean and City-block distance functions. Figure 6.9(b) compares the classification accuracies for the bagged man-made vs. natural image classifier (an ensemble of 10 classifiers was chosen for bagging) under Euclidean and City-block distance functions. Both the classifiers (using the two different distance measures) show comparable accuracies. Using Mahalanobis distance is also an option but with the large dimensionality of the feature vector used here, a reliable estimate of the covariance matrix will require a very large number of training samples.



(a) (b)

Figure 6.9: Error vs. reject curves for the man-made vs. natural image classifiers using City-block distance (*solid line*) and Euclidean distance (*dashed line*) as the dissimilarity measure; (a) single best classifiers; (b) bagged classifiers.

6.6 Experiments with SVM Classifier

Sections 6.1- 6.5 have addressed the problem of improving classifier robustness for the LVQ-based Bayesian classification framework. Can a different classification method provide a better classification accuracy? We empirically compare the performance of Support Vector Machines (SVMs) with the LVQ-based classifier for the man-made vs. natural image classification problem. SVMs have recently gained wide usage due to newly developed efficient learning techniques [139, 140]. We have used the SVM_light [141] software package for implementing this classifier. The classifiers were trained on 2,699 training samples and tested on 9,895 test samples using the edge direction coherence vector features. A number of classifiers were trained using different kernels (linear, polynomial, radial basis function, and sigmoid) for SVM. The best classification accuracy was achieved when a polynomial kernel function of degree 3 was used. Therefore, we report results for this SVM classifier only. The LVQ-based classifier achieved a classification accuracy of 96.7% on the training set and 92.3% on the test set. It selected 40 codebook vectors and took 1 msec to classify an image (excluding the feature extraction stage) on a 333.6 MHz SUN Sparc Ultra 10 machine. The SVM-based classifier achieved an accuracy of 98.3% on the training set and 92.9% on the test set. It selected 417 support vectors from the training set and took 10 msec to classify an image. Table 6.7 compares the performance of the LVQ-based classifier and this SVM classifier. Although the performance of the SVM classifier is slightly better than the LVQ-based classifier, the improvement is achieved at the cost of longer training and classification times. Combining the results of these

different two classifiers is a promising direction for future research.

Table 6.7: Comparing LVQ-based and SVM classifiers on the man-made vs. natural image classification problem.

Classifier	Accuracy on Training Set (%)	Accuracy on Test Set (%)	Vectors Stored	Classification Time per Image (msec)
LVQ	96.7	92.3	40	1
SVM	98.3	92.9	417	10

6.7 Discussion

Selecting the best feature set and the best classifier for a given classification problem is a challenging and difficult problem. We demonstrate how the classification performance can be improved using a number of techniques. Specifically, we show how incorporating feature selection, incremental learning, reject option, and classifier combination (bagging) schemes improve classifier performance. We empirically evaluate the effect of using different dissimilarity measures (Euclidean distance vs. City-block distance) on the classification accuracy of the man-made vs. natural image classifier. Both the dissimilarity measures yield comparable classification accuracies. Since LVQ enforces the use of a fixed distance metric, we have been unable to experiment with the Mahalanobis distance. Using other classifiers with the Mahalanobis distance measure seems a promising direction for future research. We also compare the LVQ-based classifier to SVM classifiers (which have been reported to have a very good generalization capability) for the man-made vs. natural image classifier. Although an SVM classifier yields slightly better classification accuracies than the LVQ-based classifier,

it is more complex to train (has a number of parameters that need to be tweaked), it requires more training and classification time, and it requires more storage space to represent the class models.

Chapter 7

Object Detection

We have shown in Chapter 4 how specific global low-level image features can be used for image classification. Organizing images into categories can be useful for browsing purposes (when the user has a vague remembrance of the pictures she would like to look at). However when users are searching for a particular concept (e.g., a person, a waterfall scene, a car, etc.), they are typically interested in specific objects and their inter-relationships. There is thus, an added need to identify objects of interest from images. Object identification in general is an unsolved problem in pattern recognition and computer vision. However, based on our success in image classification, we propose using the category information to detect objects that are likely to occur in such types of images.

The generated object-level tags can aid both in effective query formulation and in improving retrieval efficiency and accuracy. Figure 7.1(b) shows an example of a retrieval for the query in Figure 7.1(a) where the scene classification information is used. The query image is assigned the tags *outdoor* and *man-made* by the semantic

classifier. The retrieval is performed on the subset of database images that match the query indices. Since the semantic classifiers are not perfect, the image with the penguins is also retrieved in Figure 7.1(b). If information regarding the presence/absence of sky and vegetation can be extracted from outdoor images, then a user has additional freedom in formulating her query. She can now further restrict the query to those images that have sky but lack vegetation. Figure 7.1(c) shows the results of the new query. Using additional object information improves the retrieval, retrieving two additional images of sail boats in the top 10 retrievals.

7.1 Detecting Natural Textures

As the first task, we attempt to detect natural regions such as sky and vegetation in outdoor images. A future extension of the work is to detect man-made objects such as buildings, roads, cars, etc., in outdoor images. The problem of sky detection can be formulated as follows: Can regions of sky be automatically extracted from outdoor images? Examples of various sky scenes include clear skies, cloudy skies, sky at sunrise or sunset, night sky, etc. Detecting regions of sky and classifying them into sub-categories can generate additional semantic indices such as sunset scene, cloudy sky, night scene, etc. These tags can further aid in retrieval of images from large databases.

Detecting regions of sky in an outdoor image is a difficult problem. This is due to variations in color and texture of sky regions under different conditions (highly saturated colors in orange and yellow hue for sunset images, to textured clouds) as

(a)

(b)

(c)

Figure 7.1: Image retrieval results on a database of 23,898 images: (a) query image; (b) top 10 retrieved images when the search is restricted to outdoor images with man-made structures; (c) search is further restricted to images with sky and no vegetation.

well as presence of uniform colors and homogeneous textures in regions of water, walls, snow, etc. Figure 7.2 shows five database images with blocks (16×16) detected as sky by our algorithm.

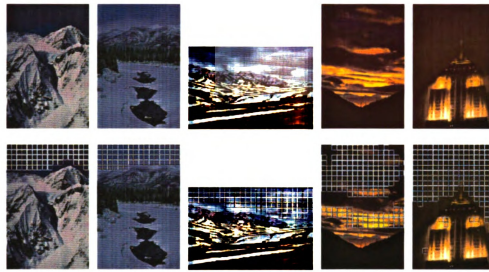


Figure 7.2: Images with blocks detected as sky.

The vegetation detector identifies regions of vegetation in outdoor images. Users can define queries in terms of presence/absence of vegetation (trees, grass, plants and bushes) and also their location in the image. Examples of various vegetation scenes include forest scenes (trees and dense foliage), regions of images with trees (either close-up or a long distance shot), grass and lawns, gardens and bushes, etc. Figure 7.3 shows five images from our database with blocks (16×16) of vegetation as detected by our algorithm. We can see that the various vegetation patches differ widely in their color and texture. While long distance shots of vegetation (and also lawns) have little texture and are green in color, close-up shots of trees and bushes are highly textured. On the other hand, scenes of flowers or trees in the autumn season have varied colors (fall colors which are not necessarily green). Under darkness, however, both texture

and color information is absent from vegetation scenes. These intra-class variations make the problem of automatic detection of vegetation regions difficult.

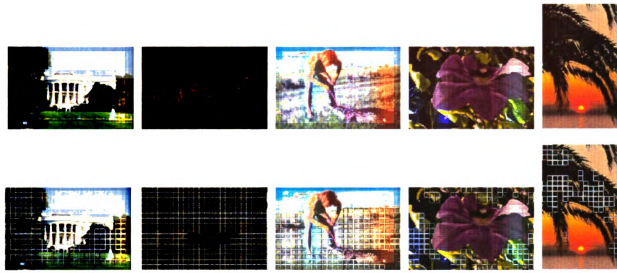


Figure 7.3: Images with blocks detected as vegetation.

We have used the Bayesian classification framework described in Chapter 3 for sky and vegetation detection. To localize these regions, we divide the image into 16×16 blocks. Color, texture, and spatial position features are extracted from each block. Classification of a block as *sky* or *non-sky* (for vegetation detector, each block is classified as *vegetation* or *non-vegetation*) is then based on the distribution of low-level features. We briefly describe the image features used to detect regions of sky in an image.

- Color: We use 6 color moment features (mean and variance in LUV color space) to describe the color in a block [28].

- Texture: We have used two different models that have been widely used in the texture analysis research.

- Gabor Features: We use 56 Gabor features (mean and variance in 4 orientations and 7 scale features) to describe the texture in a block [73]. Ma and Manjunath [38] have shown that Gabor features have a strong discrimination power for various types of textures.
- MSAR Features: We use 30 MSAR (Multi-resolution Simultaneous Auto-Regressive features: mean and variance in 3 resolutions and 5 features per resolution) to describe the texture in a block [70]. Picard and Minka [11] have shown that MSAR features outperform other texture features on the Brodatz dataset.

- Position: We use the center coordinates of a block as 2-dimensional position features.

The sky detector thus uses either 64 (color position, and Gabor) or 38 (color, position, and MSAR) feature components for classification.

For vegetation detection, we use a combination of color and MSAR features only and do not use the position features, since there are no specific restrictions on where plants, trees, branches, etc., can occur in a scene. Therefore, the feature vector for every 16×16 pixel block consists of 36 feature components.

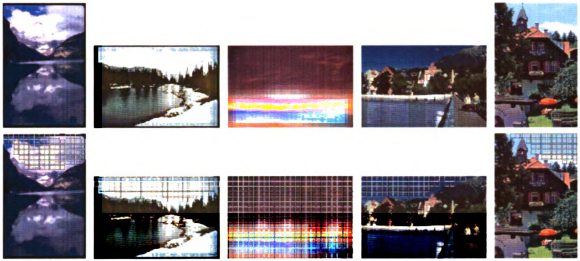
7.1.1 Detecting Regions of Sky

We have trained our *sky detector* on regions extracted from 471 images. These images are part of the Corel stock photo library and are stored in the JPEG format. A total of 29,614 sky blocks and 90,183 non-sky blocks were used for training. Both Gabor and

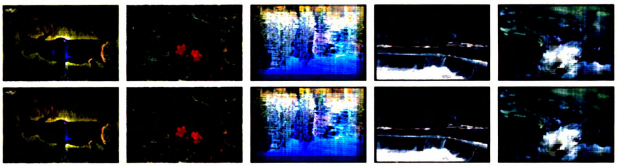
MSAR features yielded similar accuracies in terms of the number of blocks correctly classified. However, visual inspection on our database shows that MSAR features are more robust. We have empirically chosen 25 codebook vectors per class (total of 50 codebook vectors for our 2-class classification problem). The classifier achieves an accuracy of 89.4% and 96.7% for individual sky and non-sky blocks using MSAR features along with the color and position features. Figure 7.4 shows a subset of the independent test images where the classifier was able to reliably detect the presence of sky. These results show that the classifier can detect sky in varying conditions, such as in the presence of clouds, at sunrise or sunset, etc. The classifier has been trained to discriminate between sky and water (reflections of sky in water have been labeled as regions of non-sky). In most cases, the classifier is able to distinguish between sky and its reflection in water (with the help of position features). Figure 7.5 shows a subset of the test images where our classifier fails. The sky detector fails to detect regions of sky that have highly textured clouds. The detector also fails to discriminate regions of sky from regions of water in sunrise/sunset images (most sunset training images displayed scenes of entire sky and hence, the position features yielded very little discriminatory information) and other uniform regions such as regions of snow, water, and walls of buildings.

7.1.2 Detecting Regions with Vegetation

We trained the *vegetation detector* on 106,896 blocks from 400 images. The training set consisted of 24,818 vegetation blocks and 82,078 non-vegetation blocks. Using



(a) Images where sky is present.



(b) Images where sky is absent.

Figure 7.4: Correct classification results for sky detection.

a codebook size of 100 vectors per class, we achieved an accuracy of 94.4% on the training set (97.2% on non-vegetation blocks and 84.6% on vegetation blocks). Since regions of vegetation vary far more than those of sky, we require more codebook vectors per class for the vegetation detector. Figures 7.6(a) and (b) show a subset of images where regions of vegetation have been correctly classified. The vegetation detector has a very high accuracy on plants and foliage that are sufficiently textured (are not close-up images or too long distant shots). Figure 7.7 shows a subset of images with false detection and misclassification. Most of the misclassifications of vegetation blocks are usually due to lack of texture present in the regions (long distant shots or

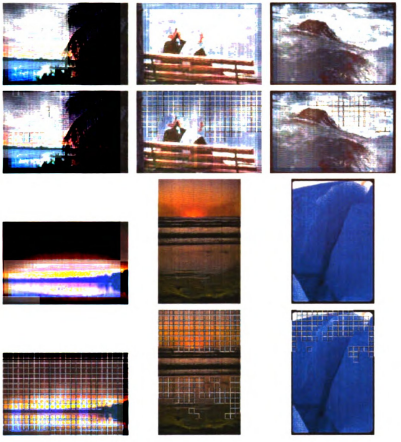
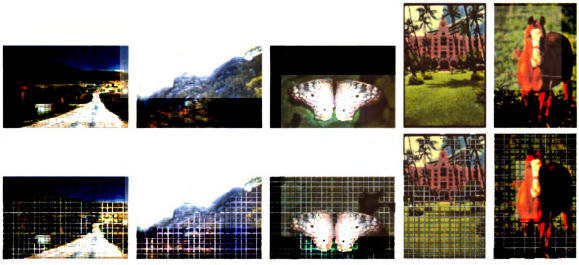


Figure 7.5: Incorrect classification results for the sky detector.

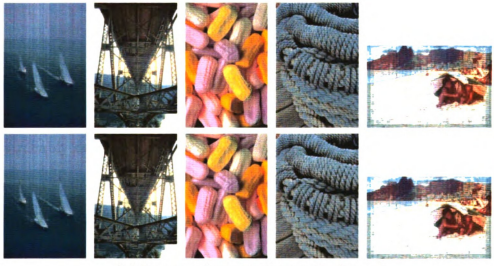
scenes of grasslands). Non-vegetation regions with strong green color (e.g., wings of a parrot, or mast of the sail in Figure 7.7) are misclassified as vegetation.

7.2 People Detection

Detecting the presence of people in images is an important step in generating semantic indices in image databases. Images can be categorized based on the presence/absence of people, number of people, or the names of the people (if faces can be reliably extracted and fed to a robust face recognition algorithm). In recent years, the problem of detecting people, especially face and skin detection, has received con-



(a) Images where vegetation is present.



(b) Images where vegetation is absent.

Figure 7.6: Correct classification results for vegetation detection.

siderable attention [142, 143, 8, 144, 145, 146]. A number of approaches have been reported in the literature for face detection. These include skin color model-based approaches [147, 148, 146], geometric model-based approaches [149], statistical approaches [145], and view-based approaches [144, 8, 150]. Although it is very easy for humans to locate and identify faces, no system exists that can reliably and automatically detect human faces in unconstrained conditions. The best reported algorithms

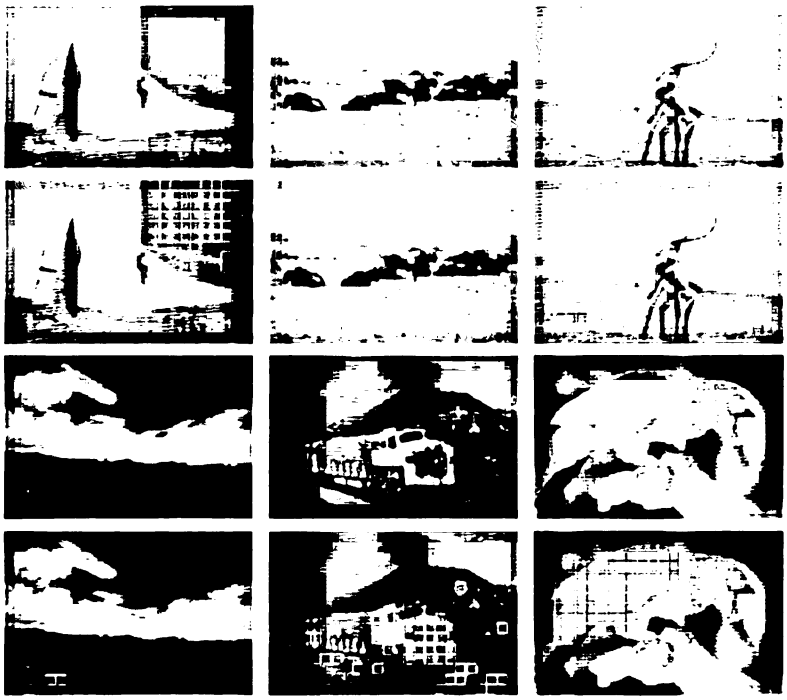


Figure 7.7: Incorrect classification results for the vegetation detector.

can only reliably detect frontal faces in constrained conditions. In the case of image databases, no assumptions can be made about the background, pose and size of faces, illumination, etc., and face detection cannot be used in isolation to detect people. Figure 7.8(a) shows three example images with people in them. While face detection algorithms can probably detect the male face in the first image in Figure 7.8(a), the other images pose a problem to such algorithms. A "weaker" (more general) filter, such as one based on the skin color model, will probably be more effective to detect people in such images. Figure 7.8(b) demonstrates the result of applying a skin color filter on the images in Figure 7.8(a). Note that the skin color model does not necessarily detect people - it acts as a filter to detect regions in the image that contain skin colored pixels. This can be followed by shape models to reliably detect head and limbs in the regions that pass the skin color filter.

(a)

(b)

Figure 7.8: Detecting people: (a) A subset of database images containing people; (b) skin regions detected by our algorithm.

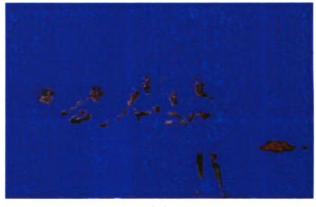
We address the problem of detecting people in arbitrary images in a hierarchical fashion. We currently employ three stages in the hierarchy, namely, skin color filter, connected component analysis, and texture filter. Each stage in the hierarchy acts as a filter to sieve out images that do not contain people. At the first stage, we employ a skin color filter to detect regions with skin color. It has been shown that human skin color forms a relatively tight cluster in certain color spaces (such as the HS plane of the HSV color space, the normalized RG plane, and the UV plane of the YUV color space) even when different races are considered [146]. A simple filter can thus be built using this model. We have used the skin color filter devised by Bakic and Stockman [151]. Figure 7.9(b) shows the result of applying the skin color filter on the picnic image in Figure 7.9(a). The second stage consists of extracting connected components of skin color. Small regions (an empirical threshold is set) are filtered out in this step. The third stage removes regions of homogeneous texture. Images with regions that pass the three filters are marked as likely to contain people. These regions are then represented by bounding rectangular blocks. Figure 7.9(c) shows the output of the skin detector after the third stage. Figure 7.10 shows a set of images with the corresponding output of the skin color filters. Note that although the system has a high accuracy where skin color regions are present, the lack of use of any shape and texture information leads to a number of false alarms. We evaluated the algorithm on a database of $4,158$ images of size larger than 256×256 (the database contained $2,121$ images with people and $2,037$ images without people). A hit rate (correct detection of people) of 94.4% was achieved as against a false alarm rate (algorithm detected skin regions where no people were present in the image)

of 30.4%. An empirical threshold of 25 × 25 was set on the size of the connected skin components. The 620 false alarms were attributed mainly to: animals and birds (51%), regions corresponding to wood (trees, branches, furniture, etc.) and flowers (18.9%), fireworks (18.4%), and dry land and buildings (8.9%). Figure 7.11 shows a set of images where non-human-skin regions pass the skin color filter. As future work, we are interested in adding geometric and shape-based filters to extract elliptical and cylindrical regions which correspond to head and limbs in people to reduce the false alarm rate [146]. The face detectors reported in the literature can also be applied to detect frontal faces in the regions that pass the skin detector.

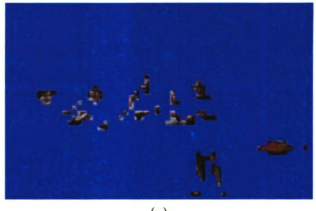
7.3 Text Detection

A successful multimedia search tool will involve a combination of image, audio, and textual information. In our thesis, we have concentrated on extracting semantics from visual content. Text present in images and video frames can also play an important role in understanding the content. For example, captions in news broadcasts and documentaries usually annotate information on *where*, *when*, and *who* of the reported events [152]. Ideally, the extracted text can be fed to an OCR for recognition. Identified words and phrases can then be used to develop semantic indices into images. Although embedded text provide important information about the image, it is not an easy problem to reliably detect and localize text regions. The size, color, orientation, and font of characters can vary considerably, embedded text is not necessarily horizontal, and the background may be very cluttered [152, 153].

(a)



(b)



(c)

Figure 7.9: Detecting Skin Regions: (a) input image; (b) pixels corresponding to skin color; (c) result after filtering regions of small size and homogeneous texture.

Figure 7.10: Detecting Skin Regions: Correct classification results.

Figure 7.11: Detecting Skin Regions: False alarms.

Jain and Yu [153] describe an automatic framework for locating text in images and video frames. Their system can take, as input, binary images, synthetic Web images, color images, and video frames. The original image is first preprocessed where the number of colors are reduced (color quantization using bit dropping and color clustering). Next, the image is decomposed into multiple foreground images. Connected component analysis and a text identification module are then applied to each foreground image. Finally, the output from all the channels (foreground images) are composed together to identify locations of text in the input image. Although the authors propose good results on scanned images, video frames, and Web images, the experiments with our database images yielded discouraging results. We have evaluated the algorithm on a database of 100 images of automobile racing that contained

text (mostly billboards with advertisements and text on the racing cars). The algorithm detected 257 blocks as text blocks in the 100 images. Of these, only 38 (14.8%) were correct classifications, and 219 (85.2%) were false detections. Even in cases where regions of text could be identified, the algorithm was unable to properly localize them, mostly combining multiple text blocks into a single large block. We attribute the poor performance of Jain and Yu's algorithm to the following two reasons. (i) The images present in our database are of very low resolution and text regions are small in size. Jain and Yu had reported results on high resolution images with predominant text regions. (ii) Our database images have a large variation in background. In most cases, the background cannot be easily distinguished from the foreground text regions. Moreover, man-made structures in the background lead to errors. These structures present strong horizontal and vertical edges similar to text regions. Figure 7.12 shows the results of the text location algorithm on three database images. Blocks detected as text are represented by rectangular bounding boxes in white. Developing algorithms to automatically locate text in general images is a promising direction for future research.

7.4 Discussion

Detecting specific objects reliably from general images is a very difficult problem. We show how image classification information can aid in the detection of specific objects occurring in that scene. Specifically, we address the problems of detecting regions of sky and vegetation in outdoor images using color, texture, and spatial position

Figure 7.12: Text location results on three database images.

features. Our classifier that classifies individual blocks in an image shows encouraging results. The classifier takes less than 20 ms to classify all the blocks in an image of size 384×256 pixels (note that this time does not include the feature extraction stage). This method can thus be used in the first stage of a multi-stage classifier where the later stages can combine information from different blocks to improve the classification accuracy. Moreover, the information about the primary codebook vector that a sky or vegetation block is assigned to can be further used to build semantic indices such as a day scene, a night scene, a sunrise/sunset scene, presence of clouds in the sky, close-up shot, long distant shot, etc., for an image. We also present empirical results for people detection using a skin color filter. Directions for future research include adding shape and geometric models to detect head and limbs in the regions that pass the skin color filter. We are also interested in extending our work to detect other objects such as buildings, cars, furniture, etc., in images classified as having man-made structures.

Chapter 8

Image Retrieval

Organizing large image databases into a small number of *categories* and providing effective indexing is imperative for accessing, browsing, and retrieving useful data in "real-time". If the categories are semantic in nature, a user has more flexibility in defining queries in terms of these semantic concepts rather than specifying queries on low-level features (color, texture, etc.). In this Chapter, we empirically evaluate the benefits of our semantic classifiers for image retrieval.

8.1 Automatic Extraction of Semantic Tags

We have shown in Chapter 4 that semantic scene information can be extracted from global low-level features. Our system achieves accuracies in the low 90% for indoor vs. outdoor and man-made vs. natural image classification problems. We now experimentally evaluate how organizing the database using semantic information improves retrieval efficiency and accuracy. We have conducted experiments on a database of

approximately 24,000 images. Every image in the database is fed to the indoor vs. outdoor and man-made vs. natural image classifiers. The classifiers are operated at a level corresponding to 5% rejection of the training patterns. Table 8.1 displays the labels assigned to all the images in our database. Around 5.3% of the database images were rejected by the indoor vs. outdoor classifier. The man-made vs. natural classifier rejected around 23% of the images. The higher rejection for the man-made vs. natural classifier was mainly due to the presence of many outliers (corresponding to images of fireworks, candy backgrounds, fruits and vegetables, images of birds and air shows, images of car races, and long distant sailing images) which were not represented in the training set.

Table 8.1: Labels assigned using semantic image classifiers.

Images	Indoor	Outdoor	Man-Made	Natural
23,898	4,661	17,989	5,760	12,432

8.1.1 Retrieval Efficiency

We have compared two methods for retrieval. The first method uses the semantic indices to filter the database images and then the retrieval is performed on the subset of images that match the query image indices. The second method retrieves images from the entire database. Our goal is to analyze the effect of organizing the database images into semantic categories on retrieval efficiency and accuracy.

One of the benefits of providing high-level semantic tags is to aid in browsing and formulating queries. Rather than formulate queries in terms of low-level features (color distributions), a user can now formulate queries in terms of semantic concepts.

For example, if a user is interested in a beach scene, the user needs to search through natural, outdoor images only. Object-level indices (presence/absence of sky, vegetation) can be further used to restrict the search on images with sky but that lack vegetation.

Another benefit in organizing the database is the improvement in classification efficiency (speed). Rather than search the entire database for a query, indexing the database reduces the search space. For example, in our database of 23,898 images, a search on a query image labeled as outdoor and man-made is performed on only 3,212 images (the search space is reduced to 13.4% of the database images) and on a query image labeled as outdoor and natural, the search is performed on only 6,808 images (28.5% of the database). Thus, indexing aids in retrieval efficiency. However, is this increase in efficiency achieved at the cost of accuracy? How does the performance of semantic classifiers affect retrieval accuracy (precision)?

8.1.2 Retrieval Accuracy

We define retrieval accuracy in terms of precision only, since it is extremely difficult to measure the recall (manually ranking the 24,000 images for each query is an extremely tedious task). We define precision as the number of images in the top K retrievals that match the semantic tags of the query image. For the sake of simplification, we use the tags assigned by only the indoor vs. outdoor and man-made vs. natural image classifiers. Note that this definition is more objective than the normal definition of precision (number of retrieved images that are "similar" to the query image). Different

users may vary in the notion of similarity between two images, however, users are more prone to agree on the class labels of two images. We have conducted the following experiments to measure the precision rate: (i) a pseudo-objective method; and (ii) a subjective method.

- Pseudo-objective Method: We present every image in the database, that was not rejected by both the classifiers, as a query. A set of $K = 20$ images are retrieved using low-level spatial color moment features from the entire database. The number of images (n_r), of the K retrieved images, that match the query indices is a pseudo-objective measure of precision. If the semantic classifiers were perfect, n_r/K would be an objective evaluation of the precision rate. For such perfect classifiers, the theoretical precision rate when the search is performed on the classified database would be 100%, i.e., all the images retrieved match the query image tags. We compare the experimental precision rate when the retrieval is performed on the entire database to the theoretical precision rate. Since our classifiers have a misclassification rate of around 6% (under a 5% reject rate), the maximum precision achievable theoretically is 88.4% ($94 * 94$ assuming that the indoor vs. outdoor and man-made vs. natural classifiers are independent), i.e., approximately 88 images out of 100 retrieved images will match both the tags of the query image. Note that the above analysis is negatively biased as it assumes a random selection of retrieved images from the set of images that match the query tags. In reality, the subset of images that match the query tag are further ranked based on the distribution of the low-level

features used for retrieval. Measuring the difference of the experimental precision rate, n_r/K, with the theoretical precision rate yields a pseudo-objective measure of the improvement in accuracy due to classification of the database images. Our experiments yielded an experimental precision rate of 72.5% when the entire database was searched as against a theoretically maximum possible precision of 88.4% when only the classified database is searched. Indexing or classification thus, improves the retrieval accuracy.

- Subjective Method: Manually evaluating the retrievals yields a more precise measure of precision. However, manually evaluating the precision rate for all the images presented as query is not possible. Not only is it a humungous task to evaluate the retrieval results, it is also subjective. We evaluated 25 randomly selected query images and evaluated the retrieval results. We counted the number of relevant images (images that matched the query image tags) in the K retrievals when (i) the entire database was searched; and (ii) when the database was classified based on the semantic indices (no filtering was done in the case of *don't care* labels in the query image; however, while counting the precision rate, the true class of the query image was taken into account). The experiment yielded an average precision rate of 73.4% when the entire database was searched and an average precision rate of 86% when the retrieval was performed on the filtered database. On an average for these 25 queries, only 28.3% of the database was searched when the semantic indices were used for filtering the database. These results further demonstrate that organizing the database

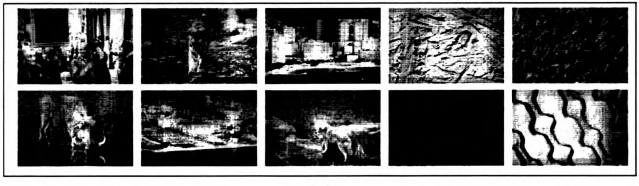
images improves both retrieval efficiency (speed) and accuracy (precision). Figure 8.1(a)-(c) demonstrate the improvement in precision for an example query. Figure 8.1(b) and (c) show the top 10 retrieval results for the query image in Figure 8.1(a) on the classified database and the entire database, respectively. The query was assigned the label indoor and man-made by the semantic classifiers. Only 2,173 of the 23,898 images were searched when the classification information was used for retrieval (Figure 8.1(b)). Note that although the indoor vs. outdoor classifier misclassified the query image (it is actually an outdoor image), the retrieval results on the classified database are more precise than those on the entire database. Our current retrieval method allows a user to select semantic tags during retrieval and in the above case, the search can be restricted to only those images with man-made labels, rather than to those images that have a combination of indoor and man-made labels. We achieve similar results for the modified query.

8.2 Discussion

In this Chapter, we demonstrate the benefits of organizing large image databases into semantic categories. Although the semantic classifiers are not perfect, we show how filtering the database according to semantic indices improves not only the retrieval efficiency (speed) but also the retrieval accuracy (precision). Our experiments on a database of 24,000 images yields an improvement in the precision rate of 14% when retrieval is performed on the classified database as opposed to the entire database.

(a)

(b)



(c)

Figure 8.1: Classification and retrieval: (a) query image; (b) top 10 retrieval results on classified database (2,173 images); (c) top 10 retrieval results on the entire database (23,898 images).

Chapter 9

Conclusion and Future Work

Content-based indexing and retrieval has emerged as an important area in computer vision and multimedia computing. Current solutions for searching image and video data primarily deal with textual and low-level image features. While extracting textual features requires manual intervention, the low-level features lack sufficient expressive power. User queries are typically based on semantics and not on low-level image features. It is therefore, imperative to provide semantic indices into large databases. Organizing and categorizing the images based on semantic indices can then aid in query formulation, browsing, and retrieval. This chapter summarizes the contributions of this thesis. Several directions for future research are also outlined.

9.1 Contributions

Psychophysical and psychological studies have shown that, in certain cases, humans can guess (with an accuracy of around 85%) the identity of scenes from low-frequency

representation of the images in as less as 40 ms without any kind of object identification. In this thesis, we show how specific global low-level image features can be used to learn certain semantic categories. Specifically, we have developed semantic image classifiers using a LVQ-based Bayesian framework for scene identification (indoor vs. outdoor image classification, man-made vs. natural image classification, sunset vs. forest vs. mountain image classification, and automatic image orientation detection) and object detection (detecting sky, vegetation, and people). Our semantic image classifiers have been tested on a large number of images (over 10,000 images each) and yield accuracies in the low 90%. Table 9.1 summarizes the results of our semantic image classifiers. The classifiers are also computationally efficient, each taking only 1 ms to classify an input image.

Table 9.1: Classification accuracies of various semantic image classifiers; I/O, M/N, OD, S/MF, and M/F represent the indoor vs. outdoor, man-made vs. natural, orientation detection, sunset vs. mountain and forest, and mountain vs. forest image classifiers, respectively.

Classification Problem	Training Set Size	Test Set Size	Accuracy on Test Set (%)
I/O	6,931	15,631	92.7
M/N	2,699	9,895	92.3
OD	8,755	17,901	93.0
S/MF	264	528	96.6
M/F	187	373	96.0

The accuracy of the above classifiers depends on the feature set used, the training samples, and their ability to learn from the training samples. We have developed a number of techniques to improve the performance and robustness of the Bayes classifiers. We have developed a learning paradigm to *incrementally train* a classifier as

additional training data become available. The proposed learning scheme estimates the already learnt training samples from the existing codebook vectors and augments these to the new training set for re-training the classifier. A classifier trained incrementally has comparable accuracies to the one which is trained using the true training samples.

We have developed a feature selection method to address the *curse of dimensionality* issue. The performance of a classifier trained on a finite number of samples starts deteriorating as more and more features are added after a certain number, called the optimum measurement complexity. We demonstrate how feature clustering can be used to considerably reduce (up to 87.5% reduction for the indoor vs. outdoor classification problem) the dimensionality of high-dimensional feature vectors. We further show that this dimensionality reduction leads to improved classification accuracies.

It is well known that introducing a reject option in a classification problem can result in a reduction in the error rate. We have developed a *rejection scheme* for VQ-based Bayes classifiers. We devise a measure of confidence in classification based on the maximum posterior probability of a class given the test sample and the ratio of highest to the second highest posterior probability. Based on the above confidence parameters, we define thresholds for outlier and ambiguity rejection. We empirically demonstrate how these thresholds can be locally adapted to improve the error vs. reject characteristics of a classifier.

It is believed that combining multiple classifiers leads to an increase in robustness of the classifier in terms of bias-variance dilemma, handling different feature types and scales, and exploiting the discrimination ability of individual features and classifiers

(experts). *Bagging* is a classifier combination strategy that uses bootstrap techniques (randomly draw n patterns with replacement from the original training data of size n) to generate a number of training sets. We empirically demonstrate how bagging can improve the accuracies of the semantic image classifiers.

Finally, we demonstrate on a large database of 24,000 images, how organizing the database according to the semantic indices not only improves retrieval efficiency, but also its accuracy. Efficiency is improved due to reduction in search space, only images with indices matching the query image indices are searched. Usually, improvements in efficiency are achieved at the cost of accuracy. However, we empirically demonstrate how semantic indices can be used to also improve the precision rate (accuracy) of the retrievals. Moreover, the semantic indices aid users in query formulation; users can formulate queries in terms of semantic concepts, such as "retrieve images of a city scene", rather than in terms of low-level color and texture features.

9.2 Future Research

There are a number of research issues which need to be addressed in the future. Our semantic image classifiers generate tags for the entire image. The system can be extended to classify segmented regions in an image. This would, in turn, require a robust segmentation algorithm. Other promising directions for future research are in the area of automatic feature extraction (can relevant features be automatically extracted from images given a classification problem?), classifier selection (compare and contrast multiple classifiers for a given problem), and information fusion (com-

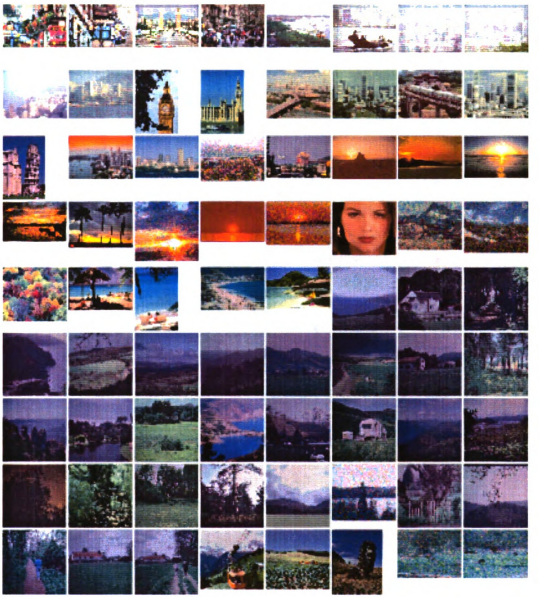
bining multiple features and multiple classifiers/experts). Developing other semantic classifiers (day vs. night) and detecting objects (buildings, cars, furniture, regions of text, etc.) are also imperative for improving the retrieval performance. The final goal is to generate multiple semantic indices to aid in browsing, searching, and retrieving image and video data in real-time.

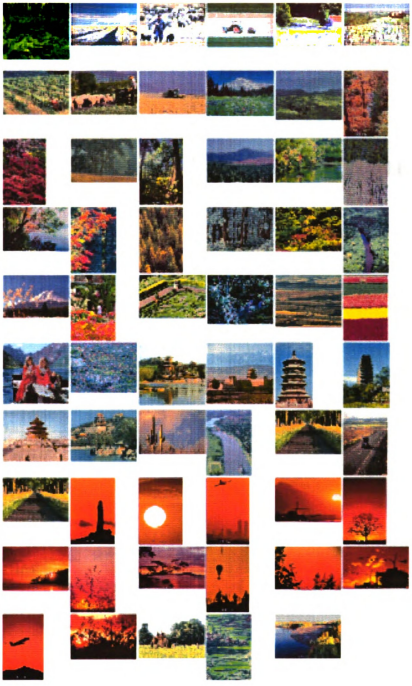
Appendices

Appendix A

Image Database Used for Defining Semantic Classes

A database of 171 images was used for defining semantic classes. These images were collected from a number of sources and include the 98 images used in [105].





Appendix B

Semantic Image Classification

Table B.1: Comparing semantic image classification systems reported in the literature; - signifies data not reported in the corresponding paper.

Index	Classification Problem	Classification Method	Database Size	Accuracy (%)	Cross Validation
Forsyth et al. [103]	12-Class Semantic Classification	Decision Tree	208	63.5	Yes
Yiu [102]	Indoor vs. Outdoor	K-NN, SVM	500	90	No
Szummer et al. [24]	Indoor vs. Outdoor	K-NN	1,343	90	Yes
Gorkani et al. [105]	City vs. Landscape	Rule-based	98	92.9	Yes
Torralba et al. [29]	Artificial vs. Natural	LDA	2,600	92	Yes
	2-Class Classification of Natural Images	LDA	1,500	97	-
	2-Class Classification of Artificial Images	LDA	-	98	-
Vailaya et al. [154]	Indoor vs. Outdoor	LVQ	15,631	92.7	Yes
	Man-made vs. Natural	LVQ	9,895	92.3	Yes
	Sunset vs. Forest vs. Mountain	LVQ	528	96	Yes

Bibliography

Bibliography

[1] "Gilder technology report." http://www.gildertech.com/html/gtr.html.

[2] "Netsizer review." http://www.netsizer.com/daily.html.

[3] http://www.yahoo.com.

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

[5] http://www.lycos.com.

[6] http://www.excite.com.

[7] http://www.google.com.

[8] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 23–38, January 1998.

[9] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and effective querying by image content," *Journal of Intelligent Information Systems*, vol. 3, pp. 231–262, 1994.

[10] A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Content-based manipulation of image databases," *SPIE Vol 2185: Storage and Retrieval for Image and Video Databases II*, pp. 34–47, February 1994.

[11] R. W. Picard and T. P. Minka, "Vision texture for annotation," *Multimedia Systems: Special Issue on Content-based Retrieval*, vol. 3, pp. 3–14, 1995.

[12] H. J. Zhang, C. Y. Low, S. W. Smoliar, and J. H. Wu, "Video parsing retrieval and browsing: An integrated and content-based solution," in *Proc. ACM Multimedia '95*, (San Francisco, CA), pp. 15–24, November 5-9, 1995.

[13] A. Hampapur, A. Gupta, B. Horowitz, C. F. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain, "Virage video engine," in *Proc. SPIE: Storage and Retrieval for Image and Video Databases V*, (San Jose, CA), pp. 188–197, February 1997.

[14] J.-Y. Chen, C. Taskiran, E. J. Delp, and C. A. Bouman, "ViBE: A new paradigm for video database browsing and search," in *Proc. Workshop on Content-Based Access of Image and Video Libraries (in conjunction with CVPR'98)*, (Santa Barbara, CA), pp. 96–100, June 1998.

[15] S. F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "VideoQ - An automatic content-based video search system using visual cues," in *Proc. ACM Multimedia Conf.*, (Seattle, WA), November 1997. Also Columbia University/CTR Technical Report, CTR-TR #478-97-12.

[16] J. R. Smith and S. F. Chang, "Visualseek: A fully automated content-based image query system," in *Proc. ACM Multimedia*, (Boston, MA), pp. 87–98, November 1996.

[17] W. Y. Ma and B. S. Manjunath, "Netra: A toolbox for navigating large image databases," in *Proc. IEEE International Conference on Image Processing*, vol. 1, (Santa Barbara, CA), pp. 568–571, October, 1997.

[18] S. Mehrotra, Y. Rui, M. Ortega, and T. S. Huang, "Supporting content-based queries over images in MARS," in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, (Ontario, Canada), pp. 632–633, June 3-6, 1997.

[19] A. Vailaya, A. K. Jain, and H. J. Zhang, "On Image Classification: City Images vs. Landscapes," *Pattern Recognition*, vol. 31, no. 12, pp. 1921–1936, 1998.

[20] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: A power tool for interactive content-based image retrieval," *IEEE Trans. on Circuits and Systems for Video Technology, Special Issue on Segmentation, Description, and Retrieval of Video Content*, vol. 8, pp. 644–655, September 1998.

[21] B. E. Rogowitz, T. Frese, J. Smith, C. A. Bouman, and E. Kalin, "Perceptual image similarity experiments," in *Proc. IS&T/SPIE Conf. on Human Vision and Electronic Imaging III*, (San Jose, CA), pp. 576–590, July 1998.

[22] T. V. Papathomas, T. E. Conway, I. J. Cox, J. Ghosn, M. L. Miller, T. P. Minka, and P. N. Yianilos, "Psychophysical studies of the performance of an image database retrieval system," in *Proc. IS&T/SPIE Conf. on Human Vision and Electronic Imaging III*, (San Jose, CA), pp. 591–602, July 1998.

[23] T. P. Minka and R. W. Picard, "Interactive learning using a 'society of models'," *Pattern Recognition*, vol. 30, no. 4, p. 565, 1997. also MIT Media Lab Perceptual Computing TR #349.

[24] M. Szummer and R. W. Picard, "Indoor-outdoor image classification," in *IEEE International Workshop on Content-based Access of Image and Video Databases (in conjunction with ICCV'98)*, (Bombay, India), January 1998.

[25] N. Vasconcelos and A. Lippman, "Library-based coding: a representation for efficient video compression and retrieval," in *Data Compression Conference'97*, (Snowbird, Utah), 1997.

[26] N. Vasconcelos and A. Lippman, "A Bayesian framework for semantic content characterization," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, (Santa Barbara, CA), pp. 566–571, June 1998.

[27] A. Vailaya, M. Figueiredo, A. Jain, and H.-J. Zhang, "A Bayesian framework for semantic classification of outdoor vacation images," in *Proc. SPIE: Storage and Retrieval for Image and Video Databases VII*, vol. 3656, (San Jose, CA), pp. 415–426, January 1999.

[28] A. Vailaya, M. Figueiredo, A. Jain, and H.-J. Zhang, "Content-based hierarchical classification of vacation images," in *Proc. IEEE Multimedia Systems'99 (Intl' Conf. on Multimedia Computing and Systems)*, vol. 1, (Florence, Italy), pp. 518–523, June 7-11, 1999.

[29] A. B. Torralba and A. Oliva, "Semantic organization of scenes using discriminant structural templates," in *IEEE Intl' Conf. on Computer Vision (ICCV'99)*, (Korfu, Greece), September 1999.

[30] A. L. Ratan, O. Maron, W. E. L. Grimson, and T. Lozano-Perez, "A framework for learning query concepts in image classification," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR'99)*, (Fort Collins, Colorado), pp. 423–429, 1999.

[31] "QBIC Demo." http://wwwqbic.almaden.ibm.com/cgi-bin/photo-demo/.

[32] MPEG Requirements Group, "MPEG-7 Context and Objectives." Doc. ISO/MPEG N2326, MPEG Dublin Meeting, July 1998.

[33] MPEG Requirements Group, "MPEG-7 Applications." Doc. ISO/MPEG N2328, MPEG Dublin Meeting, July 1998.

[34] MPEG Requirements Group, "MPEG-7 Context and Objectives." Doc. ISO/MPEG N2460, MPEG Atlantic City Meeting, October 1998.

[35] MPEG Requirements Group, "MPEG-7 Requirements." Doc. ISO/MPEG N2461, MPEG Atlantic City Meeting, October 1998.

[36] H. J. Zhang and D. Zhong, "A scheme for visual feature based image indexing," in *Proc. SPIE Conference on Storage and Retrieval for Image and Video Databases*, (San Jose, CA), pp. 36–46, February 1995.

[37] D. Zhong, H. J. Zhang, and S.-F. Chang, "Clustering methods for video browsing and annotation," in *Proc. SPIE Vol. 2670: Storage & Retrieval for Image and Video Databases IV*, (San Jose, CA), pp. 239–246, February 1996.

[38] W. Y. Ma and B. S. Manjunath, "Image indexing using a texture dictionary," in *Proc. SPIE Conference on Image Storage and Archiving System*, vol. 2606, (Philadelphia, PA), pp. 288–298, October 1995.

[39] I. Biederman, "On the semantics of a glance at a scene," in *Perceptual Organization* (M. Kubovy and J. R. Pomerantz, eds.), pp. 213–253, Hillsdale, NJ: Erlbaum, 1981.

[40] I. Biederman, "Aspects and extensions of a theory of human image understanding," in *Computational Processes in Human Vision: An Interdisciplinary Perspective* (Z. W. Pylyshyn, ed.), pp. 370–428, Norwood, NJ: Ablex, 1988.

[41] P. G. Schyns and A. Oliva, "From blobs to boundary edges: evidence for time and spatial scale dependent scene recognition," *Psychol. Sci.*, vol. 5, pp. 195–200, 1994.

[42] D. Navon, "Forest before trees: The precedence of global features in visual perception," *Cognitive Psychology*, vol. 9, pp. 353–383, 1977.

[43] R. M. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4–29, April 1984.

[44] R. M. Gray and R. A. Olshen, "Vector quantization and density estimation," in *SEQUENCES97*, 1997. http://www-isl.stanford.edu/~gray/compression.html.

[45] J. Rissanen, *Stochastic Complexity in Stastistical Inquiry.* Singapore: World Scientific, 1989.

[46] T. Pavlidis, "Algorithms for shape analysis for contours and waveforms," in *Proceedings of Fourth International Joint Conference on Pattern Recognition*, (Kyoto, Japan), pp. 70–86, November 7-10 1978.

[47] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[48] R. Schettini, "Multicolored object recognition and location," *Pattern Recognition Letters*, vol. 15, pp. 1089–1097, November 1994.

[49] B. M. Mehtre, M. S. Kankanhalli, A. D. Narsimhalu, and G. C. Man, "Color matching for image retrieval," *Pattern Recognition Letters*, vol. 16, pp. 325–331, March 1995.

[50] M. Stricker and A. Dimai, "Color indexing with weak spatial constraints," in *Proc. SPIE Vol.* 2670*: Storage & Retrieval for Image and Video Databases IV*, (San Jose, CA), pp. 29–41, February 1996.

[51] G. Pass, R. Zabih, and J. Miller, "Comparing images using color coherence vectors,," in *Proceedings of Fourth ACM Conference on Multimedia*, (Boston, MA), November 1996. http://simon.cs.cornell.edu/Info/People/rdz/rdz.html.

[52] B. Furht, ed., *The Handbook of Multimedia Computing: Chapter* 13 - *Content-Based Image Indexing and Retrieval.* LLC: CRC Press, 1998.

[53] L. S. Davis, "Shape matching using relaxation techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 60–72, January 1979.

[54] G. Cortelazzo, G. A. Mian, G. Vezzi, and P. Zamperoni, "Trademark shapes description by string-matching techniques," *Pattern Recognition*, vol. 27, no. 8, pp. 1005–1018, 1994.

[55] P. W. Huang and Y. R. Jean, "Using 2D C^+-strings as spatial knowledge representation for image database systems," *Pattern Recognition*, vol. 27, no. 9, pp. 1249–1257, 1994.

[56] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850–863, September 1993.

[57] D. J. Kahl, A. Rosenfeld, and A. Danker, "Some experiments in point pattern matching," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, pp. 105–116, February 1980.

[58] J. K. Cheng and T. S. Huang, "Image registration by matching relational structures," *Pattern Recognition*, vol. 17, no. 1, pp. 149–159, 1984.

[59] S. P. Smith and A. K. Jain, "Chord distribution for shape matching," *Computer Graphics and Image Processing*, vol. 20, pp. 259–271, 1982.

[60] W. Richards and D. D. Hoffman, "Codon representation on closed 2D shapes," *Computer Vision, Graphics, and Image Processing*, vol. 31, pp. 265–281, 1985.

[61] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Transactions on Computer*, vol. C-21, no. 1, pp. 269–281, 1972.

[62] S. A. Dudani, K. J. Breeding, and R. B. McGhee, "Aircraft identification by moment invariants," *IEEE Transactions on Computers*, vol. C-26, pp. 39–45, January 1977.

[63] A. K. Jain and A. Vailaya, "Shape-based retrieval: A case study with trademark image databases," *Pattern Recognition*, vol. 31, no. 9, pp. 1369–1390, 1998.

[64] M. Tuceryan and A. K. Jain, "Texture analysis," in *Handbook of Pattern Recognition and Computer Vision* (C. H. Chen, L. F. Pau, and P. S. P. Wang, eds.), pp. 235–276, World Scientific Publishing Company, 1993.

[65] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, pp. 786–804, 1979.

[66] M. Tuceryan and A. K. Jain, "Texture segmentation using Voronoi polygons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 211–216, 1990.

[67] D. Blostein and N. Ahuja, "Shape from texture: Integrating texture-element extraction and surface estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 1233–1251, December 1989.

[68] J. Besag, "Spatial interaction and the statistical analysis of lattice systems (with discussion)," *Journal of Royal Statistical Society Ser.B*, vol. 36, pp. 192–236, 1974.

[69] A. Pentland, "Fractal-based description of natural scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 661–674, 1984.

[70] J. Mao and A. K. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition*, vol. 25, no. 2, pp. 173–188, 1992.

[71] J. Malik and P. Perona, "Preattentive texture discrimination with early vision mechanisms," *Opt. Soc. Am. Series*, vol. A 7, pp. 923–932, 1990.

[72] J. M. Coggins and A. K. Jain, "A spatial filtering approach to texture ananlysis," *Pattern Recognition Letters*, vol. 3, pp. 195–203, 1985.

[73] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.

[74] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 39–55, 1987.

[75] R. Mohan and R. Nevatia, "Perceptual organization for scene segmentation and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 616–635, 1992.

[76] Y. Rui, A. She, and T. Huang, "Automated shape segmentation using attraction-based grouping in spatial-color-texture space," in *Proc. IEEE Intl' Conf. on Image Processing*, (Lausanne, Switzerland), pp. I53–56, September 1996.

[77] W. Y. Ma and B. S. Manjunath, "Edge flow: A framework of boundary detection and image segmentation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, (San Juan, Puerto Rico), pp. 744–749, June 1997.

[78] R. Morris, X. Descombes, and J. Zerubia, "Fully bayesian image segmentation-An engineering perspective," in *Proc. Intl' Conf. on Image Proc.*, (Santa Barbara, CA), October 1997.

[79] H. J. Zhang, J. Wu, D. Zhong, and S. W. Smoliar, "An integrated system for content-based video retrieval and browsing," *Pattern Recognition*, vol. 30, pp. 643–653, April 1997.

[80] A. Hampapur, R. Jain, and T. Weymouth, "Digital video segmentation," in *ACM Multimedia'94*, (San Francisco), pp. 357–364, 1994.

[81] R. M. Bolle, B.-L. Yeo, and M. Yeung, "Video query: Research directions," *IBM Journal on Research and Development*, vol. 42, no. 2, p. 233, 1998.

[82] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," *Multimedia Systems*, vol. 1, no. 1, pp. 10–28, 1993.

[83] H. J. Zhang, S. W. Smoliar, and J. H. Wu, "Content-based video browsing tools," in *Proc. SPIE Conference on Multimedia Computing and Networking*, (San Jose, CA), February 1995.

[84] M. Yeung and B.-L. Yeo, "Time-constrained clustering for segmentation of video into story units," in *13th International Conference on Pattern Recognition*, (Vienna), pp. 375–380, August 1996.

[85] W. Wolf, "Keyframe selection by motion analysis," in *ICASSP*, vol. II, pp. 1228–1231, May 7-10 1996.

[86] H. J. Zhang, J. Y. A. Wang, and Y. Altunbasak, "Content-based Video Retrieval and Compression: A Unified Solution," in *Proc. IEEE Intl' Conf. on Image Processing'97*, (Santa Barbara, CA), pp. 13–16, October, 26-29 1997.

[87] D. Zhong and S. F. Chang, "Video Object Model and Segmentation for Content-based Video Indexing," in *Proc. IEEE International Conference on Circuits and Systems '97*, (Hong Kong), June 1997.

[88] J. Courtney, "Automatic video indexing via object motion analysis," *Pattern Recognition*, vol. 30, pp. 607–625, April 1997.

[89] B. Shahraray and D. C. Gibbon, "Text based search of TV news stories," in *SPIE: Vol 2714*, pp. 512–518, 1995.

[90] A. G. Hauptmann and M. Smith, "Text, speech, and vision for video segmentation: The Informedia Project," in *AAAI Fall Symposium, Computational Models for Integrating language and Vison*, (Boston), November 1995.

[91] A. K. Jain, A. Vailaya, and W. Xiong, "Query by video clip," *Multimedia Systems: Special Issue on Video Libraries*, vol. 7, pp. 369–384, September 1999.

[92] N. Dimitrova and M. Abdel-Mottaleb, "Content-based video retrieval by example clip," in *Proc. SPIE Vol. 3022: Storage and Retrieval for Image and Video Databases V*, (San Jose, CA), pp. 59–70, 1997.

[93] R. Mohan, "Video sequence matching," in *Proc. Intl' Conf. on Acoustics, Appech, and Signal Processing (ICASSP'98)*, (Seattle, WA), May 1998.

[94] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker, "The QBIC project: Querying images by content using color, texture, and shape," in *Proc. SPIE: Storage and Retrieval for Image and Video Databases*, vol. 1908, pp. 173–187, February 1993.

[95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *Computer*, vol. 28, pp. 23–32, September 1995.

[96] A. Gupta and R. Jain, "Visual information retrieval," *Communications of the ACM*, vol. 40, pp. 70–79, May 1997.

[97] J. Smith and S. Chang, "Visually searching the web for content," *IEEE Multimedia Magazine*, vol. 4, no. 3, pp. 12–20, 1997.

[98] S. Chang, H. Sundaram, and W. Chen, "Videoq: An automated content based video search system using visual cues," in *4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, (Princeton, NJ), p. Demo III, October 1998.

[99] S. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong, "A fully automated content-based video search engine supporting spatiotemporal queries," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, pp. 602–615, September 1998.

[100] A. Vailaya, Y. Zhong, and A. K. Jain, "A Hierarchical System for Efficient Image Retrieval," in *13th International Conference on Pattern Recognition*, (Vienna), pp. C356–360, August 1996.

[101] J. Weng, "Cresceptron and SHOSLIF: Toward comprehensive visual learning," in *Early Vsual Learning*, pp. 183–214, New York: Oxford University Press, 1996.

[102] E. C. Yiu, "Image classification using color cues and texture orientation," Master's thesis, Department of EECS, MIT, 1996.

[103] D. A. Forsyth, J. Malik, M. M. Fleck, H. Greenspan, T. Leung, S. Belongie, C. Carson, and C. Bregler, "Finding pictures of objects in large collections of images," in *International Workshop on Object Recognition for Computer Vision*, (Cambridge, England), April 13-14 1996. http://www.cs.berkeley.edu/projects/vision/publications.html.

[104] H.-H. Yu and W. Wolf, "Scenic classification methods for image and video databases," in *Proc. SPIE, Digital Image Storage and Archiving Systems*, (Philadelphia, PA), pp. 363–371, October 1995.

[105] M. M. Gorkani and R. W. Picard, "Texture orientation for sorting photos "at a glance"," in 12^{th} *Int'l Conference on Pattern Recognition*, (Jerusalem), pp. 459–464, October 1994.

[106] R. C. Dubes and A. K. Jain, "Random field models in image analysis," *Journal of Applied Statistics*, vol. 16, no. 2, pp. 131–164, 1989.

[107] S. Z. Li, *Markov Random Field Modeling in Computer Vision.* Tokyo: Springer-Verlag, 1995.

[108] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[109] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, "Using vector quantization for image processing," *Proc. IEEE*, vol. 81, pp. 1326–1341, September 1993.

[110] M. Figueiredo and J. Leitão, "Unsupervised image restoration and edge location using compound Gauss-Markov random fields and the MDL principle," *IEEE Transactions on Image Processing*, vol. IP-6, pp. 1089–1102, August 1997.

[111] T. Cover and J. Thomas, *Elements of Information Theory.* New York: John Wiley & Sons, 1991.

[112] M. Figueiredo and A. K. Jain, "Unsupervised selection and estimation of finite mixture models," in *15th International Conference on Pattern Recognition*, (Barcelona, Spain), September 2000. To appear.

[113] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms," in *Proc. Intl' Joint Conf. on Neural Networks*, (Baltimore), pp. I 725–730, June 1992.

[114] A. K. Jain and A. Vailaya, "Image retrieval using color and shape," *Pattern Recognition*, vol. 29, pp. 1233–1244, August 1996.

[115] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 179–187, Feb 1962.

[116] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York, NY: Wiley, 1973.

[117] A. K. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, 1997.

[118] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Trans. Computers*, vol. 26, pp. 917–922, September 1977.

[119] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119–1125, Nov 1994.

[120] A. K. Jain and R. C. Dubes, "Feature definition in pattern recognition with small sample size," *Pattern Recognition*, vol. 10, no. 2, pp. 85–98, 1978.

[121] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody, "Further research on feature selection and classification using genetic algorithms," in *Proc. 5th Intl' Conference on Genetic Algortihms*, (Urbana-Champaign), pp. 557–564, July 1993.

[122] M. Raymer, W. F. Punch, E. D. Goodman, P. Sanschagrin, and L. Kuhn, "Simultaneous feature extraction and selection using a masking genetic algorithm," in *Proc. 7th Intl' Conference on Genetic Algorithms*, (San Francisco), pp. 561–567, July 1997.

[123] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[124] Y. Singer and M. Warmuth, "A new parameter estimation method for Gaussian mixtures," in *Advances in Neural Information Processing Systems 11* (M. S. Kearns, S. A. Solla, and D. A. Cohn, eds.), MIT Press, 1999.

[125] B. Dubuisson and M. Masson, "A statistical decision rule with incomplete knowledge about classes," *Pattern Recognition*, vol. 26, no. 1, pp. 155–165, 1993.

[126] K. Urahama and Y. Furukawa, "Gradient descent learning of nearest-neighbor classifiers with outlier rejection," *Pattern Recognition*, vol. 28, no. 5, pp. 761–768, 1995.

[127] T. M. Ha, "The optimum class-selective rejection rule," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 608–615, June 1997.

[128] T. Horiuchi, "Class-selective rejection rule to minimize the maximum distance between selected classes," *Pattern Recognition*, vol. 31, no. 10, pp. 1579–1588, 1998.

[129] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods for combining multiple classifiers and their applications in handwritten character recognition," *IEEE Trans. Systems, Man, Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.

[130] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66–75, 1994.

[131] I. Bloch, "Information combination operators for data fusion: A comparative review with classification," *IEEE Trans. Systems, Man, Cybernetics - Part A: Systems and Humans*, vol. 26, no. 1, pp. 52–67, 1996.

[132] J. Kittler, M. Hatef, and R. P. W. Duin, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 226–239, March 1998.

[133] J. Mao, "A case study on bagging, boosting, and basic ensembles of neural networks for OCR," in *Proc. IEEE Intl. Joint Conf. on Neural Networks*, (Anchorage, Alaska), May 1998.

[134] A. K. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4–37, Jan 2000.

[135] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[136] M. Skurichina and R. P. W. Duin, "Bagging for linear classifiers," *Pattern Recognition*, vol. 31, no. 7, pp. 909–930, 1998.

[137] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comp. and Sys. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[138] J. Huang and R. Kumar, "Boosting algorithms for image classification," in *Proc. ACCV'2000*, (Taipei, Taiwan), January 2000.

[139] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer-Verlag, 1995.

[140] C. Burges, "A tutorial on Support Vector Machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[141] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning* (B. Scholkopf, C. Burges, and A. Smola, eds.), Cambridge, USA: MIT Press, 1999.

[142] G. Yang and T. Huang, "Human face detection in a scene," in *Proceedings of IEEE conference on Computer Vision and Pattern Recognition*, (New York), pp. 453–458, 1993.

[143] V. Govindaraju, "Locating human faces in photographs," *International Journal of Computer Vision*, vol. 19, no. 2, pp. 129–146, 1996.

[144] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 39–51, January 1998.

[145] H. Schneiderman and T. Kanade, "Probabilistic modeling of local appearance and spatial relationships for object recognition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (Santa Barbara, CA), pp. 45–51, 1998.

[146] D. A. Forsyth and M. Fleck, "Automatic detection of human nudes," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 63–77, 1999.

[147] J. Yang and A. Waibel, "Tracking human faces in real time," in *Proc. IEEE Workshop on Applications of Computer Vision*, 1996.

[148] D. Chai and K. N. Ngan, "Locating facial region of a head-and-shoulders color image," in *Proc. Intl' Conference on Face and Gesture Recognition*, pp. 124–129, 1998.

[149] T. K. Leung, M. C. Burl, and P. Perona, "Finding faces in cluttered scenes using random labeled graph matching," in *Proc. Fifth Intl. Conf. on Computer Vision*, (Cambridge, MA), pp. 637–644, June 1995.

[150] N. Duta and A. K. Jain, "Learning the human face concept from black and white pictures," in *Proc. Intl' Conf. on Pattern Recognition*, (Brisbane), pp. 1365–1367, 1998.

[151] V. Bakic and G. Stockman, "Menu selection by facial aspect," in *Proc. Vision Interface '99*, (Trois-Rivieres, Canada), 1999.

[152] Y. Zhong and A. K. Jain, "Automatic caption extraction in compressed video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000. To appear.

[153] A. K. Jain and B. Yu, "Automatic text location in images and video frames," *Pattern Recognition*, vol. 31, no. 12, pp. 2055–2076, 1998.

[154] A. Vailaya and A. K. Jain, "Reject option for VQ-based bayesian classification," in *15th International Conference on Pattern Recognition*, (Barcelona, Spain), September 2000. To appear.