# COLOR-SPATIAL IMAGE INDEXING AND APPLICATIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jing Huang

August 1998

COLOR-SPATIAL IMAGE INDEXING AND APPLICATIONS

Jing Huang, Ph.D.
Cornell University 1998

We propose a new image feature called the *color correlogram* as a generic color-spatial indexing tool to tackle various problems that arise in content-based image retrieval and video browsing. Informally speaking, a correlogram represents the spatial correlation of colors in an image. While the computing and storage costs of correlograms match those of histograms, the presence of spatial information makes the former more *stable* to tolerate large image appearance changes than the latter. This makes the corrrelogram very attractive for applications such as content-based image retrieval and cut detection.

To validate this, we first show that the correlogram, used as an image feature, is *scalable* for image retrieval on very large image databases. Our experimental results on a database over 200,000 images suggest that the color correlogram is much more effective than the color histogram (and variants) with the same amount of information for these applications. We also propose a new distance metric called *relative distance metric* for comparing image feature vectors. It outperforms other distance functions in most cases and improves the performance of color histograms and histogram-based features. To further enhance the quality of retrieval, we then present two supervised learning methods — *learning the query*, *learning the metric* — and combine these learning methods with color correlograms. Our experiments show that these learning methods are quite effective with even a little effort from users.

We also adapt the correlogram to handle the problems of image subregion querying, object localization and tracking. We propose the *correlogram intersection* for object detection and *correlogram correction* for object localization. These simple methods perform better than methods based on color histograms.

Finally, we propose a method for hierarchical classification of images via supervised learning. This scheme uses correlogram as the low-level feature and performs feature-space reconfiguration using singular value decomposition to reduce noise and dimensionality. We use the training data to obtain a hierarchical classification tree that can be used to categorize new images. Our experimental results suggest that this scheme not only performs better than standard nearest-neighbor techniques, but also has both storage and computational advantages.

All our experimental results suggest that the color correlogram can serve as a good generic indexing tool for various image and video processing applications. Thus, it promises to be a basic building block for efficient and effective schemes to retrieve images from say, the world-wide web.

## BIOGRAPHICAL SKETCH

Jing was born on February 14, 1970 in Yueyang, a small city by the side of Dongting Lake in Hunan province of China. She grew up in the countryside where she enjoyed the scenery, the beautiful lotus flowers, and the delicious taste of Wucang fish. She then moved to Wuhan City and started her school life. She skipped two grades just so that she could go to the same class as her friends. This qualified her to enter the special Young Class of Tsinghua University in 1985. She finished her undergraduate studies in the Department of Applied Mathematics, preparing for graduate study in Finance with the ambition of contributing to the Chinese economy. Unfortunately, due to some political reasons, she had no choice but to stay in Applied Mathematics for three more years in the Masters program. She finally decided to go abroad and came to Cornell in 1993. She accepted the challenge of working in Computer Science, a new discipline for her, gradually passed all required exams and finally finished her PhD. In the mean time, she met Wei-Jing, who later became her husband and the inspiration and joy of her life. She will be taking up a position with IBM at T. J. Watson Research Center in August.

獻給耶和華, 我的牧者.

**ACKNOWLEDGEMENTS**

I would like to thank my advisor Ramin Zabih for his constant support and encouragement. He gave me enough time to start my research on computer vision and trusted that I would accomplish something. He gave me as much freedom as I needed to get my research work done.

I am grateful to my committee members Dan Huttenlocher, Ronitt Rubinfeld, and Robert Jarrow who spent time to read my thesis and gave me valuable comments, suggestions and corrections. Their support and serving on my committee meant a lot to me. Many thanks to my former advisor Nick Trefethen who taught me how to do research and helped me through a difficult time when I came to Cornell. I am also grateful to the administrative staff — especially Rose, Becky, Liz, Cay, Jan, Karla — and many others who helped me to finish my PhD.

I am greatly indebted to officemates S. Ravi Kumar and Mandar Mitra. They are not only my collaborators but also dear friends who share my happiness, sadness and worries. I especially thank Ravi who constantly inspires me with new problems and encourages me with his support. We also had a lot of fun playing Badminton, Tennis and Table Tennis. I was very fortunate to have Mandar as my TA (for CS681) and later as my officemate. I thank him for his patience and giving me help whenever I needed it from him. Mere words do not suffice to express my deep gratitude towards them.

Thanks to Deyu Hu who offered me great help when I was in my first year at Cornell. Thanks to Chi-Chao Chang for many hours of debugging and playing options to earn more money (even when we knew it was fake money). Many thanks also go to my former housemates in Our Father's House: Robin Gau, Jia-Huey Yuan, Shiow-Lih Chuang, and Matthew Williams.

I am very thankful that God has put in my life many wonderful brothers and sisters in my fellowship. I especially thank Ying Chen and Henry Lynn who introduced Jesus Christ to me and helped me to grow as a young Christian. Many thanks to Jia-Huey Yuan, Chien-Teh Chen, Jun Liang, Wei Guo, Qiubo Li, Ying Fan, Jun Xu, Xiaorong Lu, Anbo Liu, Weiwei Luo with whom I worked together to serve our fellowship. Their tender love and care towards me make my journey joyful and pleasant. I would also like to give thanks to Tony & Jenny Hsu, Charles & Lily Lee, Sidney & Amy Chen, Mancang & Jinzen Dong, Juliana Kan and Hanna Cheng who are older brothers and sisters to me in Christ and constantly pray for me when I need it most. I will cherish their love forever.

My thanks to my family — my parents, my brother and my sister for their love, care and support over all these years. Finally my thanks to my husband, my dearest Wei-Jing, who supports me in every circumstance, inspires me with his wisdom and knowledge, and helps me to grow spiritually and intellectually. Next to God's Grace, his commitment to me is the best thing that has happened in my life. My heart overflows with thanksgiving and praise to the Lord God Almighty who is my Shepherd. This thesis is dedicated to Him.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

畫意能達萬言

“A picture is worth ten thousand words.”
— *A Chinese proverb*

## 1.1 Motivation

When it comes to expressing ideas and conveying information, mankind has always preferred concrete visual means (images, painting) to more abstract counterparts (written text). This is evident from the antiquity of the former methods compared to the latter [Jain97]. The recent spate of technological advances in handling digital data have further strengthened our dependence on visual modes of communication. This can, for instance, be witnessed in the explosively growing amount of digital image data, especially with the proliferation of the world wide web.

Though it is fairly easy to create a large repository of visual data, the information stored there is virtually useless if it is unorganized. It is not unreasonable to draw an analogy between looking for a particular image on the web and searching for a book from a huge library without the aid of catalogs. The importance of the ability to search and retrieve images from an image collection cannot be overemphasized. Scientific organizations like NASA that possess large archives of photos and videos have to invest considerable efforts to manage this data properly so that search and retrieval become feasible.

The next obvious question is: how difficult is the problem of searching images? could one use the well-studied methods in text retrieval for images? The answer is unfortunately no. It is much harder to handle image data than text. A text document is “one-dimensional” (an array of words), whereas a digital image is “two-dimensional” (a video is “three-dimensional” because of the time component). Moreover, the size of image data is much larger than text. The most significant difference between text and image data is that words are in some sense semantic “objects”, while the image data need to be processed and interpreted to extract the perceptual meaning, which is a yet to be achieved task in compute vision and image understanding.

The standard indexing techniques that were designed for text (usually hash-based or B-tree based) are neither suitable nor sufficient for image data [Gros97]. This is because image data is likely to be explored, navigated, and retrieved by perceptual similarity, which is hard to capture. Additionally, this notion is subjective and application-specific. The indexing techniques of traditional databases cannot cope with these challenges.

One approach for indexing and retrieving image data is using manual text annotations. The annotations can then be used by some text information retrieval (TIR) systems [Salt89] to search images indirectly. There are several inherent problems with this seemingly attractive approach. First, since image data contains very rich information, it is very difficult to capture the content of an image using only a few keywords, not to mention the tedious work involved in such an annotation process itself. Second, the manual annotation process is quite subjective, ambiguous, and incomplete. If a query refers to image content that was not initially annotated, or if the user uses different words to describe the same image content, the text retrieval system will fail . For example, Figure 1.1 shows the top ten images of a query "blue flowers" from an art image database (from `http://www.thinker.org`). It is obvious that the retrieval is not satisfactory to paintings of "blue flowers". These ten images are retrieved because of the occurrence of "blue" inside the text annotations of those images.

The above problems have created great demands for automatic and effective techniques for *content-based* image (video) retrieval (CBIR) systems [FSN$^+$95, BFG$^+$96, PPS96, CS95, KCH95, SC96c, WKSS96, HJW94, CCWS$^+$97, NT92, SZ96, ZSW95, SK97, GTB97, Furh96, GJM97] etc.

Most CBIR systems adopt the following two-step approach to search image databases [SS94]: (i) (*indexing*) for each image in a database, a feature vector capturing certain essential properties of the image is computed and stored in a featurebase, and (ii) (*searching*) given a query image, its feature vector is computed, compared to the feature vectors in the featurebase, and images most similar to the query image are returned to the user (an overview of such systems can be found in [Comp95]). Such a system is called *Query-By-Example* (QBE).[1]

In a typical QBE system, a user poses a query by providing an existing image (or creating one by drawing), and the system retrieves other "similar" images from the image database. Content-based video browsing tools also provide users with similar capabilities — a user provides an interesting frame as a query, and the system retrieves other similar frames from a video sequence. For example, a FBI agent might want to locate a criminal from a video clip by supplying a mug shot of that criminal.

Besides the basic content-based image retrieval and video browsing tasks, several related problems also need to be addressed. What if the user gets back irrelevant results (see examples in Figure 1.2 and Figure 1.3. Actually very few relevant images to query images at the left top corner.)? In such cases, users might like to provide feedback information that describes the relevance of the retrieved results. The CBIR system should then process this feedback efficiently and return (better) results according to the user's criteria. (Notice that this dynamic querying may change the neighboring structure of each image in the database, thus rendering even special data structures inefficient.)

While most CBIR systems retrieve images based on overall image appearance comparison, users are also interested in *object searching* [FFB96, FMM$^+$96, RM97, CBGM97, Ma97, Gong98, GTB97, NT92]. In the case of object searching, the user specifies an "interesting" subregion (usually an interesting object) of an image as a query. The system should then retrieve images containing this subregion or object (according to human perception) from the image database. This task, called *image subregion querying* (which is a general recognition problem under arbitrary

[1]For a complete image database management system, there are other important issues need to be addressed [GJM97, Gong98], such as indexing data structures for search (R-tree variants), querying schemes, and query languages. These issues are studied by the database research community are not considered in our work.

context), is made challenging by the wide variety of effects (such as different viewing positions, camera noise and variation, object occlusion, etc. See Figure 1.4) that cause the same object to have drastically different appearances in different images.

The system should also be able to solve the *localization* problem, i.e., it should find the location of the object in an image. The lack of an effective and efficient image segmentation process for large, arbitrary, and heterogeneous image databases implies that objects have to be located in unsegmented images, making the localization problem more difficult.

Similar demands arise in the context of content-based video browsing. It is highly desirable to allows users to explore and to select video content from a large available source of video collections. Therefore, video browsing, which involves both parsing (or extracting) and organizing high-level content so that users can easily get an overview of a video, is a crucial part of video management. A primary task in video parsing is *cut detection*, which segments a video into different camera shots and helps to extract key frames for video parsing and querying. A flexible tool for browsing video databases should also provide users with the capability to pose object-level queries that have semantic content, such as “track this person in a sequence of video”. To handle such queries, the system has to find which frames contain the specified object or person, and has to locate the object in those frames.

When the size of image databases grows bigger and bigger, browsing and searching the image corpus through *semantic image categories* is the most desirable and useful way [CSBB97]. Most CBIR systems generate low-level image features such as color, texture, shape, motion, etc., for image indexing and retrieval. This is partly because low-level features (e.g., color histograms, texture patterns) can be computed automatically. The semantics of images, with which users prefer most of their interaction, however, are seldom captured by low-level features. On the other hand, there is no effective method yet to automatically generate good semantic features of general images. No contemporary computer vision or image understanding technique can automatically annotate the contents of generic real-world scenes. One common compromise is to obtain some semantic information through manual annotations, which is neither accurate nor efficient as mentioned before.

## 1.2 Challenges

### 1.2.1 Feature Extraction

The various tasks described above — content-based image retrieval, object-level image subregion querying, object localization, cut detection, and semantic organization of an image collection — become especially challenging when the image database is gigantic. For example, the collection of images available on the Internet is gigantic and unorganized. The huge size itself poses great challenges to searching images from the web [SC96b].

The indexing of an image database is often referred as feature extraction. Mathematically, a feature is an $n$-dimensional vector, with its components computed by some image analysis. The most commonly used visual cues are color, texture, shape, spatial information, and motion in video. For example, a feature may represent the color information in an image — such as color histograms, color binary sets [SC96a], color coherent vectors [PZM96], color correlograms [HKM$^+$97]), or description of texture[2]. The $n$ components of a feature may be derived from one

[2]One way is to use randomness, periodicity, and directionality [LP96], or to use wavelets [MM96], or to use coarseness, contrast, and directionality [FSN$^+$95]

visual cue or from composite cues, such as the combination of color and texture [BCGM98].

All the above visual cues, such as color or texture, are *low-level* image features. Objects, semantic categories or types of event depicted in images are *high-level* features. Regions or blobs generated as results of image segmentation, are called *middle-level* features in our work.[3]

There is a big gap between human perception and the low-level features. This gap limits the query scheme to be QBE which is not natural to human interaction with the image retrieval system. High-level features, which are perceived better, facilitate a more natural user interaction with the image retrieval system. For example, the query "show me a picture of a sunset" is more desirable than a query "show me a picture with $50\%$ of red color of a patch of $30 \times 30$, $30\%$ of orange color, and $10\%$ of dark color, and $\dots$" which some CBIR systems ask users to specify the percentages of color distribution. Even asking the users to draw a picture of sunset is not reasonable. Users might not be able to draw a picture of sunset, nor sketch a picture that they want.

High-level features are almost impossible to extract without human assistance, however. If we turn to object recognition and image segmentation to generate high-level or middle-level features, we would find that these traditional techniques cannot be applied directly to the generic data of our collection, for the images are arbitrary unstructured, unconstrained, and noisy. Thus, new approaches to these problems are warranted. Right now, most high-level features are generated manually [CPG$^+$97]. Low-level features can be generated automatically without human input, and hence are used in most image retrieval systems [FSN$^+$95, SC96c, Ma97, CBGM98, BFG$^+$96, CCWS$^+$97, CMOY96, DRD97, HKM$^+$97, PZM96, HKR93, KKS91, LP96, MM96, CS95, ORC$^+$97, PPS96, RS96, RGT97, SD96, Swai93].

A good low-level feature $f(\mathcal{I})$ for an image $\mathcal{I}$ should be designed to have certain qualities: (i) *perceptual similarity:* $|f(\mathcal{I}) - f(\mathcal{I}')|$ should be large if and only if $\mathcal{I}$ and $\mathcal{I}'$ are not "similar" (here $|\cdot|$ is a distance function) (ii) *efficiency:* $f(\cdot)$ should be fast to compute, and (iii) *economy:* $f(\mathcal{I})$ should be small in size. The size of the feature not only affects the efficiency of retrieval, but also affects the design of indexing data structures, such as multi-dimensional indexing schemes.

Perceptual similarity determines the effectiveness of the feature for the purpose of retrieval. This is hard to achieve using just low-level features. For example, the following two pairs of images (figure 1.5) are the same scenes taken from different point of views with camera zoom, or at different time. The color histogram method fails to label the two images similar (rather the second images are ranked $111335$-th and $38466$-th similar to the first images by color histogram) in a database of more than $200,000$ images (actually the Web has more than this number of images).

It is certainly desirable and important that an image feature be *stable* to tolerate such large image appearance changes. In addition, the performance of an image feature should be as much as possible insensitive to the size of the image database. In other words, the ranks of the "right" images should not change much as the size of the image database grows bigger and bigger — this is called *scalability*. Some features have good retrieval performance when the size of the image collection is small (for example several thousands of images). When the size of the collection gets bigger, the features are more prone to false matches, therefore reduce the performance of image retrieval. Such an example is the color histogram (see Figure 1.6). We will discuss the scalability in detail later in Section 3.2.

[3]There are other different categories for image features [Gong98, CSBB97].

### 1.2.2 Feature Comparison

Given two features, a *distance function* computes the difference of the two vectors that hopefully measures the *similarity* of visual appearance between two given images. The greater the distance, the less the similarity. Since features are treated as vectors, the distance function is often defined as Euclidean norm, city-block metric (or $L_1$ norm), or weighted Euclidean norm [HSE95] (the angular distance is a popular distance function in document retrieval, but it is seldom used in image retrieval).

Sometimes a feature can be treated as a distribution of a random variable in the image, such as color in color histograms. The distance function should be designed to measure the difference of two distributions. One commonly used function for computing the difference of two distributions is the Mahalanobis distance [DH73]. The other is a divergence measure based on Kullback's minimum cross-entropy principle [PHB97, OPH96]. Recently, [RGT97] proposed an "earth mover's distance" which computes the work load required to transforming one distribution to another distribution.

It seems that the distance functions should be a metric. Research in computer vision and cognitive science however suggests that human perceptual judgments about visual similarity are inherently non-metric, i.e., they may not obey the triangle inequality. Some non-metric similarity measures are tested for image classification [JWG98].

## 1.3 Contributions

In this work, we propose a new color feature for image indexing/retrieval called the *color correlogram* and show that it can be effectively used in the various image and video processing tasks described in 1.1 [HKM$^+$97, HKM97, HKMZ98, HKZ98]. The highlights of this feature are: (i) it describes the global distribution of local spatial correlations of colors, (ii) it is easy to compute, and (iii) the size of the feature is fairly small, (iv) it is stable to tolerate large appearance changes, and (v) it is also scalable to large image databases. Experimental evidence, tested on an image database of over $200,000$ images, shows that this new feature (i) outperforms both the traditional histogram method and the very recently proposed histogram refinement method [PZ96] for image indexing/retrieval, and (ii) outperforms the histogram-based approaches for the other video browsing tasks listed above.

Informally, a correlogram is a table indexed by color pairs and distance, where the $k$-th entry for $\langle i, j\rangle$ specifies the probability of finding a pixel of color $j$ at a distance $k$ from a pixel of color $i$. Such an image feature turns out to be stable in tolerating large changes in appearance of the same scene caused by changes in viewing positions, changes in the background scene, partial occlusions, camera zoom that causes radical changes in shape, etc. (see Figure 3.2 and Figure 3.3). We provide efficient algorithms to compute the correlogram.

We also propose a different distance metric to compare feature vectors. The $L_1$ distance metric, used commonly to compare vectors, considers the absolute component-wise differences between vectors. The *relative distance metric* we use calculates relative differences instead and in most cases performs better than the absolute metric (the improvement is significant especially for histogram-based methods). In addition, we study empirically other distance measures, such as the divergence measure, one non-metric measure proposed by [JWG98] to compare the effectiveness of these distance functions for content-based image retrieval.

To further improve retrieval results by dynamic querying, we address how relevance feedback can be used to improve the performance of correlograms for content-based image retrieval. We present two supervised learning methods (in the form of labeled examples): *learning the query* and *learning the metric*. The first scheme is based on the spatial locality of feature vectors corresponding to similar images. Learning is effected by modifying the query vector to incorporate the positive examples. The second scheme is based on "distorting" our view of the feature space. An new similarity distance between an image and the query is learned from the relevance feedback. By using a weighted metric, it is possible to shorten the distance between the query and relevant images and elongate the distance between the query and irrelevant images. Our results on a large image database of over $20,000$ images suggest that these learning methods are quite effective for content-based image retrieval.

We then investigate the applicability of correlograms other tasks such as image subregion querying, object localization, cut detection, and image classification. We propose the *correlogram intersection* method for the image subregion querying problem and show that this approach yields significantly better results than the traditional histogram intersection method. The histogram-backprojection approach used for the localization problem in [SB91] has serious drawbacks. We discuss these disadvantages and introduce the idea of *correlogram correction*. We show that it is possible to locate objects in images more accurately by using local color spatial information in addition to histogram backprojection.

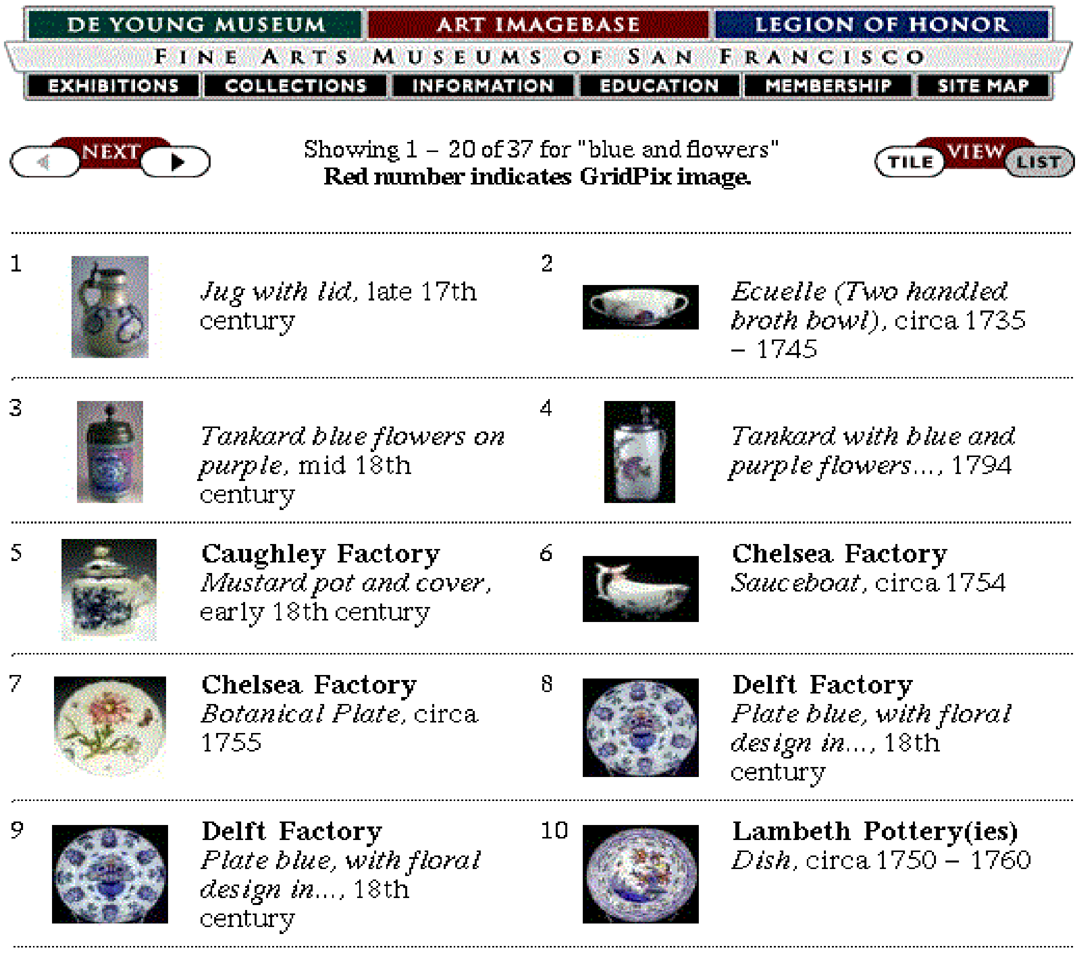
To help video parsing and browsing, we use correlograms to compare video frames and detect cuts by looking for adjacent frames that are very different. Once again we show that using the correlogram as the feature vector yields superior results compared to using histograms.

We also propose a new method for hierarchical classification of images via supervised learning. This scheme uses (banded) correlograms and then performs feature-space reconfiguration using singular value decomposition to reduce noise and dimensionality. We use the training data to obtain a hierarchical classification tree that can be used to categorize new images. Our experimental results suggest that this scheme not only performs better than standard nearest-neighbor techniques and color histograms, but also has both storage and computational advantages.

Our preliminary results thus indicate that the correlogram method is a more accurate and effective approach to these tasks compared to the color histogram method. What is more, the computational cost of the correlogram method is about the same as that of other simpler approaches, such as the histogram method. We believe, therefore, that correlograms form a significantly superior but equally economical alternative to the color histogram.

## 1.4 Organization

In Chapter 2, we propose the image feature *color correlograms* for the *color-spatial indexing*. We define the color correlogram and show how to compute it efficiently. Chapter 3 discusses content-based image retrieval and the use of the correlogram for this problem. Chapter 6 describes how to use relevance feedback combined with color correlograms to improve the results of content-based image retrieval. Chapter 4 discusses the the use of the correlogram for image subregion querying. Applications of the correlogram to video browsing problems are described in Chapter 5. We also propose a new scheme for image classification which makes use of the properties of correlograms in Chapter 7. Finally, Chapter 8 concludes with some remarks and scope for further work.



Showing 1 – 20 of 37 for "blue and flowers"
**Red number indicates GridPix image.**

1 *Jug with lid*, late 17th century

2 *Ecuelle (Two handled broth bowl)*, circa 1735 – 1745

3 *Tankard blue flowers on purple*, mid 18th century

4 *Tankard with blue and purple flowers...*, 1794

5 **Caughley Factory** *Mustard pot and cover*, early 18th century

6 **Chelsea Factory** *Sauceboat*, circa 1754

7 **Chelsea Factory** *Botanical Plate*, circa 1755

8 **Delft Factory** *Plate blue, with floral design in...*, 18th century

9 **Delft Factory** *Plate blue, with floral design in...*, 18th century

10 **Lambeth Pottery(ies)** *Dish*, circa 1750 – 1760

Figure 1.1: Top ten images retrieved to the query “blue flowers”.

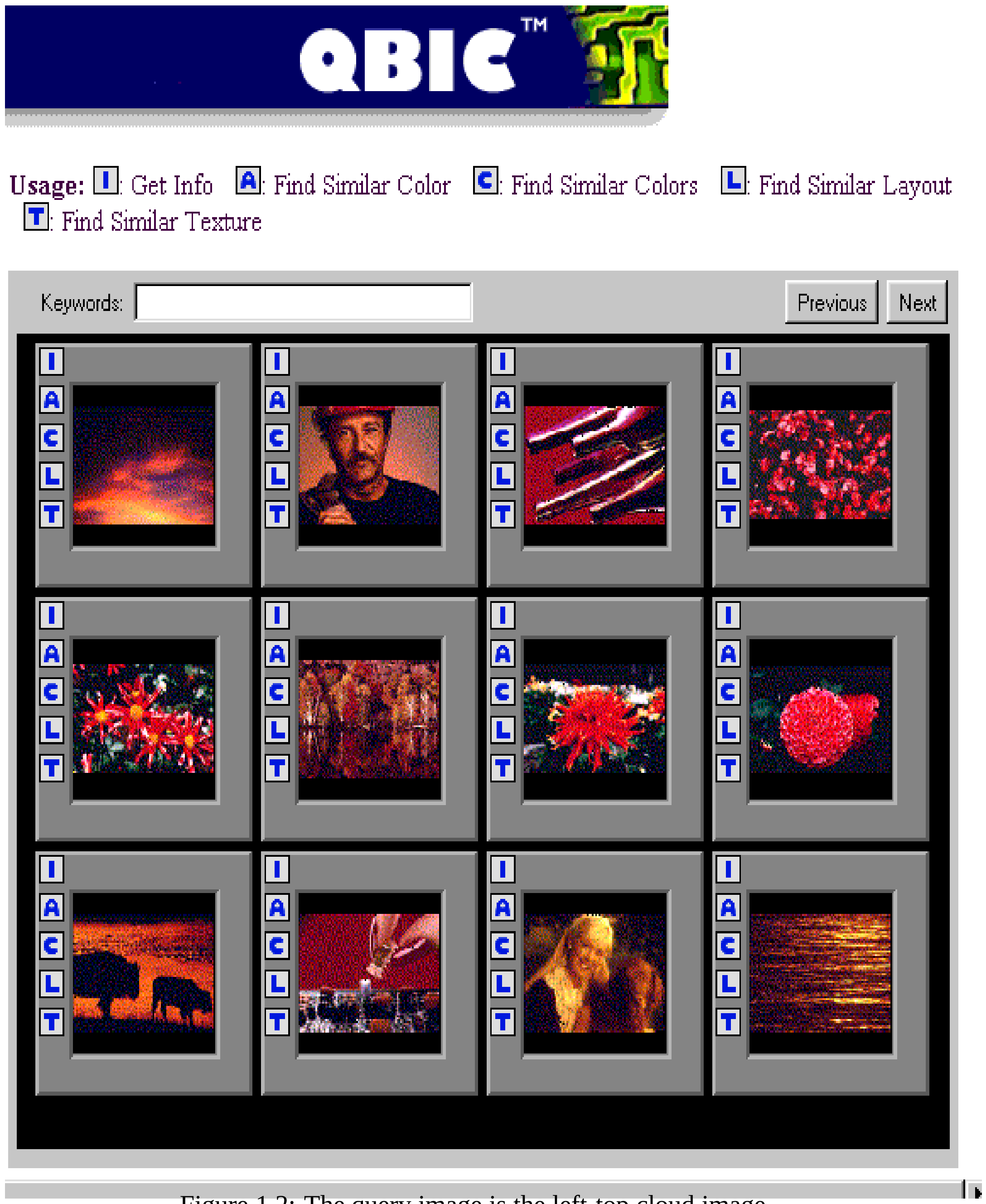
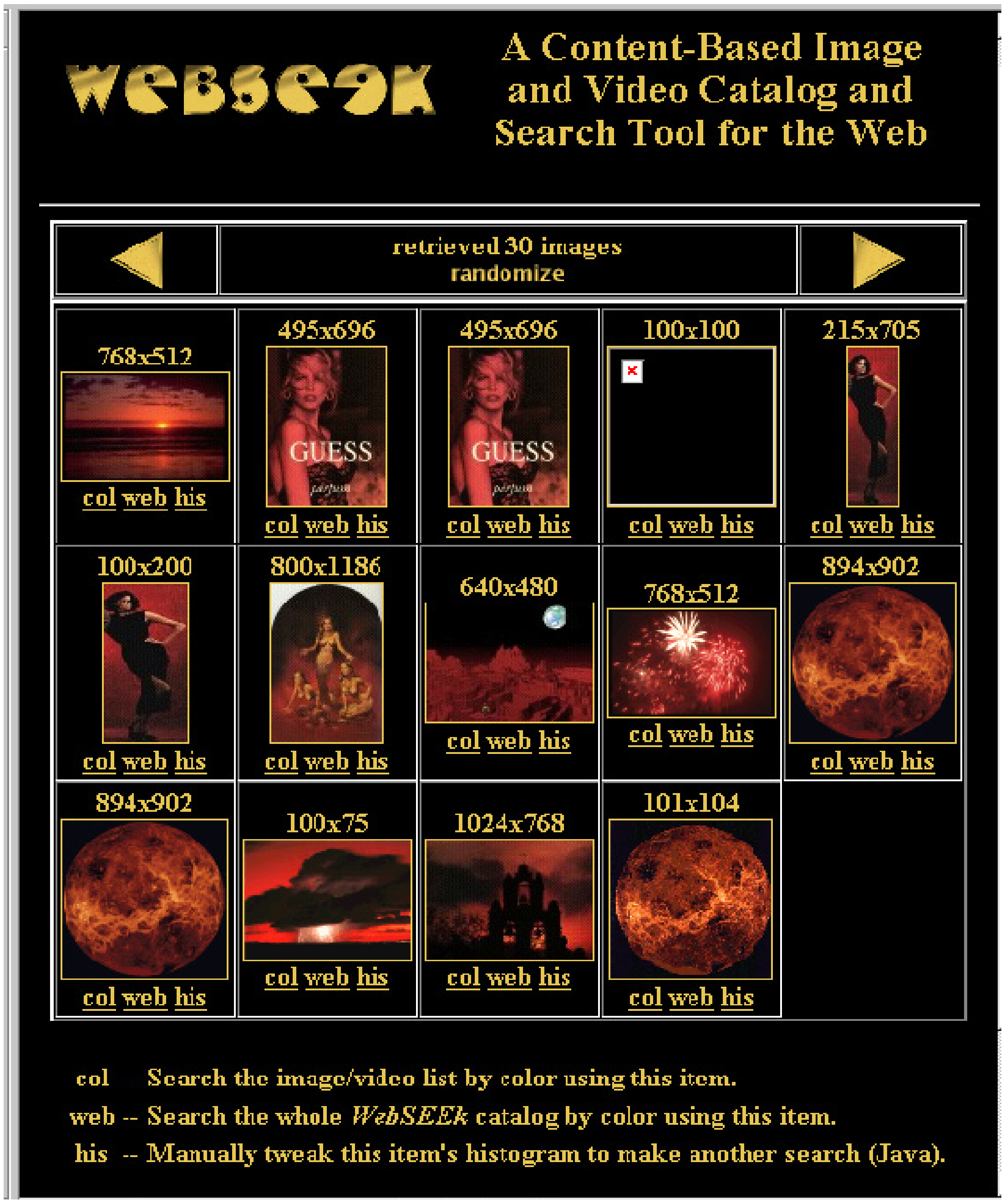

QBIC™
Usage: I: Get Info A: Find Similar Color C: Find Similar Colors L: Find Similar Layout T: Find Similar Texture
Keywords:
Previous
Next
I
A
C
L
T

Figure 1.2: The query image is the left-top cloud image.



col Search the image/video list by color using this item.
web -- Search the whole *WebSEEk* catalog by color using this item.
his -- Manually tweak this item's histogram to make another search (Java).

Figure 1.3: The query image is the left-top sunset image.

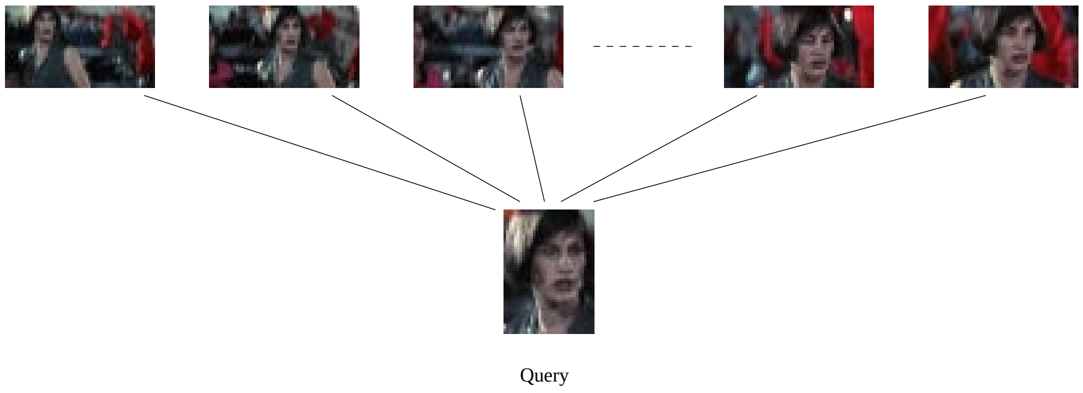


Figure 1.4: Image subregion querying.

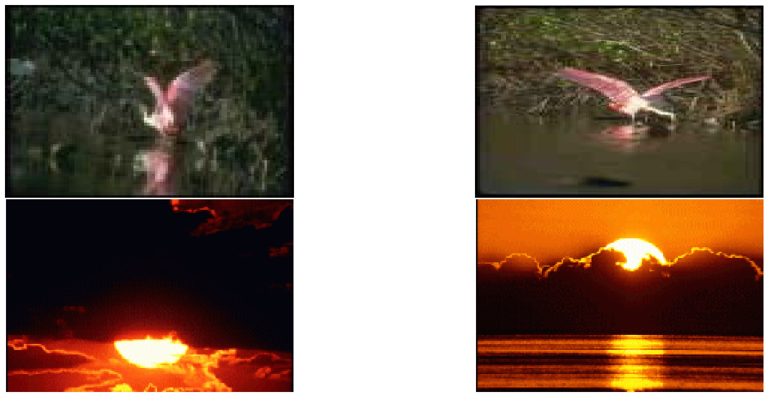
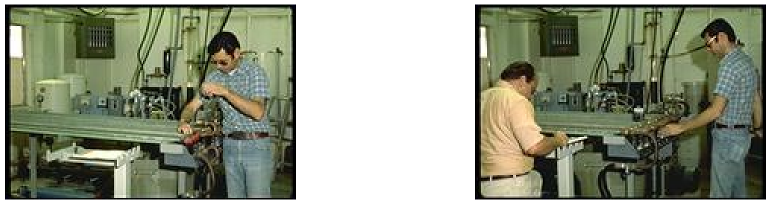
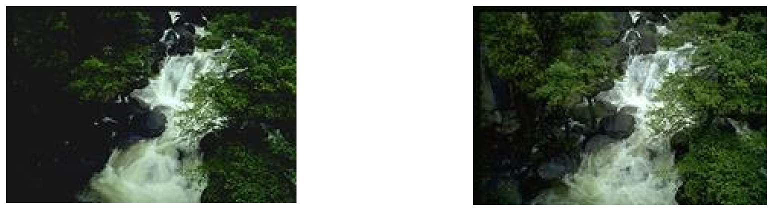


Figure 1.5: Large image appearance change.



The rank of the second image changes from rank 38 to rank 308.



The rank of the second image changes from rank 54 to rank 3024.

Figure 1.6: Color histogram may not be *scalable* when the number of images increases from $18,000$ to $200,000$.

# Chapter 2

# Color-Spatial Image Indexing

Indexing an image database is analogous to cataloging a library. Several different techniques accomplish this task with varying degrees of success.

A common approach in the database realm is to first assume a *a priori* definition of regions and objects of images and then perform *spatial indexing*, which is an indexing technique that exploits the absolute or relative spatial information of these objects inside images. Images can then be compared using this spatial information [Smit97]. Notice that this technique does not incorporate image features such as color, texture, etc. and it is also difficult to extend the scheme to incorporate these features [SS96]. Moreover, the assumption about predefined objects is unrealistic from a practical point of view.

To alleviate some of these problems, most image retrieval systems use image features for indexing (see Chapter 1). For instance, a popular scheme is to use color histograms for *color indexing*. This has turned out to be quite successful [SB91, FSN$^+$95, CS95, PPS96, BFG$^+$96, SC96c, Swai93] because color is an important cue in human visual perception.

The color histogram, however, does not include any spatial layout information, which is also an equally important cue in human visual perception. To handle this major drawback of color histograms, several new features that strive to integrate both color and spatial information have been proposed [DRD97, HCP95, LGS97, PZM96, RS96, SC96a, SD96]. Our work proposes a novel scheme that makes use of the spatial correlation between color pairs which in turn aids in achieving a seamless integration of both color and spatial information.

## 2.1 The Color Correlogram

"Correlograms are graphs (or tables) showing how autocorrelation changes with distance." [UF85]. Traditionally the distance meant the time distance between pairs of observations. Spatial analysts adapted the idea to spatial distance, and we adapt the idea to spatial distance of color pixels in an image.

### 2.1.1 Definition

A color correlogram (henceforth correlogram) expresses how the spatial correlation of color changes with distance. A color histogram (henceforth histogram) captures only the color distribution in an image and does not include any spatial information. Thus, the correlogram is one kind of spatial extension of the histogram.

**Notation.** Let $\mathcal{I}$ be an $n \times n$ image. (For simplicity of exposition, we assume that the image is square.) The colors in $\mathcal{I}$ are quantized into $m$ colors $c_1, \ldots, c_m$. (In practice, $m$ is deemed to be a constant and hence we drop it from our running time analysis.)

For a pixel $p = (x, y) \in \mathcal{I}$, let $\mathcal{C}(p)$ denote its color. Let $\mathcal{I}_c \triangleq \{p \mid \mathcal{C}(p) = c\}$. Thus, the notation $p \in \mathcal{I}_c$ is synonymous with $p \in \mathcal{I}, \mathcal{C}(p) = c$. For convenience, we use the $L_\infty$-norm to measure the distance between pixels, i.e., for pixels $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$, we define $|p_1 - p_2| \triangleq \max\{|x_1 - x_2|, |y_1 - y_2|\}$ (thus the eight neighbors of $p$ are distance one from it). We denote the set $\{1, 2, \ldots, n\}$ by $[n]$.

**Definitions.** The *histogram* $h$ of $\mathcal{I}$ is defined as follows:
for a color $c_i$, $i \in [m]$

$$H_{c_i}(\mathcal{I}) \triangleq \|\mathcal{I}_{c_i}\| \tag{2.1}$$

i.e., the number of pixels of color $c_i$ in $\mathcal{I}$. Randomly taking any pixel $p$ from the image $\mathcal{I}$, the probability that the color of $p$ is $c_i$ is

$$h_{c_i}(\mathcal{I}) = \Pr[p \in \mathcal{I}_{c_i}] = \frac{H_{c_i}(\mathcal{I})}{n^2} \tag{2.2}$$

The histogram is easily computed in $O(n^2)$ time, which is *linear* in the size of $\mathcal{I}$. Note that the above definition of the histogram is as a probability distribution $h$ (see Appendix A). Some authors prefer to define histograms purely in terms of the count $H$, thus making it depend on the image size. Our definition of the histogram is size-invariant, which is more appropriate for varying-sized images.

In the light of viewing the histogram as a probability distribution of colors, we consider this question: pick any pixel $p_1$ of color $c_i$ in the image $\mathcal{I}$, at distance $k$ away from $p_1$ pick another pixel $p_2$, what is the probability that $p_2$ is of color $c_j$? This gives us the conditional probability distribution that depicts the spatial correlation between pixels (see Appendix A). We define the correlogram formally below.

Let a distance $d \in [n]$ be fixed *a priori*. Then, the *correlogram* of $\mathcal{I}$ for $i, j \in [m]$, and $k \in [d]$, is defined to be

$$\gamma^{(k)}_{c_i, c_j}(\mathcal{I}) \triangleq \Pr[p_2 \in \mathcal{I}_{c_j}, |p_1 - p_2| = k \mid p_1 \in \mathcal{I}_{c_i}]. \tag{2.3}$$

Therefore the correlogram of an image $\mathcal{I}$ is a table indexed by color pairs and distance, where the $k$-th entry for $\langle i, j \rangle$ specifies the probability of finding a pixel of color $j$ at a distance $k$ from a pixel of color $i$. Note that the size of the correlogram is $O(m^2 d)$.

If we only consider the correlation between the identical colors, we have the *autocorrelogram* of $\mathcal{I}$, which is defined by

$$\alpha^{(k)}_c(\mathcal{I}) \triangleq \gamma^{(k)}_{c,c}(\mathcal{I}). \tag{2.4}$$

This information is a subset of the correlogram and requires only $O(md)$ space.

While choosing $d$ to define the correlogram, we need to address the following trade-off. A large $d$ would result in expensive computation and large storage requirements. A small $d$ might compromise the quality of the feature. Thus $d$ has to be chosen according the application that the correlogram is used for. We will return to this problem when we apply the correlogram to different problems.

**Example.** Consider the simple case when $m = 2$ and $n = 8$. Two sample images are shown in 2.1. The two images have exactly the same histogram. The autocorrelograms corresponding to these two images are shown in 2.2. The change of autocorrelation of the foreground color (yellow) with distance is perceptibly different for these images: in the second image, the correlation drops to zero at distance $2$, and then reach a peak at distance $4$; in the first image, the correlation monotonically drops to zero at distance $4$. This exactly shows the difference between the long rectangle in the first image and the two separated squares in the second image.

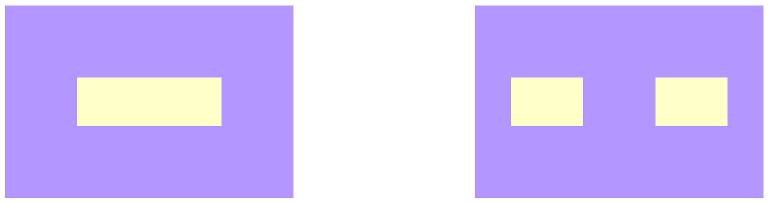


Figure 2.1: Sample images: image 1, image 2.

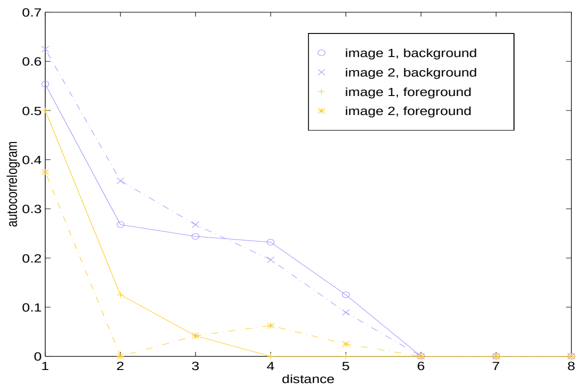


Figure 2.2: Autocorrelograms for two images in Figure 2.1.

**Distance Measure.** Since the correlogram is another feature vector for an image, we can compare them via the $L_1$ and $L_2$ distance measures which are commonly used when comparing two feature vectors. In practice, the $L_1$ distance measure performs better than the $L_2$ distance measure because the former is statistically more robust to outliers[1] [RL87].

We will discuss the usability of a distance measure in detail in the next chapter (see 3.3). If we use the $L_1$ measure both for histograms and correlograms, the following formulae can be used to compute the distance between images $\mathcal{I}$ and $\mathcal{I}'$:

$$|\mathcal{I} - \mathcal{I}'|_{h,L_1} \triangleq \sum_{i\in[m]} |h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')| \tag{2.5}$$

$$|\mathcal{I} - \mathcal{I}'|_{\gamma,L_1} \triangleq \sum_{i,j\in[m],k\in[d]} |\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) - \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')| \tag{2.6}$$

In the following sections, we look at some efficient algorithms to compute the correlogram. Our algorithms are amenable to easy parallelization. Thus, the computation of the correlogram could be easily speeded up. We also provide some extensions to the correlogram.

[1]Since in this case the $x$-axis represents the quantized color bins and outliers only appear in $y$-axis.

## 2.2 Computation

First, to compute the correlogram, it suffices to compute the following count, which is similar to the co-occurrence matrix [Hara79] for texture analysis of gray images.

If

$$\Gamma^{(k)}_{c_i,c_j}(\mathcal{I}) \triangleq \left|\{p_1 \in \mathcal{I}_{c_i}, p_2 \in \mathcal{I}_{c_j}, |p_1 - p_2| = k\}\right|, \tag{2.7}$$

then,

$$\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) = \frac{\Gamma^{(k)}_{c_i,c_j}(\mathcal{I})}{h_{c_i}(\mathcal{I}) \cdot 8k} \tag{2.8}$$

The denominator is the total number of pixels at distance $k$ from any pixel of color $c_i$. The factor $8k$ is due to the eight neighbors which are $k$ distance (in $L_\infty$-norm) away from the center pixel. As the histogram is the color distribution in $\mathcal{I}$, the correlogram is the color correlation distribution in $\mathcal{I}$. Therefore both qualify as global features.

The naive algorithm would be to consider each $p_1 \in \mathcal{I}$ of color $c_i$ and for each $k \in [d]$, count all $p_2 \in \mathcal{I}$ of color $c_j$ with $|p_1 - p_2| = k$. Unfortunately, this takes $O(n^2d^2)$ time. To obviate this expensive computation, we define the quantities

$$\lambda^{c,h}_{(x,y)}(k) \triangleq |\{(x+i, y) \in \mathcal{I}_c \mid 0 \leq i \leq k\}| \tag{2.9}$$

$$\lambda^{c,v}_{(x,y)}(k) \triangleq |\{(x, y+j) \in \mathcal{I}_c \mid 0 \leq j \leq k\}| \tag{2.10}$$

which count the number of pixels of a given color within a given distance from a fixed pixel in the positive horizontal/vertical directions.

Our algorithms work by first computing $\lambda^{c_j,v}_p$ and $\lambda^{c_j,h}_p$. In the following section we show that when $d$ is a constant the running time is $O(n^2d)$, which is also linear of the size of the image.

### 2.2.1 Dynamic Programming

Let a pixel $p = (x, y)$. The following equation is easy to check

$$\lambda^{c,h}_{(x,y)}(k) = \lambda^{c,h}_{(x,y)}(k-1) + \lambda^{c,h}_{(x+k,y)}(0) \tag{2.11}$$

with the initial condition

$$\lambda^{c,h}_p(0) = 1 \text{ if } p \in \mathcal{I}_c \text{ and 0 otherwise.}$$

Now, $\lambda^{c,h}_p(k)$ is computed for all $p \in \mathcal{I}$ and for each $k = 1, \ldots, d$ using Equation 2.11. The correctness of this algorithm is obvious. Since we do $O(n^2)$ work for each $k$, the total time taken is $O(n^2d)$.

In a similar manner, $\lambda^{c,v}_p$ can also be computed efficiently. Now, modulo boundaries, we have (see Figure 2.3)

$$\begin{aligned}\Gamma^{(k)}_{c_i,c_j}(\mathcal{I}) &= \sum_{(x,y)\in\mathcal{I}_{c_i}} \Big(\lambda^{c_j,h}_{(x-k,y+k)}(2k) + \lambda^{c_j,h}_{(x-k,y-k)}(2k) \\ &+ \lambda^{c_j,v}_{(x-k,y-k+1)}(2k-2) + \lambda^{c_j,v}_{(x+k,y-k+1)}(2k-2)\Big)\end{aligned}$$

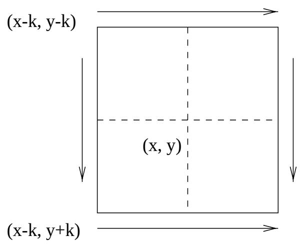


Figure 2.3: For a pixel $(x, y)$ of color $c_i$.

I.e., for each pixel $(x, y)$ of color $c_i$, we count the number of pixels of color $c_j$ on the four line segments of distance $k$ away from $(x, y)$: two horizontal lines starting from pixel $(x-k, y+k)$ and pixel $(x-k, y-k)$; two vertical lines starting from pixel $(x-k, y+k+1)$ and pixel $(x+k, y-k+1)$. The summation runs through the image once, and computation takes just $O(n^2)$ time.

The hidden constants in the overall running time of $O(n^2 d)$ are small and hence this algorithm is *linear* of the size of the image when $d$ is a constant.

## 2.3 Extensions

In this section, we will look at some extensions to color correlograms. The general theme behind the extensions are: (1) improve the storage efficiency of the correlogram while not compromising its image discrimination capability, and (2) use additional information (such as edge information) to further refine the correlogram, boosting its image retrieval performance. These extensions can not only be used for the image retrieval problem, but also in other applications like cut-detection (see Chapter 4).

First, we describe our extension to correlograms which improves the storage efficiency. Next, we describe the combination of edge information with correlograms.

**Banded Correlograms.**

In Section 2.1.1, we saw that the correlogram (resp. autocorrelogram) takes $m^2 d$ (resp. $md$) space. It would be advantageous if the storage requirements were trimmed further. This leads to the definition of *banded correlogram* for a given $b$. (For simplicity, assume $b$ divides $d$.) For $1 \leq k \leq b$,

$$\overline{\gamma}^{(k)}_{c_i,c_j}(\mathcal{I}) \triangleq \sum_{k'=kb}^{(k+1)b-1} \gamma^{(k')}_{c_i,c_j}(\mathcal{I}). \tag{2.12}$$

In a similar manner, the banded autocorrelogram $\overline{\alpha}^{(k)}_{c_i}(\mathcal{I})$ can also be defined. The space requirements for the banded correlogram (resp. banded autocorrelogram) is $m^2 d/b$ (resp. $md/b$). (Note that when $b = d$, $\overline{\gamma}$ measures the *density* of a color $c_j$ near the color $c_i$, thus suggesting a local structure of colors.) The distance metrics defined in Equation 2.6 is easily extended to this case.

Note that banded correlograms are seemingly more susceptible to false matches since

$$|\mathcal{I} - \mathcal{I}'|_{\overline{\gamma},L_1} \leq |\mathcal{I} - \mathcal{I}'|_{\gamma,L_1},$$

which follows by triangle inequality. Although the banded correlograms have less detailed information as correlograms, our results show that the approximation of $\gamma$ by $\overline{\gamma}$ has only negligible effect on the quality of the image retrieval problem and other applications.

**Edge Correlogram.** We can also combine the color information with geometric features, such as edges. Suppose $\mathcal{E} : \mathcal{I} \rightarrow \{0, 1\}$ is the edge information of image $\mathcal{I}$, i.e., $\mathcal{E}(p) = 1$ if $p$ is on an edge and 0 otherwise. (Such information can be obtained using various edge-detection algorithms.) Now, the question is whether this information can be combined with (auto)correlograms so as to improve the discriminative power of correlograms. We outline one scheme to do this[2]. In this scheme, each of the $m$ color bins is refined to bucket $\mathcal{I}$ with $2m$ bins.

$$\mathcal{C}(p) = \begin{cases} c_+ & \text{if } \mathcal{C}(p) = c \text{ and } \mathcal{E}(p) = 1 \\ c_- & \text{if } \mathcal{C}(p) = c \text{ and } \mathcal{E}(p) = 0 \end{cases} \tag{2.13}$$

It is easy to see that the definition of both correlograms and autocorrelograms directly extend to this case. The storage requirements become $4m^2d$ (resp. $2md$) for correlograms (resp. autocorrelograms). Note however that the number of $p$ such that $\mathcal{E}(p) = 1$ is usually very small. Since we mostly deal with autocorrelograms, the statistical importance of $\alpha^{(k)}_{c_+}$ becomes insignificant, thus rendering the whole operation meaningless. A solution to this problem is to define *edge autocorrelogram* in which cross correlations between $c_+$ and $c_-$ are also included. The size of edge autocorrelogram is thus only $4md$. We can further trim the storage by the banding technique.

## 2.4 Summary

In this chapter, we have introduced a new image feature called *color correlogram*. The idea is to exploit *spatial correlation* between colors and examine the change of this correlation with distance. We suggest that color correlogram has certain discrimination ability. We show that how to compute color correlogram efficiently and how to extend the idea of spatial correlation to include other information into color correlogram. We also propose the banding technique to trim the size of correlograms.

In the following chapters, we discuss how to use color correlograms effectively and efficiently for content based image retrieval, image subregion querying and localization, video cut detection, supervised learning in content-based image retrieval feedback, and image classification.

[2]Instead of counting pixels exactly on the edges, we can count pixels that are within some distance from edges.

# Chapter 3

# Content-Based Image Retrieval

The problem of content-based image retrieval (CBIR) needs no motivation (see Chapter 1). The success of a solution to this problem crucially depends on (i) the stability and scalability of the image features used and (ii) the characteristics of the distance function used for comparing the image features.

In this chapter, we investigate the use of color correlograms as image feature vectors. We also investigate different distance measures such as the city-block metric ($L_1$ norm), a new metric called the *relative distance* measure, and some non-metric measures (the divergence measure and $L_{\frac{1}{2}}$ function). Our experimental evidence shows that the color correlogram is both stable and scalable for image retrieval from large image collections. It outperforms both the traditional histogram method and the very recently proposed histogram refinement method [PZ96] without compromising efficiency.

**Organization.** Section 3.1 provides an account of different image features commonly used in CBIR systems. Section 3.2 discusses the stability and scalability issues. Section 3.4 gives a brief summary of related work that combine spatial information with colors. Section 3.3 discusses the distance measures and Section 3.5 describes the performance measures used. Some practical considerations that would improve efficiency are listed in Section 3.6. The experiments and results are presented in Section 3.7 and 3.8. Section 3.9 concludes the chapter.

## 3.1 Image Features for CBIR

Image features affect every aspect of a CBIR system, and so it is important to carefully choose the right image features for any CBIR system. Most of the CBIR systems explore low-level image features such as color, texture, shape, motion, etc. because they can be computed automatically. Middle-level features like regions and blobs which can be generated without human assistance are used in object-level image retrieval. We discuss the characteristics of some of these features, focusing mainly on how they are extracted and compared. The CBIR systems we discuss will be query-by-example (QBE) style, unless otherwise specified.

### 3.1.1 Color

Color is one of the most prominent perceptual features. Most commercial CBIR systems include color as one of the features (e.g., QBIC of IBM, Virage, etc.).

The easy-to-compute color histogram is a popular and widely used image feature . To use color histograms for image retrieval, there are three basic steps: (i) *Color space quantization* — for a choice of color space $\mathcal{C}$ (e.g., RGB, HSV, LXY [Lee92], CIE, Munsell etc.), the division of $\mathcal{C}$ into $m$ levels, $c_1, c_2, \ldots, c_m$; (ii) *Histogram binning* — a linear-time scan of the image to count the number of color pixels in each color bin; and (iii) *Histogram matching* — choosing a distance function $\mathcal{D}$ to measure the similarity of two histograms. The choice of the color space $\mathcal{C}$, the different ways of dividing the color space (say, uniformly or non-uniformly), and the choice of the distance function $\mathcal{D}$, determine the performance of a CBIR system that uses color histograms. We briefly discuss two CBIR systems based on these parameter choices.

**FINDIT.** The parameters in FINDIT, developed by Swain *et. al.* [Swai93] are (i) HVC (Hue-Value-Chroma) color space, (ii) histogramming Hue-Chroma, and (iii) the *histogram intersection* distance function $\mathcal{D}$:

$$\mathcal{D}(h(I), h(Q)) = \frac{\sum_{j=1}^{m} \min\left(h(I)_j, h(Q)_j\right)}{\sum_{j=1}^{n} h(Q)_j} \tag{3.1}$$

where $h(I)_j, h(Q)_j$ are color counts of color $j$ in image $\mathcal{I}$ and $\mathcal{Q}$. Note that this definition is *not* symmetric in $\mathcal{I}$ and $\mathcal{Q}$; thus $\mathcal{D}$ is not a metric.

To retrieve answers for one query, the above distance function takes $O(mN)$ time, where $m$ is the number of colors and $N$ is the number of images in the database. To save this expensive on-line computation, only prominent colors can be used in the summation of Equation 3.1. This *decomposing* step [Gong98] reduces the comparison to $O(m \log m + BN)$ if $B$ largest number of color used. If $B = o(m)$, then the retrieval could be faster, at the expense of some false positives.

**QBIC.** IBM's QBIC is a CBIR system that exploits multiple image features including color histogram, texture, and shape. It allows QBE as well as drawing a picture. QBIC uses color histogram in the following way: (i) RGB color space, (ii) non-uniform partitioning into 256 clusters based on the perceptual distance of colors contained in the RGB space, and (iii) a quadratic distance function to compare histograms:

$$\mathcal{D}^2(h(I), h(Q)) = \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}(h(I)_i - h(Q)_i)^2 = (h(I) - h(Q))^T A(h(I) - h(Q)). \tag{3.2}$$

The matrix $A$ indicates the quantities of perceptual similarity between two colors. Therefore $\mathcal{D}^2$ not only compares the corresponding color bins, but also compares different color bins. For instance, this measure can be made to indicate that orange is 'more similar to' red than to blue. Since $\mathcal{D}^2$ involves the whole matrix-vector multiplication, it is not decomposable as histogram intersection. The computation of one query takes $O(m^2N)$, which is not efficient.

In summary, the color histogram is suitable for retrieving images based on overall color impression. Since it does not include any spatial information. it has limited image discriminative power.

### 3.1.2 Texture

Texture is another important visual cue that has been intensively studied in pattern recognition. Texture perception studies [RL93] indicate that humans discriminate natural texture patterns in three important dimensions — repetitiveness, directionality, and granularity. We have to address two issues when we use texture for image retrieval: (i) a texture model that captures human perception of texture; and (ii) a distance function that measures the similarity between texture patterns. In the following we examine several texture-based CBIR systems.

Liu and Picard [LP96] propose "periodicity", "directionality", and "randomness" for texture modeling and retrieval. These features approximate the three important dimensions of human texture perception mentioned earlier. Their approach extracts two sets of texture feature vectors for the structured texture pattern and the random texture pattern, with a posterior probability value indicating the confidence level. The distance function is designed to take into account both the structured part and the random part, weighted by the confidence values. This probabilistic measure of similarity is less sensitive to threshold parameters that used for extracting texture features. The tests on the Brodatz texture database suggest a perceptually more satisfying retrieval.

The QBIC system [ABF$^+$95] uses another set of perceptual texture descriptors: "coarseness", "contrast", and "directionality". Assume these three factors are uncorrelated, a weighted $L_2$ distance function is used. The weights are variances of the coarseness measure, the contrast measure, and the directionality measure over all the images in the database. Therefore these weights should be updated if the database changes reasonably.

Manjunath and Ma [MM96] use Gabor wavelet representation of texture, which is not directly related to the human perception of texture. The Gabor filters are designed to detect orientations, scale tunable edges, and lines. Each feature vector consists of pairs of mean and variance of different orientation and scale combination. Assuming that the local texture regions are spatially homogeneous, the distance of two feature vectors are $L_1$ differences normalized by their corresponding variances.

All these texture models have very good performance on the standard Brodatz test data. It is not clear, however, if these models can deal with non-homogeneous regions that occur in natural scenes.

### 3.1.3 Edge/Shape

The shape of objects/regions is yet another important low-level image feature for image retrieval. Edge detection, and sometimes segmentation, are used to extract shape features. There are many different types of shape analysis: statistical moment-based approach [Tego94], Hausdorff-distance based approach [HKR93], polygonal approximations [JV96, MG 1993], and contour-oriented representations [Ma97, ORC$^+$97, KKS91]. Because of the problems of lighting effects and occlusion, shape characterization techniques are preferred to local shape segmentation methods [BFG$^+$96]. In the following, we briefly discuss several contour-oriented approaches.

ART MUSEUM [KKS91] is a CBIR system designed for artistic color paintings. A query is given by drawing a rough sketch of the desired painting. The contour binary images are extracted by some edge-detection procedures and normalized to have the same length. Two contour images are matched by a block-block correlation function since they are binary values. This system is suitable for trademark images which have clear shape boundaries.

NETRA [Ma97] is another CBIR system that also exploits shape features. Three types of Fourier-based shape descriptors are used: curvature, centroid distance, and complex coordinate. These shape features are invariant to rotation, scaling, and translation. Two shape feature vectors are compared using $L_2$ metric. MARS [ORC$^+$97] also uses a modified Fourier descriptor to store the boundary of an object.

Shape features based on some shape descriptors have limited discriminating capability. One problem with shape descriptors is that it is hard to find a good perceptual measurement of similar shapes. For example, similar moments do not guarantee similar shapes. In summary, shape is an important visual cue that can be combined with color to improve the performance of image retrieval [JV96].

### 3.1.4 Motion

Motion is the most important cue in video sequences. It is mostly used for object tracking. Recently in content-based video browsing and searching, motion has been used for video classification (based on global motion estimation) [CCWS$^+$97, KD97] and improved browsing and indexing using mosaic images [SAG95, IAH95]. We will briefly discuss VideoQ system [CCWS$^+$97] since it is the only one that uses motion for content based video retrieval.

VideoQ allows a user to sketch a motion trail represented by a sequence of 2-D points and a time $t$. Motion trails are matched either by spatial information only (using $L_2$ distance of points) or spatial-temporal information (i.e., matching frame by frame). This query-by-motion-trail is especially useful when browsing and searching sports videos.

### 3.1.5 Regions

Since low-level image features have weak connection to semantic content of images, some CBIR systems are designed to achieve region (or object) level image retrieval [ABF$^+$95, CBGM97, Gong98, Ma97]. Exploiting image regions gives more explicit information of the image structure and configuration. It also facilitates *image subregion querying.* In the following we describe some recent work in object-based methods.

In [CBGM97], regions (or blobs) are obtained by the Expectation-Maximization (EM) algorithm performed on joint color and texture feature space. Each region has a feature vector which contains two dominant colors in HSV color space, mean texture descriptors (i.e. anisotropy, orientation, and contrast), and spatial descriptors (centroid and the scatter matrix which contains the shape information). Regions are compared by using the Mahalanobis distance between feature vectors of the corresponding region, normalized by an exponential function.

In [Gong98], regions are obtained by color image segmentation which is specially designed to handle uneven illumination conditions, such as shade, highlighting, etc. HVC color space is used and the linear C-V correlation is explored to prevent over-segmentation. Color (three H, S, V values) and shape features (location, size, circularity, eccentricity, orientation, and region profile) of each region serve as indexing features. $L_1$ distance metric is used to compare feature vectors of two regions.

In [Ma97], homogeneous regions are segmented by a boundary detection procedure which exploits edge flow of both color and texture [MM97]. Color, texture, and shape features of each region are indexed separately. When a query consists of more than one of these features, results coming from individual features are sorted by a weighted similarity measure. The parameters of the system are tuned differently for different categories of images.

To achieve object-level querying, most of the approaches use some sort of segmentation first. Segmenting an image into object regions is one of the oldest, yet most challenging problems in computer vision. Except for some narrowly defined problem domains where domain knowledge and *a priori* object models are available, accurate and complete segmentation on generic real-world scene is seldom available. Shade, highlight, and sharp contrast etc, which are very common in the natural scenery, are major challenges to image segmentation.

### 3.1.6 Summary of Image Features

In summary, each feature has its own pros and cons: color histogram is a global feature that is easily computable and used for overall color impression retrieval, but it is prone to false matches; texture descriptors are effective to retrieve texture patterns but are problematic with non-homogeneous regions in natural scenes; shape features can improve retrieval when used in conjunction with other features but have limited power by themselves; motion helps browsing and searching for video clips; and regions facilitate image subregion querying but are difficult to extract in natural scenes.

## 3.2 Stability and Scalability

Ideally the performance of retrieval should be insensitive to the size of the image database. The stability of a feature to tolerate large image appearance changes and the scalability of the feature for large image databases are two major challenges to the development of effective and efficient CBIR systems for large image databases and of web image search engines.

An image feature must be stable to tolerate large image appearance changes — changes in view positions, camera position, camera zoom, object occlusion, lighting, etc. Two pictures may be taken from the same scene at different time, but the objects in the scene may completely change their poses and positions. For example, consider the image pairs shown in Figure 3.1. Each pair shows images of the same scene, but the change in camera position is large thus resulting in different background and lighting conditions.

An image feature should also be able to handle any number of images. When the number of images is astronomical, there is always some probability that some arbitrary images are more similar to the query image than the relevant ones. This problem is worsened if there is a large appearance change between the query and the relevant images. Therefore, the image feature must scale well with the size of the databases. Unfortunately, this issue has not been addressed very well in general, except in [PZ98] where some preliminary results are available.

Unless image features are both stable and scalable, many irrelevant images will get ranked higher than relevant images. For example, color histogram is insensitive to small changes in viewing positions. Since it does not include any spatial information however, it is not stable to tolerate large image appearance changes and is therefore liable to cause false positive matches despite its seeming effectiveness. For instance, the pair of motorcycle images shown in Figure 3.1 (photographs of the same scene taken from different viewpoints) are likely to be labeled dissimilar by the naive histogram method. (In our database of $18,000$ images, the right motorcycle image is considered the $375$-rd most similar with respect to the left image by color histogram.) When databases are large, this problem is specially acute since false positives are even more likely to occur in such scenarios, i.e., it is likely that different images may have (almost) the same histograms (In our database of $200,000$ images, the rank of the right image of motorcycle drops from $375$ to $12,059$).

In order to measure the stability and scalability quantitatively, we propose the following approach. To measure stability, we manually choose pairs of images of the same scene. One of a pair is a chosen to be query (*unique answer query*) and the other is marked to be the unique answer to this query in the whole image database (see Figure 3.2). These kind of queries allows us to measure the stability objectively since different users may label different relevant images. The stability is measured by where the answer is ranked by a feature. The lower the rank is, the more

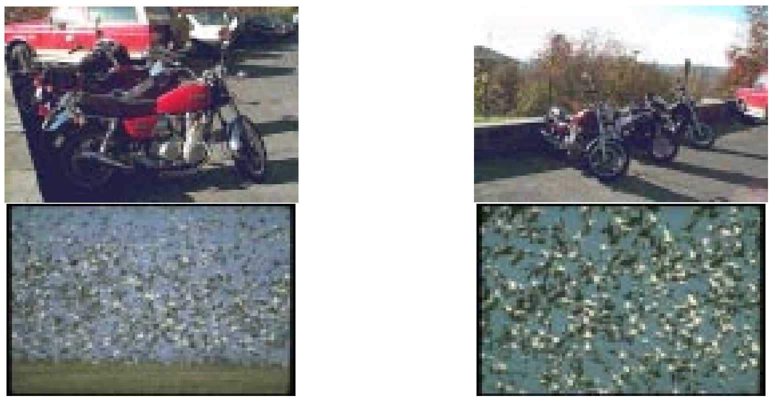


Figure 3.1: Large image appearance change.

stable the feature is (ideally the rank should be 1). To measure scalability we use the change in the relative performance measure (such as rank or precision) when the size of database is increased.

## 3.3 Distance Measures

Features alone cannot completely guarantee stability. The distance function used to compare features also plays an important role. An ideal distance function $\mathcal{D}$ and the feature $f(\mathcal{I})$ would satisfy the perceptual similarity:

$$\mathcal{D}(f(\mathcal{I}_1), f(\mathcal{I}_2)) \text{ is small} \Longleftrightarrow \mathcal{I}_1 \text{ and } \mathcal{I}_2 \text{ are perceptually similar.}$$

The most cases, features are treated as points in high-dimensional space. Therefore it is naturally to define distance functions in terms of Euclidean norms. The $L_1$ norm and $L_2$ norm are commonly used when comparing two feature vectors. In practice, the $L_1$ distance measure performs better than the $L_2$ distance measure because the former is statistically more robust to outliers [RL87] when the outliers come from the measurement (i.e., $y$) axis. Hafner *et al.* [HSE95] suggest using a more sophisticated quadratic form of distance measure, which tries to capture the perceptual similarity between any two colors. To avoid intensive computation of quadratic functions, they propose to use low-dimensional color features as filters before using the quadratic form for the distance measure.

Sometimes when features describe distributions of some random variables in an image, such as color histograms, texture orientation histograms, we can borrow some statistical measures to define the distance function for features, such as the Mahalanobis distance [DH73], Chi-Square test, and K-S test [PTVF92]. Based on the minimal cross-entropy principle, the Kullback-Liebler divergence measure can be used to compare two distributions:

$$\mathcal{D}(f(I), f(J)) = \sum_{k=1}^{n} f(I)^k \log \frac{f(I)^k}{f(J)^k}.$$

Notice that these statistical measures are not metrics.

A recent metric proposed to measure the distance between two distributions is the Earth-Mover's Distance (EMD) [RTG98]. EMD reflects the minimal amount of work that must be performed to transform one distribution into the other by moving the "distribution mass" around. This measure comes from the transportation problem in combinatorial optimization. EMD can be computed by solving a linear programming problem. Therefore it is computationally expensive.

Though it seems that the distance functions should be a metric, recent research in computer vision and cognitive science, however, suggest that human perceptual judgment about visual similarity are inherently non-metric, i.e., they may not obey the triangle inequality or may not be symmetric. Some non-metric similarity measures are suggested for image classification [JWG98].

We will use the $L_1$ distance measure for comparing histograms and correlograms because it is simple and robust. The following formulae are used to compute the distance between images $\mathcal{I}$ and $\mathcal{I}'$:

$$|\mathcal{I}-\mathcal{I}'|_{h,L_1} \triangleq \sum_{i\in[m]} |h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')| \tag{3.3}$$

$$|\mathcal{I}-\mathcal{I}'|_{\gamma,L_1} \triangleq \sum_{i,j\in[m],k\in[d]} |\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) - \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')| \tag{3.4}$$

From these equations, it is clear that the contributions of different colors to the dissimilarity are equally weighted. Intuitively, however, this contribution should be weighted to take into account some additional factors.

**Example.** Consider two pairs of images $\langle\mathcal{I}_1,\mathcal{I}_2\rangle$ and $\langle\mathcal{I}'_1,\mathcal{I}'_2\rangle$. Let $h_{c_i}(\mathcal{I}_1) = 1000$, $h_{c_i}(\mathcal{I}_2) = 1050$, $h_{c_i}(\mathcal{I}'_1) = 100$, and $h_{c_i}(\mathcal{I}'_2) = 150$. Even though the absolute difference in the pixel count for color bucket $i$ is 50 in both cases, clearly the difference is more significant for the second pair of images. Thus, the difference $|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|$ in Equation (3.3) should be given more importance if $|h_{c_i}(\mathcal{I}) + h_{c_i}(\mathcal{I}')|$ is small and vice versa. We could therefore replace the expression $|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|$ in Equation 3.3 by

$$\frac{|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|}{1 + h_{c_i}(\mathcal{I}) + h_{c_i}(\mathcal{I}')} \tag{3.5}$$

(the 1 in the denominator prevents division by zero).

This intuition has theoretical justification in [Haus92] which suggests that sometimes, a "relative" measure of distance $d_\mu$ is better. For $\mu > 0, r, s \geq 0$, $d_\mu$ is defined by

$$d_\mu(r,s) = \frac{|r-s|}{\mu + r + s}. \tag{3.6}$$

It is straightforward to verify that (i) $d_\mu$ is a metric, (ii) for $r, s \geq 0$, $d_\mu(r,s) \in [0,1)$, and (iii) for $0 \leq r \leq s \leq t$, $d_\mu(r,s) \leq d_\mu(r,t)$, $d_\mu(s,t) \leq d_\mu(r,t)$.

$d_\mu$ can be applied to feature vectors also. We have set $\mu = 1$. So the $d_1$ distance measure for histograms and correlograms is:

$$|\mathcal{I}-\mathcal{I}'|_{h,d_1} \triangleq \sum_{i\in[m]} \frac{|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|}{1 + h_{c_i}(\mathcal{I}) + h_{c_i}(\mathcal{I}')} \tag{3.7}$$

$$|\mathcal{I}-\mathcal{I}'|_{\gamma,d_1} \triangleq \sum_{i,j\in[m],k\in[d]} \frac{|\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) - \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')|}{1 + \gamma^{(k)}_{c_i,c_j}(\mathcal{I}) + \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')} \tag{3.8}$$

We also study the effectiveness of non-metric measures such as the divergence measure and the measure suggested in [JWG98]. We use the empirical Jeffrey-divergence: (which is numerically stable, symmetric, robust to noise and the size of histogram bins)

$$\mathcal{D}(f(I), f(J)) = \sum_{k=1}^{n} (f(I)^k \log \frac{f(I)^k}{\hat{f}^k} + f(J)^k \log \frac{f(J)^k}{\hat{f}^k})$$

where

$$\hat{f} = \frac{f(I) + f(J)}{2}$$

and $L_{\frac{1}{2}}$ measure used in [JWG98]

$$\mathcal{D}(f(I), f(J)) = \sum_{k=1}^{n} \sqrt{(f(I)^k - f(J)^k}.$$

## 3.4 Related Work in CBIR

Recently several approaches have attempted to incorporate spatial information with color [HCP95, SD96, SC96a, RS96, PZM96, PZ96, PYL97, RGT97, GPF98]. *Region-based* approaches divide images into subregions (by some segmentation) and impose spatial constraints on the image comparison while *histogram-based* approaches augment histograms with spatial properties or constraints. In the following, we briefly review some of the previous work.

Hsu *et al.* [HCP95] select two representative colors signifying the “background” and the principal “object” in an image. The maximum entropy algorithm is then used to partition an image into rectangular regions. Only one selected color dominates a region. The similarity between two images is the degree of overlap between regions of the same color. Unfortunately, this method uses very coarse color segmentation and is susceptible to false positives.

Smith and Chang [SC96a] partition an image into binary color sets. They first select colors that are “sufficiently” present in a region. The colors for a region are represented by a binary color set that is computed using histogram back-projection [SB91]. The binary color sets and their location information constitute the feature.

Stricker and Dimai [SD96] divide an image into five fixed overlapping regions and extract the first three color moments of each region to form a feature vector for the image. The storage requirements for this method are low. The use of overlapping regions makes the feature vectors relatively insensitive to small rotations or translations.

To obtain better region representations, color clustering is used to segment an image into a set of contiguous regions [RGT97, GPF98]. In [RGT97] a *color signature* represents an image. The signature of an image contains points which represent clusters of similar colors in CIE-LAB space, and the weight of a point is the fraction of the image area with that color. The size of a signature is typically eight to 12. Signatures are compared using the EMD function, which involves solving a linear programming problem. Therefore processing a query is not fast.

In [GPF98], color clustering is exploited in the HVC color space. The maximum number of clusters is set to 20. Based on color clusters, contiguous regions are obtained by region labeling. A feature vector for a color cluster includes three H, V, C values, the normalized pixel count, and the cluster aggregation degree value. For each region, a 24-dimensional feature vector is created to include average H, V, C values and 21 shape profile values (the same as in [Gong98]). The feature vectors are compared by weighted $L_1$ distance normalized by the standard variances of each feature value over the entire database.

Pass and Zabih [PZ96] use another approach. They partition histogram bins by the spatial coherence of pixels. A pixel is coherent if it is a part of some "sizable" similar-colored region, and incoherent otherwise. A color coherence vector (CCV) [PZM96] represents this classification for each color in the image. CCV's are fast to compute and appear to perform better than histograms. The notion of CCV is also extended in [PZ96], by using additional feature(s) to further refine the CCV-refined histogram. One such extension uses the center of the image (the center-most 75% of the pixels are defined as the "center") as the additional feature. The enhanced CCV is called CCV with *successive refinement* (CCV/C) and performs better than CCV.

Park *et al.* [PYL97] use a graph representation Modified Color Adjacency Graph (MCAG) (inspired by CAG from [MMK95]) to combine the color histogram information and spatial adjacency information. Each node of MCAG is a color, and the node attribute encodes the pixel count of the RGB component. Each edge records the pixel count of the common boundary between two adjacent color regions. To make use of the geometric statistics of each color component, a Spatial Variance Graph (SVG) is constructed in addition to MCAG. The similarity of two graphs is measured by graph intersection generalized from histogram intersection [SB91]. The similarity of two images is measured by a weighted sum of similarities of corresponding MCAG's and SVG's.

The correlogram is neither a region-based nor a histogram-based method. Unlike purely local properties, such as pixel position, gradient direction, or purely global properties, such as color distribution, correlogram takes into account the local color spatial correlation as well as the global distribution of this spatial correlation. While any scheme that is based on purely local properties is likely to be sensitive to large appearance changes, correlogram is more stable to tolerate these changes (see Section 3.8.2); while any scheme that is based on purely global properties is susceptible to false positive matches, correlogram seems to be scalable for CBIR (see Section 3.8.7).

## 3.5 Performance Measures

The image retrieval problem is the following: let $\mathcal{D}$ be an image database and $\mathcal{Q}$ be the query image. Obtain a permutation of the images in $\mathcal{D}$ based on $\mathcal{Q}$, i.e., assign $\mathrm{rank}(\mathcal{I}) \in [|\mathcal{D}|]$ for each $\mathcal{I} \in \mathcal{D}$, using some notion of similarity to $\mathcal{Q}$. This problem is usually solved by sorting the images $\mathcal{I} \in \mathcal{D}$ according to $|f(\mathcal{I}) - f(\mathcal{Q})|$, where $f(\cdot)$ is a function computing feature vectors of images and $|\cdot|_f$ is some distance metric defined on feature vectors.

**Performance Measure.** Let $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_q\}$ be the set of query images. For a query $\mathcal{Q}_i$, let $\mathcal{I}_i$ be the unique correct answer. We use two performance measures:

1. $r$-*measure* of a method which sums up over all queries, the rank of the correct answer, i.e., $\sum_{i=1}^{q} \mathrm{rank}(\mathcal{I}_i)$. We also use the average $r$-measure which is the $r$-measure divided by the number of queries $q$.

2. $p_1$-*measure* of a method which is $\sum_{i=1}^{q} 1/\mathrm{rank}(\mathcal{I}_i)$, i.e., the sum (over all queries) of the precision at recall equal to 1. The average $p_1$-measure is the $p_1$-measure divided by $q$.

Images ranked at the top contribute more to the $p_1$-measure. Note that a method is good if it has a low $r$-measure and a high $p_1$-measure.

3. *Recall* vs. *Scope:* Let $\mathcal{Q}$ be a query and let $\mathcal{Q}'_1, \ldots, \mathcal{Q}'_a$ be multiple "answers" to the query ($\mathcal{Q}$ is called a *category query*). Now, the *recall* $r$ is defined for a *scope* $s > 0$ as $|\{\mathcal{Q}'_i \mid$

$\mathrm{rank}(\mathcal{Q}'_i) \leq s\}|/a$. This measure is simpler than the traditional *recall vs. precision* but still evaluates the effectiveness of the retrieval [HCP95, SC96a].

## 3.6 Efficiency Considerations

As image databases grow larger, CBIR systems need to address efficiency issues in addition to the issue of retrieval effectiveness. We investigate several general methods to improve the efficiency of indexing and searching, without compromising effectiveness.

**Parallelization.** The construction of a featurebase for an image database is readily parallelizable. We can divide the database into several parts, construct featurebases for these parts simultaneously on different processors, and finally combine them into a single featurebase for the entire database.

**Partial Correlograms.** In order to reduce space and time requirements, we choose a small value of $d$. This does not impair the quality of correlograms or autocorrelograms very much because in an image, local correlations between colors are more significant than global correlations. Sometimes, it is also preferable to work with *distance sets,* where a distance set $D$ is a subset of $[d]$. We can thus cut down storage requirements, while still using a large $d$. Note that our algorithm can be modified to handle the case when $D \subset [d]$.

Though in theory the size of a correlogram is $O(m^2 d)$ (and the size of an autocorrelogram is $O(md)$), we observe that the feature vector is not always dense. This sparsity could be exploited to cut down storage and speed up computations. Replacing all sufficiently small entries by zero and using a sparse representation for correlograms allows us cut down storage and speed up computations.

**Filtering.** There is typically a tradeoff between the efficiency and effectiveness of search algorithms: more sophisticated methods which are computationally more expensive tend to yield better retrieval results. Good results can be obtained without sacrificing too much in terms of efficiency by adopting a two-pass approach [HSE95]. In the first pass, we retrieve a set of $N$ images in response to a query image by using an inexpensive (and possibly crude) search algorithm. Even though the ranking of these images could be unsatisfactory, we just need to guarantee that useful images are contained in this set. We can then use a more sophisticated matching technique to compare the query image to these $N$ images only (instead of the entire database), and the best images are likely to be highly ranked in the resulting ranked list. It is important to choose an appropriate $N$ in this approach[1] — the initially retrieved set should be good enough to contain the useful images and should be small enough so that the total retrieval time is reduced.

## 3.7 Experimental Setup

It is important to evaluate performance scientifically in order to ensure the validity of the results. We develop several different benchmarks to evaluate performance. Our initial results are using a database *IC14* with 14,554 color images of size $232 \times 168$. This includes 11,667 images used in Chabot [CS95], 1,440 images used in QBIC [FSN$^+$95], and 1,005 images from Corel. It also includes a few groups of images in PhotoCD format and a number of MPEG video frames from the web [PZ96]. Our heterogeneous image database is thus very realistic and helps us in a fair

[1]Equivalently, we could select some threshold image score.

evaluation of various methods. It consists of images of animals, humans, landscapes, and various objects like tanks, flags, etc.

In order to test scalability, we use two other larger image databases with $18,140$ images *IC18* and $210,911$ images *IC200* by adding CNN video frames to the *IC14*. Standard statistical tests will be used to determine if one feature (or distance measure) is significantly better than another.

We consider the RGB colorspace with quantization into 64 colors (this number seems to work the best with *IC14* [PZM96]. One can also examine the whole collection and quantize the color space using the "optimal number" [WK96]. In our case, the performance does not vary much with HSV instead of RGB.) To improve performance, we first smooth the images by a small amount. We use the distance set $D = \{1, 3, 5, 7\}$ (so, $d = 4$) for computing the autocorrelograms. We use $b = 4$ for the banded autocorrelogram. This results in a feature vector that is as small as the histogram. The query response time from *IC18* (including I/O time) is under 14 seconds for autocorrelograms, 5 seconds for banded autocorrelograms and 4 seconds for histograms.

Our benchmark consists of 77 queries, each with a *unique correct* answer identified by hand from *IC14* before running any experiments (examples of some queries and answers are shown in Figure 3.2 and Figure 3.3). The queries are chosen to represent various situations like different views of the *same scene*, large changes in appearance, small lighting changes, spatial translations, shape changed from camera zooms, etc. These data serve as the ground truth to test different image features. We also run 4 category queries, each with $a > 1$ answers (see Figure 3.4)– Query 1 ($a = 22$ owl images), Query 2 ($a = 17$ fox images), Query 3 ($a = 6$ movie scenes), and Query 4 ($a = 6$ moving car images).

We compare six image features: color histogram (hist), color coherent vectors with successive refinement (CCV/C), autocorrelograms (auto), banded autocorrelograms (b-auto), edge autocorrelograms (e-auto), and banded edge autocorrelograms (be-auto). These results are tested on *IC14* using the $L_1$ norm. We then test the scalability of three features: color histogram, banded autocorrelograms, and histogram refinement with $4$ level of edge density and $4$ level of textureness hr(16) [PZ98] (therefore the size of hr(16) is 16 times of that of the histogram) on image databases *IC18* and *IC200*. We also compare the effectiveness of four different distance measures: $L_1$, $d_1$, Jeffrey-divergence, and $L_{\frac{1}{2}}$.

## 3.8 Results

### 3.8.1 Unique Answer Queries

Table 3.1 compares the overall performance on *IC14* obtained using the histogram, CCV, and correlogram methods when the color space is divided into 8 bins. $L_1$ is the distance measure used.

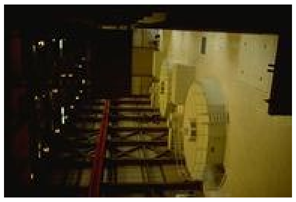
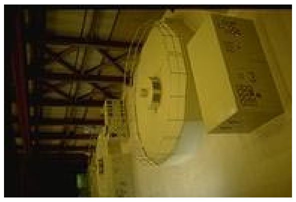
Table 3.1: Performance of various methods (8 colors).

| Method | hist | ccv | corr | auto |
|---|---|---|---|---|
| avg. $r$-measure | 664 | 626 | **68** | 87 |
| avg. $p_1$-measure | 0.06 | 0.08 | **0.42** | 0.27 |

Both the $r$-measure and the $p_1$-measure show that correlograms produce significantly better

| | | |
|---|---|---|
| hist: 496. | ccv(s): 245. | auto: 2. |
| b-auto: 2. | e-auto: 2. | be-auto: 2. |

| | | |
|---|---|---|
| hist: 411. | ccv(s): 56. | auto: 1. |
| b-auto: 1. | e-auto: 1. | be-auto: 1. |

| | | |
|---|---|---|
| hist: 367. | ccv(s): 245. | auto: 1. |
| b-auto: 1. | e-auto: 8. | be-auto: 9. |

Figure 3.2: Sample queries and answers with ranks for various methods retrieved from *IC14*. (Lower ranks are better.)

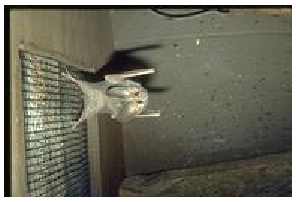
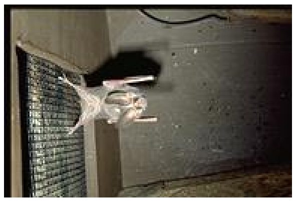
results than the other histogram-based features. In fact, 8-color correlograms perform better than 64-color histograms (see Table 3.2). The change in the average $r$-measure for the correlogram method shows that this method ranks the correct answers 596 positions higher on average compared to the histogram method.

The overall performance of the autocorrelogram, histogram, CCV, and CCV/C using 64 color buckets is compared in Table 3.2. Observe that all the correlogram-related methods are on the same par in terms of performance and are significantly better than histogram and CCV/C. On average, in the autocorrelogram-based method, the correct answer shows up second while for histograms and CCV-based methods, the correct answer shows up at about $80$-th and $40$-th places. The banded autocorrelograms perform only slightly worse than the autocorrelogram. With the same feature size as the histograms, the banded autocorrelograms retrieve the correct answers at more than $79$ positions higher than histograms. Thus the banded autocorrelogram serves much better than the color histogram for image retrieval. Since the autocorrelograms already achieve good retrieval results, the edge correlograms do not generate too much improvement.

Also note that the banded edge autocorrelograms have higher $p_1$-measure than the edge auto-

| | | |
|---|---|---|
| hist: 310. | ccv(s): 160. | auto: 5. |
| b-auto: 5. | e-auto: 1. | be-auto: 1. |

| | | |
|---|---|---|
| hist: 198. | ccv(s): 6. | auto: 12. |
| b-auto: 13. | e-auto: 5. | be-auto: 4. |

| | | |
|---|---|---|
| hist: 119. | ccv(s): 25. | auto: 2. |
| b-auto: 3. | e-auto: 1. | be-auto: 1. |

Figure 3.3: Sample queries and answers with ranks for various methods retrieved from *IC14*. (Lower ranks are better.)

correlograms. This is because most of the ranks go higher while only a few go lower. Though the $r$-measure becomes worse, the $p_1$-measure becomes better. It is remarkable that banded autocorrelogram has the same amount of information as the histogram, but seems lot more effective than the latter.

For 73 out of 77 queries, autocorrelograms perform as well as or better than histograms. In the cases where autocorrelograms perform better than color histograms, the average improvement in rank is 104 positions. In the four cases where color histograms perform better, the average improvement is just two positions. Autocorrelograms, however, still rank the correct answers within top six in these cases.

**Statistical Significance Analysis.** We adopt the approach used in [PZ96] to analyze the statistical significance of the improvements. We formulate the null hypothesis $H_0$ which states that the autocorrelogram method is as likely to cause a negative change in rank as a non-negative one. Under $H_0$, the expected number of negative changes is $M = 38.5$, with a standard deviation $\sigma = \sqrt{77}/2 \approx 4.39$. The actual number of negative changes is $4$, which is less than $M - 7\sigma$. We can reject $H_0$ at more than $99.9\%$ standard significance level.

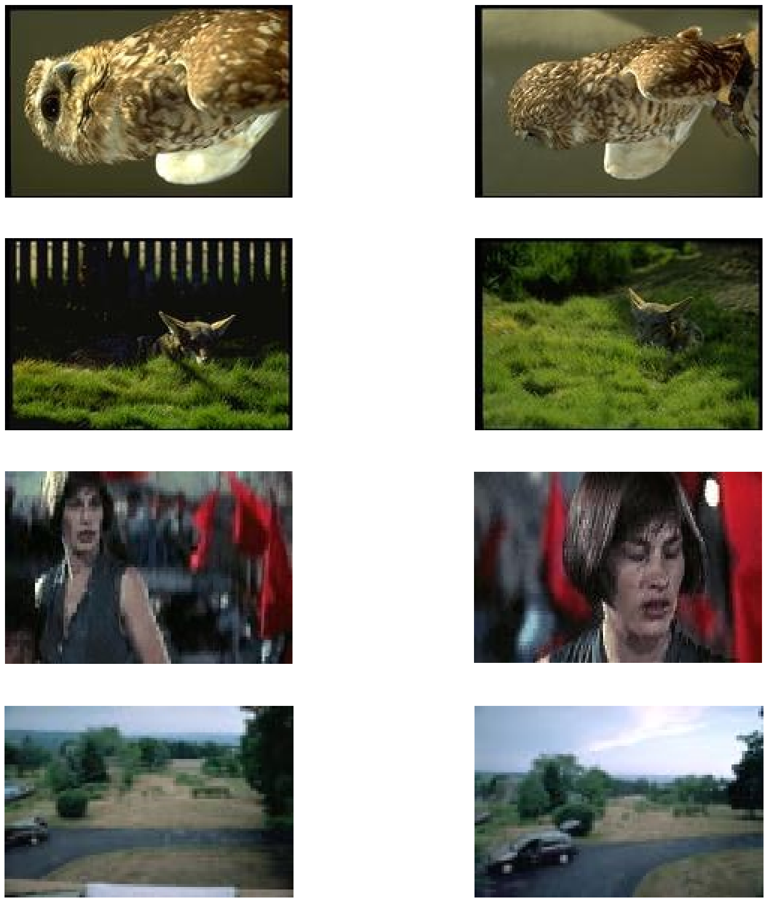


Figure 3.4: Sample category queries and answers.

For 67 out of 77 queries, autocorrelograms perform as well as or better than CCV/C. In the cases where autocorrelograms perform better than CCV/C, the average improvement in rank is 66 positions. In the ten cases where CCV/C perform better, the average improvement is two positions. Autocorrelograms, however, still rank the correct answers within top 12 in these cases. Again, statistical analysis shows that autocorrelograms are better than CCV/C.

From a usability point of view, we make the following observation. Given a query, the user is guaranteed to locate the correct answer by just checking the top two search results (on average) in the case of autocorrelogram. On the other hand, the user needs to check at least the top 80 search results (on average) to locate the correct answer in the case of histogram (or top 40 search results for the CCV/C). In practice, this implies that the former is a more “usable” image retrieval scheme than the latter two.

### 3.8.2 Stability

The results show that the autocorrelogram tolerates large changes in appearance of the same scene caused by changes in viewing positions, changes in the background scene, partial occlusions, camera zoom that causes radical changes in shape, etc. Since we choose small values $\{1, 3, 5, 7\}$

Table 3.2: Comparison of various image retrieval methods.

| Method | hist | ccv/c | auto | b-auto | e-auto | be-auto |
|---|---|---|---|---|---|---|
| avg. $r$-measure | 81.8 | 42.5 | 2.2 | 2.5 | **1.9** | 2.0 |
| avg. $p_1$-measure | 0.28 | 0.41 | 0.75 | 0.72 | 0.78 | **0.79** |

for the distance set $D$, the autocorrelogram distills the global distribution of local color spatial correlations. In the case of camera zoom (for example, the third pair of images on the left column of Figure 3.2), though there are big changes in object shapes, the local color spatial correlations as well as the global distribution of these correlation do not change that much. We illustrate this by looking at how the autocorrelation of yellow color changes with distance in the following three images (Figure 3.8.2). Notice that the size of yellow circular and rectangular objects in the query image and the image ranked one are different. Despite this, the correlation of yellow with yellow for the local distance of the image ranked one is closer to that of the query image than the image ranked, say, two (Figure 3.8.2).

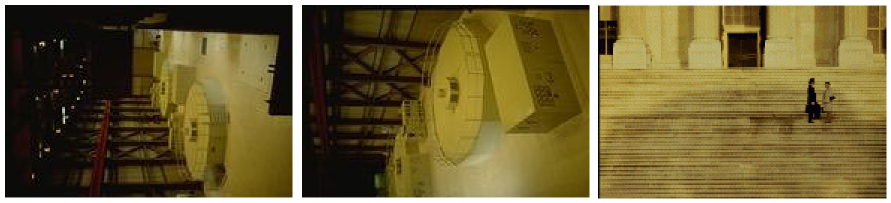


Figure 3.5: The query image, the image ranked one, and the image ranked two.

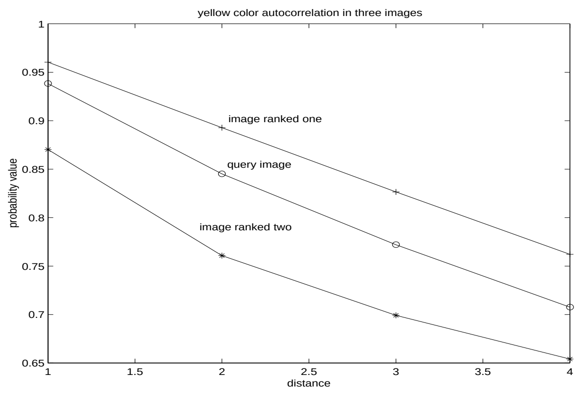


Figure 3.6: The change of autocorrelation of yellow color with distance.

### 3.8.3 Recall Comparison

Table 3.3 shows the performance of three features on our four category queries. The $L_1$ distance metric is used. Once again, autocorrelograms perform the best.

Table 3.3: Scope *vs.* recall results for category queries. (Larger numbers indicate better performance.)

| | Recall | | | | | |
|---|---|---|---|---|---|---|
| | Query 1 | | | Query 2 | | |
| Scope | hist | ccv/c | auto | hist | ccv/c | auto |
| 10 | .14 | .19 | **.24** | .13 | .19 | **.38** |
| 30 | .19 | .19 | **.38** | .31 | .38 | **.63** |
| 50 | .19 | .24 | **.57** | .31 | .38 | **.75** |

| | Recall | | | | | |
|---|---|---|---|---|---|---|
| | Query 3 | | | Query 4 | | |
| Scope | hist | ccv/c | auto | hist | ccv/c | auto |
| 10 | .20 | .20 | **1.0** | .20 | .20 | **.60** |
| 30 | .40 | .20 | **1.0** | .20 | .20 | **.80** |
| 50 | .40 | .60 | **1.0** | .20 | .20 | **.80** |

### 3.8.4 Filtering

Table 3.4 shows the results of applying a histogram filter before using the autocorrelogram (we use 64-color histograms and autocorrelograms). As we see, the quality of retrieval even improves

Table 3.4: Performance of auto($L_1$) with hist($d_1$) filter.

| Method | unfiltered | filtered |
|---|---|---|
| $r$-measure | 172 | 166 |
| $p_1$-measure | 58.02 | 58.60 |

somewhat (because false positives are eliminated). As anticipated, the query response time is less since we consider the correlograms of only a small filtered subset of the featurebase.

### 3.8.5 Relative Distance Metric

Table 3.5 compares the results obtained using $d_1$ and $L_1$ distance measures on histograms, one of histogram refinement variants hr(16), and banded-autocorrelogram. The test results are 77 queries[2] from *IC18*. Using $d_1$ distance measure is clearly superior. The improvement is specially noticeable for histograms and variants of histogram refinement methods (see Figure 3.7).

Among the three features, banded autocorrelograms perform the best in terms of both $r$-measure and $p_1$-measure with the same size as histograms. The histogram refinement (hr(16)) certainly improves upon the histogram with additional local spatial information.

A closer examination of the results shows, however, that there are instances where the $d_1$ distance measure performs poorly compared to the $L_1$ distance measure on histograms and CCV/C.

[2]not exactly the same 77 pairs of images as before because some images are missing in *IC18* and *IC200* on NT machines.

Table 3.5: Comparison of $L_1$ and $d_1$ on database *IC18*

| Method | hist | hr(16) | b-auto | hist | hr(16) | auto |
|---|---|---|---|---|---|---|
| | $L_1$ distance measure | | | $d_1$ distance measure | | |
| avg. $r$-measure | 94.4 | 12.5 | **2.3** | 6.9 | 6.6 | **2.8** |
| avg. $p_1$-measure | 0.30 | 0.59 | **0.76** | 0.62 | 0.75 | **0.77** |

An example is shown in Figure 3.8. It seems that the failure of the $d_1$ measure is related to the large change of overall image brightness (otherwise, the two images are almost identical). We need to examine such scenarios in greater detail. Autocorrelograms, however, are not affected by $d_1$ in this case. Nor does $d_1$ improve the performance of autocorrelogram much. In other words, autocorrelograms seem indifferent to the $d_1$ distance measure. This needs to be formally investigated.

### 3.8.6 Other Distance Measures

Tables 3.6, 3.7, and 3.8 compare the results obtained from image database *IC18* using four distance functions — $L_1$, $d_1$, Jeffrey-divergence measure, and $L_{\frac{1}{2}}$ on three features — histograms, histogram refinement hr(16), and banded autocorrelogram.

For histograms, $d_1$ gives the best results. When compared to $L_1$, it reduces the average rank from $94$ to 7, and doubles the precision value at recall. The divergence measure and $L_{\frac{1}{2}}$ measure also perform better than $L_1$ with significant computation overhead.

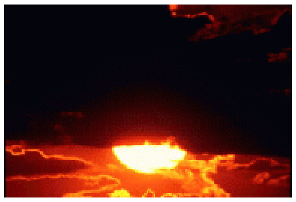
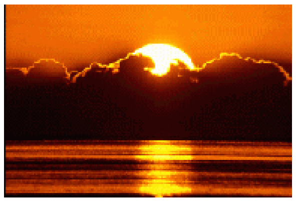
For histogram refinement feature hr(16), the divergence measure performs the best in terms of $r$-measure, while $d_1$ performs the best in terms of $p_1$-measure. $L_1$ gives the worst performance. From the computation and performance points of view, $d_1$ is the best choice among the four distance measures for hr(16).

For banded autocorrelograms, $L_1$ gives the lowest rank while $d_1$ gives the highest precision value at recall one. The divergence measure performs the worst. Since there is not much difference in terms of performance between $L_1$ and $d_1$, $L_1$ seems to be the best distance function for banded-autocorrelograms.

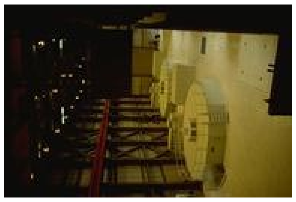
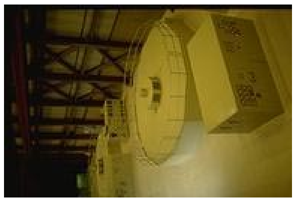
In summary, we observe that different distance functions perform differently on the same feature, and different features benefit differently from the same distance function. Our results indicate that $d_1$ is an economical and effective distance function for content-based image retrieval.

Table 3.6: Comparison of various distance functions on histograms from *IC18*.

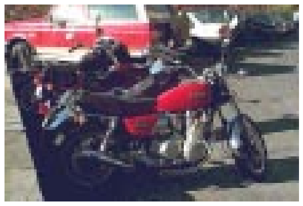
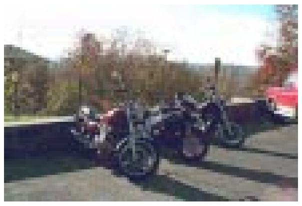
| Method | $L_1$ | $d_1$ | divergence | $L_{\frac{1}{2}}$ |
|---|---|---|---|---|
| avg. $r$-measure | 94.4 | **6.9** | 17.5 | 25.9 |
| avg. $p_1$-measure | 0.30 | **0.62** | 0.50 | 0.42 |

hist: 1035. hr(16): 404. b-auto:2. ($L_1$)
hist: 1. hr(16): 3. b-auto:2. ($d_1$)

hist: 509. hr(16): 92. b-auto:1. ($L_1$)
hist: 1. hr(16): 2. b-auto:1. ($d_1$)

hist: 375. hr(16): 54. b-auto:3. ($L_1$)
hist: 11. hr(16): 7. b-auto:5. ($d_1$)

Figure 3.7: Cases where $d_1$ is much better than $L_1$ on *IC18*.

### 3.8.7 Scalability

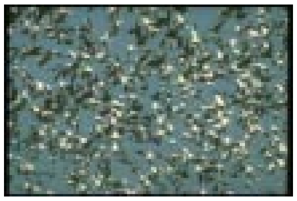
We compare the scalability of different features in Table 3.9, Table 3.10, Table 3.11, and Table 3.12. The distance functions also affect the scalability of image features.

If the $L_1$ metric is used, both histograms and histogram refinement hr(16) are not scalable, but banded autocorrelograms retain the performance very well when the size of the image database increases from $18,000$ to $200,000$. We also plot the bar chart of rank changes of hist, hr(16), and b-auto in Figure 3.9, Figure 3.10, Figure 3.11. For histograms, there are about $20$ queries whose rank changes are greater than $1000$. The relative change of $p_1$-measure for hr(16) is small, however. This is because most of the ranks do not change much (centered around $50$, see Figure 3.10), while the big changes occur to those with already high ranks.

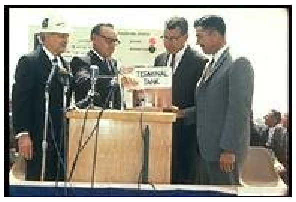
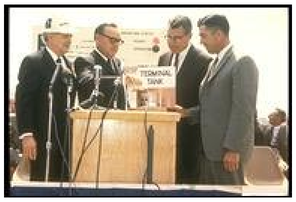
If the $d_1$ metric is used, all three features are scalable, and hr(16) changes the least. If the divergence measure is used, all three features are not scalable. While the relative rank changes are big, the relative $p_1$ changes are small. This is again because some queries with high rank are susceptible to false matches when more images are added in. As a result, their rank changes are too big to boost up the relative rank changes. If $L_{\frac{1}{2}}$ is used, only b-auto is scalable, and its changes are insignificant.

Combining the results with those from Section 3.8.5 and Section 3.8.6, we show that $d_1$ is

hist: 30. hr(16): 3. b-auto: 5. ($L_1$)
hist: 7. hr(16): 294. b-auto: 7. ($d_1$)



hist: 64. hr(16): 2. b-auto: 1. ($L_1$)
hist: 188. hr(16): 7. b-auto: 1. ($d_1$)

hist: 1. hr(16): 1. b-auto: 1. ($L_1$)
hist: 65. hr(16): 1. b-auto: 1. ($d_1$)

Figure 3.8: Cases where $d_1$ is worse than $L_1$ on *IC18*.

not only effective and economical, but also scalable for all three image features. The divergence metric is not recommended because it is not scalable; $L_{\frac{1}{2}}$ is not recommended for histogram-based features. These two distance measures are anyway expensive to compute.

We also conclude that histograms are not scalable unless $d_1$ metric is used; banded autocorrelograms are scalable unless the divergence metric is used; and histogram refinement methods hr(16) are not scalable unless $d_1$ metric is used.

## 3.9 Summary

In Summary, we applied the color correlogram to the problem of content-based image indexing/retrieval. The highlights of the correlogram are:

- it describes the global distribution of local spatial correlations of colors;
- it is easy to compute;
- the size of the feature is fairly small; the size of banded autocorrelogram can be the same as the histogram;

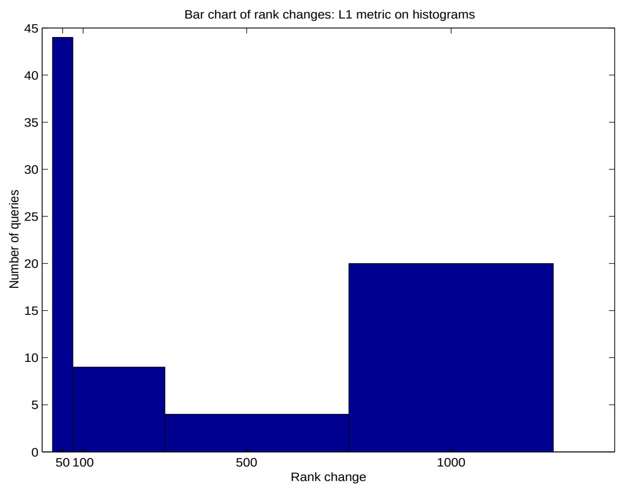


Figure 3.9: Bar chart of rank changes of histograms.

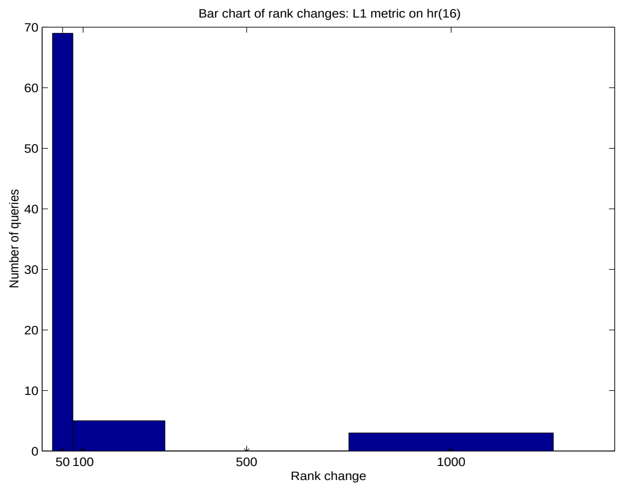


Figure 3.10: Bar chart of rank changes of histogram refinement hr(16).

Table 3.7: Comparison of various distance functions on histogram refinement hr(16) from *IC18*.

| Method | $L_1$ | $d_1$ | divergence | $L_{\frac{1}{2}}$ |
|---|---|---|---|---|
| avg. $r$-measure | 12.5 | 6.6 | **4.0** | 4.7 |
| avg. $p_1$-measure | 0.59 | **0.75** | 0.70 | 0.67 |

Table 3.8: Comparison of various distance functions on banded autocorrelograms from *IC18*.

| Method | $L_1$ | $d_1$ | divergence | $L_{\frac{1}{2}}$ |
|---|---|---|---|---|
| avg. $r$-measure | **2.3** | 2.8 | 12 | 3.3 |
| avg. $p_1$-measure | 0.76 | **0.77** | 0.69 | 0.70 |

- it is stable to tolerate large image appearance change caused by changes in camera viewing positions, changes in the background scene, partial occlusions, camera zoom that causes radical changes in shape, etc.

- its performance is very well scalable for very large image databases.

Our experimental evidence shows that this new feature outperforms both the traditional histogram method and the very recently proposed histogram refinement method for content-based image indexing/retrieval. With the same feature size, banded autocorrelograms achieve much better retrieval performance than histograms.

We discussed the scalability issues for histograms, one variant of histogram refinement and correlograms. We also studied different distance measures for comparing image features, such as $L_1$, Jeffrey-divergence measure and $L_{\frac{1}{2}}$ measure. While the newly proposed $d_1$ metric has negligible impact on correlograms, it seems to improve the quality and scalability for histogram-based features. It is also simpler and more economical to compute than Jeffrey's divergence measure or the $L_{\frac{1}{2}}$ measure. We recommend using $d_1$ metric for histogram-based features, and the traditional $L_1$ metric for correlograms.

Table 3.9: Results from *IC18* and *IC200* compared by $L_1$. Lower rank $\Longrightarrow$ better; higher $p_1$ $\Longrightarrow$ better; smaller change $\Longrightarrow$ more scalable.

| Database | Method | hist | hr(16) | b-auto |
|---|---|---|---|---|
| *IC18* | avg. $r$-measure | 94.4 | 12.5 | **2.3** |
| *IC200* | avg. $r$-measure | 2042.5 | 464.2 | **2.9** |
| | relative change(%) | 2064 | 3614 | **26** |
| *IC18* | avg. $p_1$-measure | 0.30 | 0.59 | **0.76** |
| *IC200* | avg. $p_1$-measure | 0.23 | 0.56 | **0.71** |
| | relative change(%) | 23 | **5** | 6 |

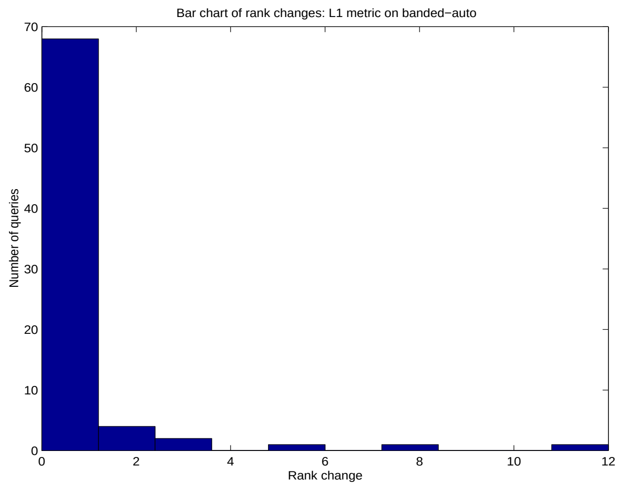


Figure 3.11: Bar chart of rank changes of banded autocorrelograms.

Table 3.10: Results from *IC18* and *IC200* compared by $d_1$. Lower rank $\Longrightarrow$ better; higher $p_1$ $\Longrightarrow$ better; smaller change $\Longrightarrow$ more scalable.

| Database | Method | hist | hr(16) | b-auto |
|---|---|---|---|---|
| *IC18* | avg. $r$-measure | 6.9 | 6.6 | **2.8** |
| *IC200* | avg. $r$-measure | 10.7 | 7.1 | **3.2** |
| | relative change(%) | 55 | **8** | 14 |
| *IC18* | avg. $p_1$-measure | 0.62 | 0.75 | **0.77** |
| *IC200* | avg. $p_1$-measure | 0.59 | 0.75 | 0.75 |
| | relative change(%) | 5 | **0** | 3 |

Table 3.11: Results from *IC18* and *IC200* compared by divergence measure. Lower rank $\Longrightarrow$ better; higher $p_1$ $\Longrightarrow$ better; smaller change $\Longrightarrow$ more scalable.

| Database | Method | hist | hr(16) | b-auto |
|---|---|---|---|---|
| *IC18* | avg. $r$-measure | 17.5 | **4.0** | 11.7 |
| *IC200* | avg. $r$-measure | 149.7 | **25.1** | 46.1 |
| | relative change(%) | 755 | 528 | **294** |
| *IC18* | avg. $p_1$-measure | 0.50 | 0.70 | 0.70 |
| *IC200* | avg. $p_1$-measure | 0.41 | **0.67** | 0.65 |
| | relative change(%) | 18 | **4** | 7 |

Table 3.12: Results from *IC18* and *IC200* compared by $L_{\frac{1}{2}}$. Lower rank $\Longrightarrow$ better; higher $p_1$ $\Longrightarrow$ better; smaller change $\Longrightarrow$ more scalable.

| Database | Method | hist | hr(16) | b-auto |
|---|---|---|---|---|
| *IC18* | avg. $r$-measure | 25.9 | 4.7 | **3.3** |
| *IC200* | avg. $r$-measure | 234.8 | 44.2 | **4.2** |
| | relative change(%) | 807 | 840 | **3** |
| *IC18* | avg. $p_1$-measure | 0.42 | 0.67 | **0.70** |
| *IC200* | avg. $p_1$-measure | 0.34 | 0.65 | **0.69** |
| | relative change(%) | 19 | 3 | **1** |

# Chapter 4

# Image Subregion Querying and Localization

The notion of image content arises not only from the overall image appearance, but also from the objects present inside the image. This is an important reason why people prefer to query for images using abstractly defined objects/categories (e.g., “retrieve all images with a red rose”). Conventional object recognition techniques are not advanced enough to recognize general objects in arbitrary images.

A simpler version of the object recognition problem is the *image subregion querying* problem: given as input a subregion $\mathcal{Q}$ of an image $\mathcal{I}$ and an image set $\mathcal{S}$, retrieve from $\mathcal{S}$ those images $\mathcal{I}'$ in which the query $\mathcal{Q}$ appears according to human perception (denoted $\mathcal{I}' \subseteq \mathcal{Q}$). The set of images might consist of a database of still images, or videos, or some combination of both. The problem is made even more difficult than image retrieval by a wide variety of effects that cause the same object to appear different (such as changing viewpoint, different pose and shape, camera noise and occlusion).

The image subregion querying problem arises in image retrieval and in video browsing when a user is more interested in objects themselves rather than the background. For example, a user might wish to find pictures in which a given object appears, or scenes in a video that contain a specified person.

Once it is determined that an image contains an object, it becomes necessary in many cases (like tracking of objects in a video sequence) to locate the object inside the image. This is called the *localization* problem.

In this chapter, we study the use of color correlograms for solving the image subregion querying and localization problems. We show that using correlograms can substantially improve upon the results obtained using the histogram intersection and backprojection methods.

**Organization.** : Section 4.1 briefly reviews related work in image subregion querying and localization problems. Section 4.2 and Section 4.3 present our correlogram-based methods for the image subregion querying and localization problems. Experimental results are presented within each section.

## 4.1 Related Work

The image subregion querying problem is closely related to object recognition, which has been studied for a long time by the computer vision community [Robe65]. Some of the early work in object recognition and detection [MN78], suggests that geometric cues such as edge, surface, and depth information be identified before object recognition is attempted. Most of such object recognition systems compare the geometric features of the model with those of an image using various forms of search (some of which are computationally intensive [HU86, GL 1987]). Huttenlocher *et al.* [HKR93] propose using the Hausdorff distance to compare images. This method operates on the edge maps of images and is fairly fast. It has been successfully tested on cases where the model is allowed only rigid motions with respect to the image.

Such geometric information is hard to extract from an image, since geometric and photometric properties are relatively uncorrelated [RB95]. Moreover, the central tasks involved in this approach — edge detection and region segmentation — are difficult for unconstrained data that is encountered in the context of image retrieval and video browsing.

An alternative approach to model-based recognition is *appearance matching*. First, a database of object images under different view positions and lighting conditions is constructed. Then, *principal component analysis* is used to analyze only the photometric properties and ignore geometric properties [MN95, HLO96, RB95]. This model-based method is effective only when the principal components capture the characteristics of the whole database. For instance, it yields good results on the Columbia object database in which all images have a uniform known background. If there is a large variation in the images in a database, however, a small set of principal components is unlikely to do well on the image subregion querying task. In addition, the learning process requires homogeneous data and deals poorly with outliers. Therefore, this approach seems suitable only for domain-specific applications, but not for image subregion querying from a large heterogeneous image database such as the one used in our experiment.

Since the color information (e.g., histogram) is very easy to extract from an image, it has been successfully used for object indexing, detection, and localization [SB91, FSN$^+$95, CS95, MMK95, SH95, BE92, Syed97, FFB96, FMM$^+$96, DRD97]. We briefly review some of these approaches below.

Swain and Ballard [SB91] propose *histogram intersection* for object identification and *histogram backprojection* for object localization. The technique is computationally easy, does not require image segmentation or even foreground/background separation, and is insensitive to small changes in viewing positions, partial occlusion, and object deformation. Histogram backprojection is a very efficient process for locating an object in an image. It has been shown that this algorithm is not only able to locate an object but also to track a moving object. The advantages and disadvantages inherent to histograms in general have been discussed in detail in Section 4.2 and Section 4.3.

One disadvantage of color histograms is that they are sensitive to illumination changes. Slater and Healey [SH95] propose an algorithm that computes invariants of local color distribution and uses these invariants for 3-D object recognition. Illumination correction and spatial structure comparison are then used to verify the potential matches.

Matas *et al.* [MMK95] propose the color adjacency graph (CAG) as a representation for multiple-colored objects. Each node of a CAG represents a single color component of the image. Edges of the CAG label reflectance ratios [NB93] of adjacent color components. CAG's improve over histograms by incorporating coarse color segmentation into histograms. The set of visible colors and their adjacency relationship remain stable under changes of viewpoint and non-rigid transformations. The recognition and localization problems are solved by subgraph matching. Their approach yields excellent results, but the computational cost of subgraph matching is fairly high.

Das *et al.* modify the CAG to approximate the spatial relationship of color regions. They first obtain prominent colors by extracting peaks in a histogram. A Spatial Proximity Graph (SPG) is constructed with nodes representing each color peak, and edge representing possible adjacency between color regions. The adjacency information is obtained by examining color peaks in each cells (here an image is divided into some number of cells). A SPG may have fake edges indicating wrong adjacency information, but it does not matter. A query is first matched against the database using peak-matching to get a list of candidate images. The second phase of matching is done on the SPGs of those candidate images. Although the sub-graph isomorphism is **NP**-complete, some heuristics are employed to speed up the matching process.

Forsyth *et al.* [FMM$^+$96] offer different object models in order to achieve object recognition under general contexts. Their focus is on classification rather than identification. The central process is based on grouping (i.e., segmentation) and learning. They fuse different visual cues such as color and texture for segmentation; texture and geometric properties for trees; and color, texture and specialized geometric properties for human bodies.

## 4.2 Correlogram Intersection

The image subregion querying problem is a harder problem than image retrieval based on whole image matching. To avoid exhaustive searching subregions in an image, one scheme is to define *intersection* of color histograms [SB91]. The scheme can be interpreted in the following manner. (This interpretation helps us to generalize the method to correlograms easily.)

Given the histograms for a query $\mathcal{Q}$ and an image $\mathcal{I}$, the intersection of these two histograms can be considered as the histogram of an abstract entity notated as the intersection $\mathcal{Q} \cap \mathcal{I}$, (which will not be defined but serves as a conceptual and notational convenience only.) With the color count of the intersection defined as

$$H_{c_i}(\mathcal{Q} \cap \mathcal{I}) \triangleq \min\{H_{c_i}(\mathcal{Q}), H_{c_i}(\mathcal{I})\}, \tag{4.1}$$

we can define the intersection of the histograms of $\mathcal{Q}$ and $\mathcal{I}$ as

$$h_{c_i}(\mathcal{Q} \cap \mathcal{I}) \triangleq \frac{H_{c_i}(\mathcal{Q} \cap \mathcal{I})}{|\mathcal{Q}|}. \tag{4.2}$$

Note that this definition is *not* symmetric in $\mathcal{Q}$ and $\mathcal{I}$. The distance $|\mathcal{Q} - \mathcal{Q} \cap \mathcal{I}|_{h,L_1}$ is a measure of the presence of $\mathcal{Q}$ in $\mathcal{I}$. When $\mathcal{Q}$ is a subset of $\mathcal{I}$, $|\mathcal{Q} - \mathcal{Q} \cap \mathcal{I}|_{h,L_1} = 0$ because all the color counts in $\mathcal{Q}$ are less than those in $\mathcal{I}$, and the histogram intersection simply gives back the histogram for $\mathcal{Q}$.

In an analogous manner, we define the intersection correlogram as the correlogram of the intersection $\mathcal{Q} \cap \mathcal{I}$ (again, merely an abstract entity.) With the count

$$\Gamma^{(k)}_{c_i,c_j}(\mathcal{Q} \cap \mathcal{I}) \triangleq \min\{\Gamma^{(k)}_{c_i,c_j}(\mathcal{Q}), \Gamma^{(k)}_{c_i,c_j}(\mathcal{I})\}, \tag{4.3}$$

we can define the intersection correlogram as follows:

$$\gamma^{(k)}_{c_i,c_j}(\mathcal{Q} \cap \mathcal{I}) \triangleq \frac{\Gamma^{(k)}_{c_i,c_j}(\mathcal{Q} \cap \mathcal{I})}{H_{c_i}(\mathcal{Q} \cap \mathcal{I}) \cdot 8k}. \tag{4.4}$$

Again we measure the presence of $\mathcal{Q}$ in $\mathcal{I}$ by the distance $|\mathcal{Q} - \mathcal{Q} \cap \mathcal{I}|_{\gamma,L_1}$, say if $L_1$ distance measure were chosen. If $\mathcal{Q} \subseteq \mathcal{I}$, then the latter "should have at least as many counts of correlating color pairs" as the former. Thus the counts $\Gamma$ and $H$ for $\mathcal{Q} \cap \mathcal{I}$ are again those of $\mathcal{Q}$, and the correlogram of $\mathcal{Q} \cap \mathcal{I}$ becomes exactly the correlogram of $\mathcal{Q}$, giving $|\mathcal{Q} - \mathcal{Q} \cap \mathcal{I}|_{\gamma,L_1} = 0$.

We see that the distance between $\mathcal{Q}$ and $\mathcal{Q} \cap \mathcal{I}$ vanishes when $\mathcal{Q}$ is actually a subset of $\mathcal{I}$; this affirms the fact that both correlograms and histograms are global features. Such a *stable* property is not satisfied by all features – for instance, spatial coherence [PZ96] is not preserved under subset operations. Therefore, methods for subregion querying based on such unstable features are not likely to perform as good as the histogram or correlogram based methods.

### 4.2.1 Performance Measure

We use the following measures to evaluate the performance of various competing image subregion querying algorithms. If $\mathcal{Q}_1, \ldots, \mathcal{Q}_q$ are the query images, and for the $i$-th query $\mathcal{Q}_i$, $\mathcal{I}_1^{(i)}, \ldots, \mathcal{I}_{a_i}^{(i)}$ are the only images that "contain" $\mathcal{Q}_i$, (i.e., $\mathcal{Q}_i$ "appears in" $\mathcal{I}_j^{(i)}, j = 1, \ldots, a_i$,) yet due to the presence of false matches the image subregion querying algorithm may return this set of "answers" with various ranks.

1. *Average r-measure* gives the mean rank of the answer-images averaged over all queries. It is given by either of the following expressions:

$$\frac{1}{q}\sum_{i=1}^{q}\frac{1}{a_i}\sum_{j=1}^{a_i}\mathrm{rank}(\mathcal{I}_j^{(i)}) \tag{4.5}$$

$$(\sum_{i=1}^{q}\sum_{j=1}^{a_i}\mathrm{rank}(\mathcal{I}_j^{(i)}))/\sum_{i=1}^{q}a_i \tag{4.6}$$

   The *macroaveraged* $r$-measure given by Equation 4.5 treats all queries with equal importance, whereas the *microaveraged* $r$-measure defined by Equation 4.6 gives greater weightage to queries that have a larger number of answers. In both cases a lower value of the $r$-measure indicates better performance.

2. *Average precision* for a query $\mathcal{Q}_i$ is given by $(1/a_i)\sum_{j=1}^{a_i} j/\mathrm{rank}(\mathcal{I}_j^{(i)})$, where $\mathcal{I}_1^{(i)}, \ldots, \mathcal{I}_{a_i}^{(i)}$ are the answers for query $\mathcal{Q}_i$ in the order that they were retrieved. This quantity gives the average of the precision values over all recall points (with $1.0$ being perfect performance).

3. *Recall/Precision vs. Scope*: For a query $\mathcal{Q}_i$ and a *scope* $s > 0$, the *recall* $r$ is defined as $|\{\mathcal{I}_j^{(i)} \mid \mathrm{rank}(\mathcal{I}_j^{(i)}) \leq s\}|/a_i$, and the *precision* $p$ is defined as $|\{\mathcal{I}_j^{(i)} \mid \mathrm{rank}(\mathcal{I}_j^{(i)}) \leq s\}|/s$. These measures are simpler than the traditional average precision measure but still evaluate the effectiveness of the subregion query retrieval. For both measures, higher values indicate better performance.

### 4.2.2 Experiments

The image database is the same as *IC14* in Section 3.7. We use 64 color bins for histograms and autocorrelograms. The distance set for autocorrelograms is $D = \{1, 3, 5, 7\}$. Our query set for this task consists of 30 queries. Queries have 2 to 16 answer images with the average number of answers per query being nearly $5$. The query set is constructed by selecting "interesting" portions of images from the image database. The answer images contain the object depicted in the query, but often with its appearance significantly changed due to changes in viewpoint, or different lighting, etc. Examples of some queries and answers (and the rankings according to the histogram and autocorrelogram intersection methods) are shown in Figure 4.1 and Figure 4.2.

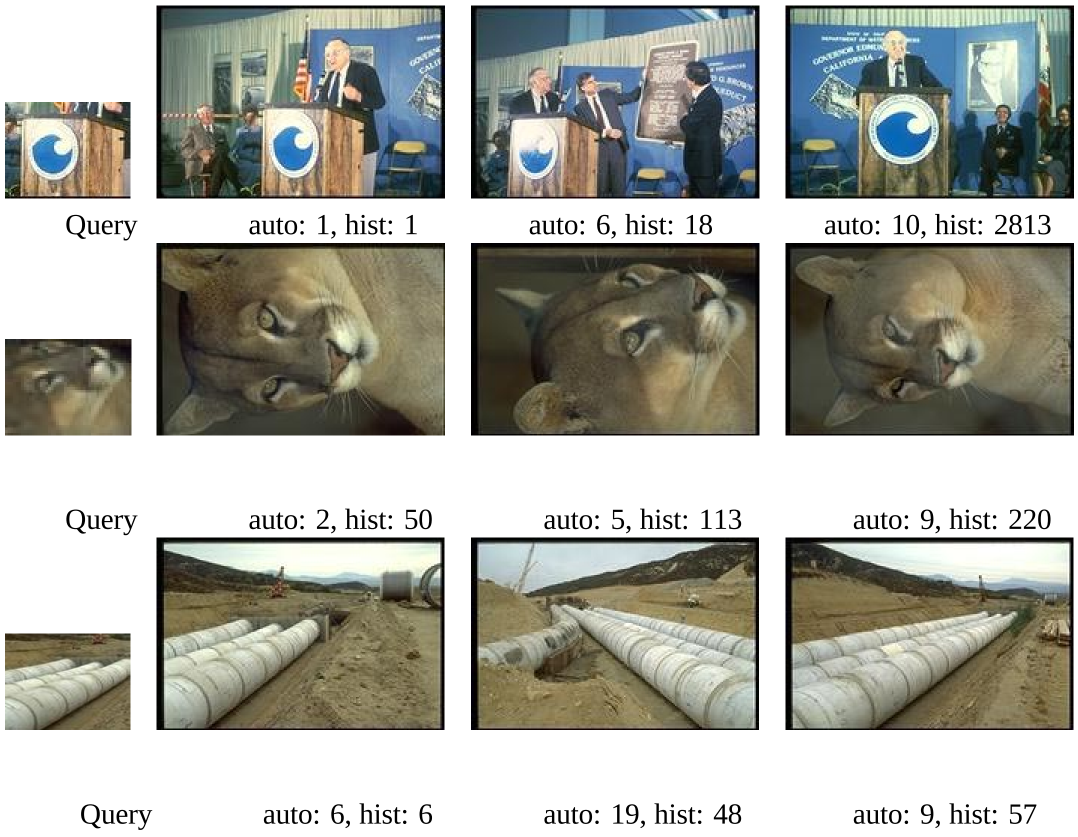


Figure 4.1: Sample queries and answer sets with ranks for various methods. (Lower ranks are better.)

### 4.2.3 Results

The histogram and autocorrelogram intersection methods for subregion querying are compared in Tables 4.1 and 4.2. For each of the evaluation measures proposed above, the autocorrelogram performs better. The average rank of the answer images improves by over 30 positions when the autocorrelogram method is used, and the average precision figure improves by an impressive 56% (see Table 4.1). Table 4.2 shows the precision and recall values for the two methods at various scopes. Once again, autocorrelograms perform consistently better than histograms at all scopes. Doing a query-by-query analysis, we find that autocorrelograms do better in terms of the average $r$-measure on 23 out of the 30 queries. Similarly, autocorrelograms yield better average precision on 26 out of 30 queries. Thus, for a variety of performance metrics, autocorrelograms yield better results. This suggests that the autocorrelogram is a significantly superior method for subregion querying problem.

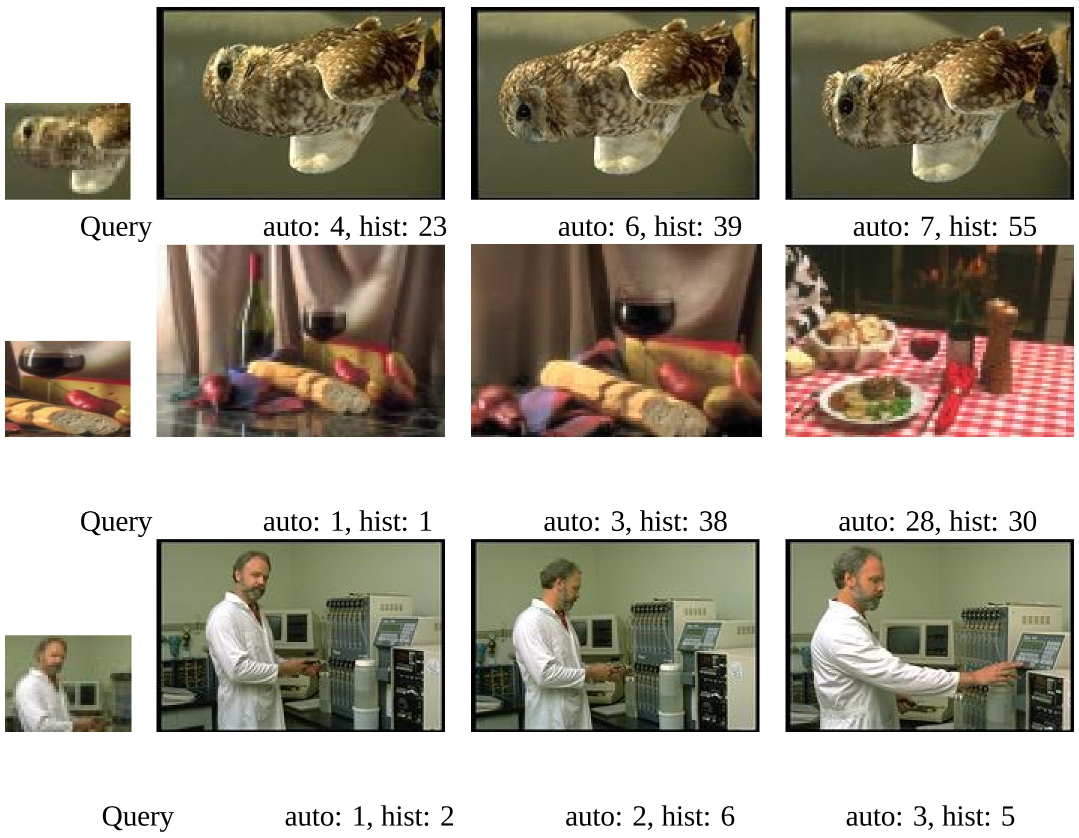


Figure 4.2: Sample queries and answer sets with ranks for various methods. (Lower ranks are better.)

## 4.3 Localization using Correlograms

The localization (or location) problem is the following: given a query image (also referred to as the target or model) $\mathcal{Q}$ and an image $\mathcal{I}$ such that $\mathcal{Q} \subseteq \mathcal{I}$, find the "location" in $\mathcal{I}$ where $\mathcal{Q}$ is "present". It is hard to define the notion of location mathematically because the model is of some size. We use the location of the center of the model for convenience.

This problem arises in tasks such as real-time *object tracking* or *video searching*, where it is necessary to localize the position of an object in an image. Given an algorithm that solves the location problem, tracking an object $\mathcal{Q}$ in an image frame sequence $\vec{\mathcal{I}} = \mathcal{I}_1, \ldots, \mathcal{I}_t$ is equivalent to finding the location of $\mathcal{Q}$ in each of the $\mathcal{I}_i$'s. Efficiency is also required in this task because huge amounts of data need to be processed.

The location problem can be viewed as a special case of the image retrieval problem in the following manner. Let $\mathcal{I}|_p$ denote the subimage in $\mathcal{I}$ of size $\mathcal{Q}$ located at position $p$. (The assumption about the size of the subimage is without loss of generality.) The set of all subimages $\mathcal{I}|_1, \ldots, \mathcal{I}|_{|\mathcal{I}|}$ present in $\mathcal{I}$ constitutes the image database and $\mathcal{Q}$ is the query image. The solution $\mathcal{I}|_p$ to this retrieval problem gives $p$, the location of $\mathcal{Q}$ in $\mathcal{I}$. The above interpretation is a *template matching* process.

Table 4.1: Performance of Histogram and Autocorrelogram Intersection methods – I. (Smaller $r$-measure and larger precision are better.)

| Method | Avg. $r$-measure (macro) | Avg. $r$-measure (micro) | Avg. precision |
|---|---|---|---|
| Hist | 56.3 | 61.3 | 0.386 |
| Auto | 22.5 | 29.1 | 0.602 |

Table 4.2: Performance of Histogram and Autocorrelogram Intersection methods – II. (Larger values are better.)

| Method $\downarrow$ | Precision(%) | | | Recall | | |
|---|---|---|---|---|---|---|
| Scope $\rightarrow$ | 5 | 10 | 25 | 5 | 10 | 25 |
| Hist | 0.273 | 0.223 | 0.133 | 0.311 | 0.460 | 0.681 |
| Auto | 0.493 | 0.347 | 0.165 | 0.541 | 0.718 | 0.850 |

One straightforward approach to the location problem is using this template matching process. Template matching takes the query $\mathcal{Q}$ as a template and moves this template over all possible locations in the image $\mathcal{I}$ to find the best match. This method is likely to yield good results when the object appearance does not change much. When the object appearance has large changes due to different view positions that cause scale and shape changes, occlusions etc, special techniques have to be applied to deal with those image appearance changes (such as *deformable* template matching [JZL96, JDL96, IWYS96]). In addition, template matching is computationally expensive. Attempts have been made to make template matching more efficient [MR90, VMH96, BE92]. The *histogram backprojection* method is one such approach to this problem. We discuss this approach below.

The basic idea behind histogram backprojection is (i) to compute a "goodness value" for each pixel in $\mathcal{I}$ (the goodness of each pixel is the likelihood that this pixel is in the target); and (ii) obtain the subimage (and hence the location) whose pixels have the highest goodness values.

Formally, the method can be described as follows. The *ratio histogram* is defined for a color $c$ as

$$\pi_{c,h}(\mathcal{I}|\mathcal{Q}) \triangleq \min\{\frac{H_c(\mathcal{Q})}{H_c(\mathcal{I})}, 1\}.$$

The goodness of a pixel $p \in \mathcal{I}_c$ is defined to be $\pi_{c,h}(\mathcal{I}|\mathcal{Q})$. The contribution of a subimage $\mathcal{I}|_p$ is given by

$$\Pi_p(\mathcal{I}|\mathcal{Q}) \triangleq \sum_{q \in \mathcal{I}|_p} \pi_{\mathcal{I}(q),h}(\mathcal{I}|\mathcal{Q}). \tag{4.7}$$

Then, the location of the model is given by

$$\arg\max_{p \in \mathcal{I}} \Pi_p(\mathcal{I}|\mathcal{Q}).$$

The above method generally works well in practice, and is insensitive to changes of image resolution or histogram resolution [SB91]. Note that back-projecting the ratio-histogram gives the *same* goodness value to all pixels of the same color. It emphasizes colors that appear frequently in the query but not too frequently in the image. This could result in overemphasizing certain colors in $\mathcal{Q}$. A color $c$ is said to be *dominant* in $\mathcal{Q}$, if $\pi_{c,h}(\mathcal{I}|\mathcal{Q})$ is maximum over all colors. If $\mathcal{I}$ has a subimage $\mathcal{I}|_p$ (which may be totally unrelated to $\mathcal{Q}$) that has many pixels of color $c$, then this method tends to identify $\mathcal{Q}$ with $\mathcal{I}|_p$, thus causing an error in some cases. Figure 4.3 shows

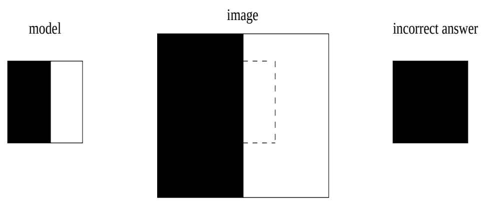


Figure 4.3: False match of histogram-backprojection.

a simple example illustrating this problem. Suppose $\mathcal{Q}$ has 6 black pixels and $4$ white pixels, and image $\mathcal{I}$ has $100$ black pixels and $100$ white pixels. Then $\pi_{b,h} = 0.06$, $\pi_{w,h} = 0.04$. The location of the model according to the backprojection method is in an entirely black patch, which is clearly wrong.

Another problem with histogram backprojection is inherited from histograms which have no spatial information. Pixels of the same color have the same goodness value irrespective of their position. Thus, false matches occur easily when there are multiple similarly colored objects, as shown in the examples of red roses in Figure 4.4 and zebras in Figure 4.5.

### 4.3.1 Correlogram Correction

To alleviate the problems with histogram backprojection, we incorporate local spatial correlation information by using a *correlogram correction* factor in Equation 4.7. The idea is to integrate discriminating local characteristics while avoiding local color template matching [EM95]. We define a *local correlogram contribution* based on the autocorrelogram of the subimage $\mathcal{I}|_p$ so that the goodness of a pixel depends on its position in addition to its color.

For each pixel $p \in \mathcal{I}$, the local autocorrelogram $\alpha_p^{(k)}$ is computed for each distance $k \in D$ ($D$ should contain only small values so that $\alpha_p^{(k)}$ captures local information for each $p$)

$$\alpha_p^{(k)} = \frac{\Gamma^{(k)}_{\mathcal{I}(p),\mathcal{I}(p)}(\{p\})}{8k}$$

where $\{p\}$ represents the pixel $p$ along with its neighbors considered as an image. Now, the correlogram contribution of $p$ is defined as

$$\pi_{p,\gamma}(\mathcal{I}|\mathcal{Q}) \stackrel{\Delta}{=} |\mathcal{Q}_{\mathcal{I}(p)} - \{p\}|_{\gamma,L_1}$$

In other words, the contribution of $p$ is the $L_1$-distance between the local autocorrelogram at $p$ and the part of the autocorrelogram for $\mathcal{Q}$ that corresponds to the color of $p$.

Combining this contribution with Equation 4.7, the final goodness value of a subimage $\mathcal{I}|_p$ is given by

$$\Pi_p(\mathcal{I}|\mathcal{Q}) \stackrel{\Delta}{=} \sum_{q \in \mathcal{I}|_p} \Big(\beta \pi_{\mathcal{I}(q),h}(\mathcal{I}|\mathcal{Q}) + (1-\beta)\pi_{q,\gamma}(\mathcal{I}|\mathcal{Q})\Big) .$$

where $0 \leq \beta \leq 1$.

It turns out that the correlogram contribution by itself is also sensitive and occasionally overemphasizes less dominant colors. Suppose $c$ is a less dominant color (say, background color) that has a high autocorrelation. If $\mathcal{I}$ has a subimage $\mathcal{I}|_p$ (which may be totally irrelevant to $\mathcal{Q}$) that has many pixels of color $c$ with high autocorrelations, then correlogram backprojection has a tendency to identify $\mathcal{Q}$ with $\mathcal{I}|_p$, thus potentially causing an error. Since the problems with histograms and correlograms are in some sense complementary to each other, the best results are obtained when the goodness of a pixel is given by a weighted linear combination of the histogram and correlogram backprojection contributions. Adding the local correlogram contribution to histogram backprojection remedies the problem that histograms do not take into account any local information, while on the other hand, the histogram contribution ensures that background colors are not overemphasized. We call this *correlogram correction*.

This can also be understood by drawing an analogy between this approach and Taylor expansion. The goodness value obtained from histogram backprojection is like the average constant value in the Taylor expansion and the local correlogram contribution is like the first order term in the approximation. Therefore, the best results are obtained when the goodness value of a pixel is a weighted linear combination of the histogram backprojection value and the correlogram contribution.

### 4.3.2 Performance Measure

Let an indicator variable $\operatorname{loc}(\mathcal{Q}, \mathcal{I})$ be 1 if the location returned by a method is within reasonable tolerance of the actual location of $\mathcal{Q}$ in $\mathcal{I}$. Then, given a series of queries $\mathcal{Q}_1, \ldots, \mathcal{Q}_q$ and corresponding images $\mathcal{I}_1, \ldots, \mathcal{I}_q$, the *success ratio* of the method is given by

$$\frac{\sum_{i=1}^{q} \operatorname{loc}(\mathcal{Q}_i, \mathcal{I}_i)}{q}$$

For tracking an object $\mathcal{Q}$ in a sequence of frames $\vec{\mathcal{I}} = \mathcal{I}_1, \ldots, \mathcal{I}_t$, the *success ratio* is therefore $\sum_{i=1}^{t} \operatorname{loc}(\mathcal{Q}, \mathcal{I}_i)/t$.

### 4.3.3 Experiments and Results

We use the same database to perform the location experiments. A model image and an image that contains the model are chosen. For the location problem, 66 query images and 52 images that contain these models are chosen and tested. Both the histogram backprojection and autocorrelogram correction are tried. $D = \{1, 3, 5\}$ and $\beta = 0.5$ were the parameters.

For the tracking problem, we choose three videos: bus(133 frames), clapton (44 frames), and skydive (85 frames). We use $D = \{1, 3, 5\}$ and $\beta = 0.8$ for this problem.

For the location problem, Table 4.3 shows the results for 66 queries, and Figure 4.4 and Figure

Table 4.3: Results for location problem (66 queries).

| Method | hist | auto |
|---|---|---|
| Success Ratio | 0.78 | 0.96 |

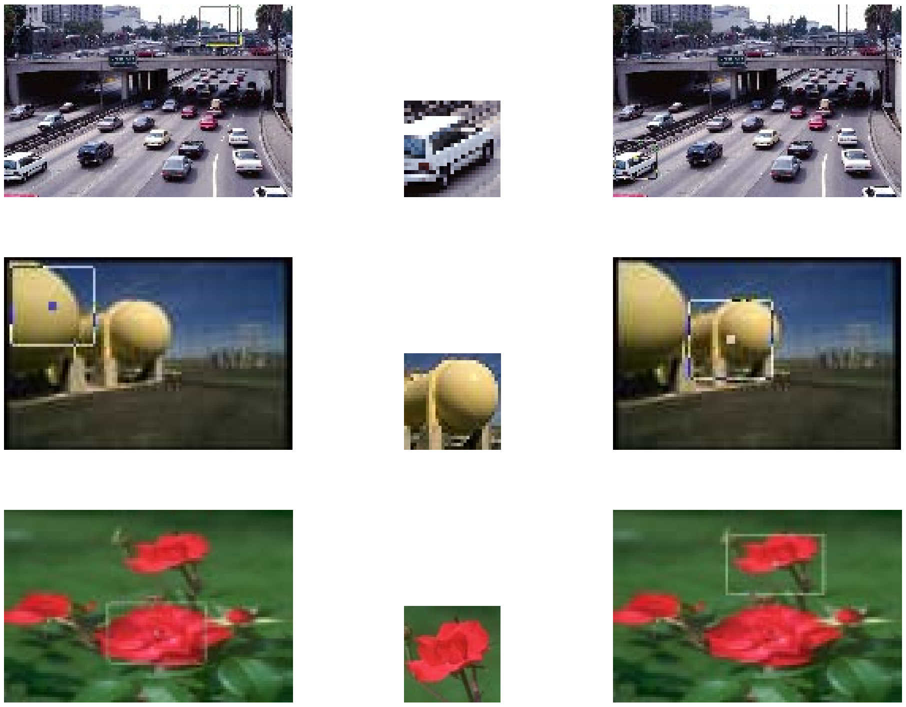


Figure 4.4: Location problem: histogram output, query image, and correlogram output.

4.5 shows some examples for the location problem. For the tracking problem, Table 4.4 shows the result of histogram backprojection and correlogram correction for the three test videos. These results clearly show that correlogram correction alleviates many of the problems associated with simple histogram backprojection. Figure 4.6 shows sample outputs.

Table 4.4: Results (success ratios) for the tracking problem.

| Method | hist | auto |
|---|---|---|
| bus | 0.93 | 0.99 |
| clapton | 0.44 | 0.78 |
| skydive | 0.96 | 0.96 |

## 4.4 Summary

Image subregion querying and localization are vital tools that are indispensable feature of any image management system. We propose the use of color correlograms in devising algorithms for these problems. We introduce the concept of correlogram intersection (as a generalization of histogram intersection) and correlogram correction. Our algorithms are very simple and inexpensive. Experiments suggest that with little more computation and storage, correlograms achieve better performance for image subregion querying and localization than histograms.

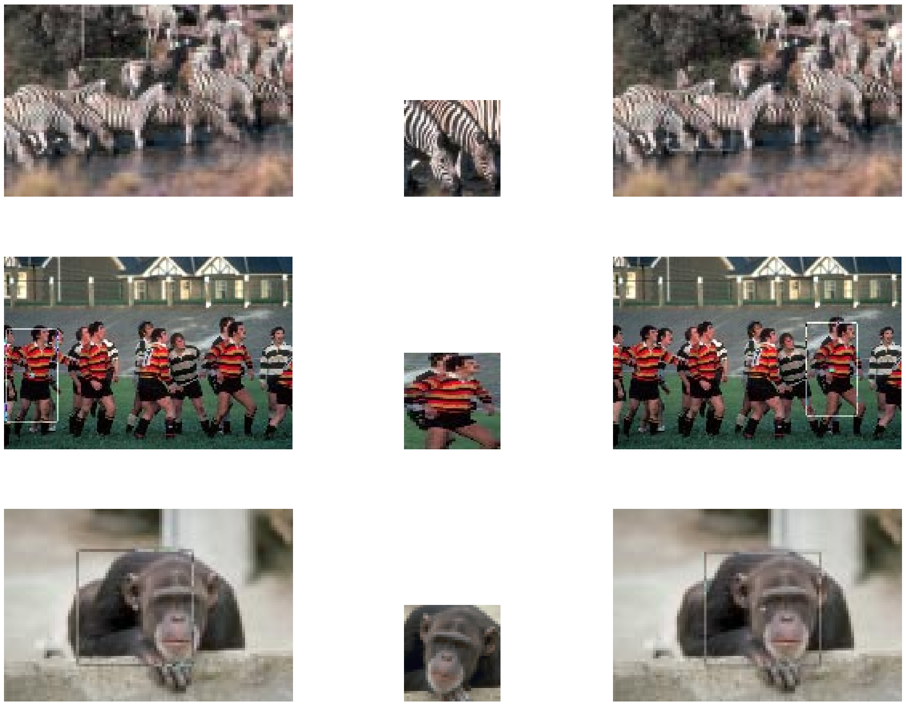


Figure 4.5: Location problem: histogram output, query image, and correlogram output.

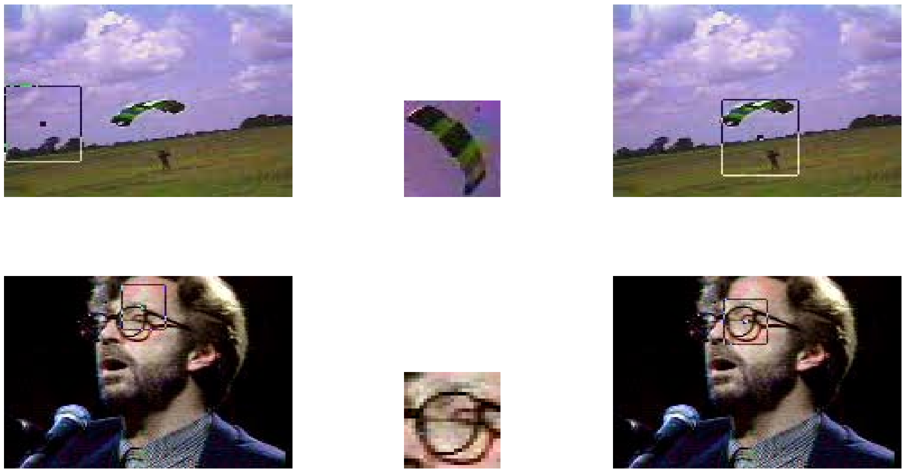


Figure 4.6: Tracking problem: histogram output, query image, and correlogram output.

# Chapter 5

# Cut Detection

In earlier chapters, we focused on the use of correlograms for still-image related problems. In this chapter, we show that correlograms can also be used for some problems in the video domain. In particular, we consider the fundamental problem of detecting cuts in video sequences and show that correlograms can be applied to this problem with reasonable success.

## 5.1 Background

The increasing amount of video data requires automated video analysis. The first step to the automated video content analysis is to segment a video into camera shots (also known as *key frame extraction*. See Figure 5.1). A camera shot $\vec{\mathcal{I}} = \mathcal{I}_1, \ldots, \mathcal{I}_t$ is an unbroken sequence of frames from one camera. If $\vec{\mathcal{J}}$ denotes the sequence of cuts, then a cut $\mathcal{J}_j$ occurs when two consecutive frames $\langle \mathcal{I}_i, \mathcal{I}_{i+1} \rangle$ are from "different" shots.

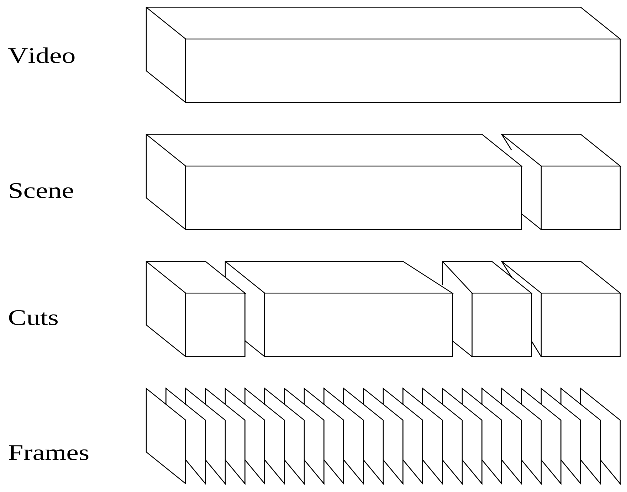


Figure 5.1: Video parsing.

Cut detection algorithms assume that consecutive frames in a same shot are somewhat more *similar* than frames in a different shot. (Other gradual transitions such as fade and dissolve are not studied here because certain mathematical models can be used to treat these special-case chromatic editing effects.) Different cut detectors use different features to compare the similarity between two consecutive frames, such as pixel difference, statistical differences, histogram comparisons, edge differences, etc [BR96]. One way to detect cuts using a feature $f$ is by ranking $\langle \mathcal{I}_i, \mathcal{I}_{i+1} \rangle$ according to $|\mathcal{I}_i - \mathcal{I}_{i+1}|_f$. Let $\mathrm{cuts}(\vec{\mathcal{I}})$ be the number of actual cuts in $\vec{I}$ and $\mathrm{rank}(\mathcal{J}_i)$ be the rank of the cut $\mathcal{J}_i$ according to this ranking.

Histograms are the most common used image features to detect cuts because they are efficient to compute and insensitive to camera motions. Histograms, however, are not stable to tolerate local changes in images that false positives easily occurs in this case (see Figure 5.2). Since correlograms seem to be stable to tolerate large appearance changes for image retrieval, we use correlograms for cut detection.

**Performance Measure.** *Recall* and *precision* are usually used to compare the performance of cut detection. However, it is difficult to measure the performance of different algorithms based on recall vs. precision curves [BR96]. Therefore we look at recall and precision values separately. In order to avoid using "optimal" threshold values, we use *precision vs. scope* to measure false positives and *recall vs. scope* to measure false negatives (misses). We choose scope values to be the exact cut number $\mathrm{cuts}(\vec{I})$ and $2\,\mathrm{cuts}(\vec{I})$. We also use the *excessive rank value*, which is defined by

$$\sum_{i=1}^{\mathrm{cuts}(\vec{I})} \left(\mathrm{rank}(\mathcal{J}_i) - i\right),$$

and the *average precision value* which is defined by

$$\frac{1}{\mathrm{cuts}(\vec{I})} \sum_{i=1}^{\mathrm{cuts}(\vec{I})} \frac{i}{\mathrm{rank}(\mathcal{J}_i)}.$$

Note that a smaller excessive rank value and a larger average precision value indicate better result (perfect performance would have values zero and one respectively).

## 5.2 Experiments and Results

We use 64 colors for histograms and banded autocorrelograms. We use 5 video clips from television, movies, and commercials. The clips are diverse enough to capture different kinds of common scenarios that occur in practice. The results are shown in Table 5.1 and Table 5.2.

Table 5.1: Recall vs. Scope for cut detection. (Smaller values are better.)

| Method $\rightarrow$ | hist | | | banded-auto | | |
|---|---|---|---|---|---|---|
| Video $\downarrow$ | $\mathrm{cuts}(\vec{\mathcal{I}})$ | $2\,\mathrm{cuts}(\vec{\mathcal{I}})$ | Ex. rank | $\mathrm{cuts}(\vec{\mathcal{I}})$ | $2\,\mathrm{cuts}(\vec{\mathcal{I}})$ | Ex. rank |
| 1 (9 cuts) | 0.78 | 1.0 | 14 | 1.0 | 1.0 | 0 |
| 2 (15 cuts) | 0.87 | 1.0 | 7 | 1.0 | 1.0 | 0 |
| 3 (16 cuts) | 0.75 | 1.0 | 25 | 0.81 | 1.0 | 9 |
| 4 (7 cuts) | 1.0 | 1.0 | 0 | 1.0 | 1.0 | 0 |
| 5 (10 cuts) | 0.9 | 0.9 | $> 10$ | 0.9 | 1.0 | 3 |

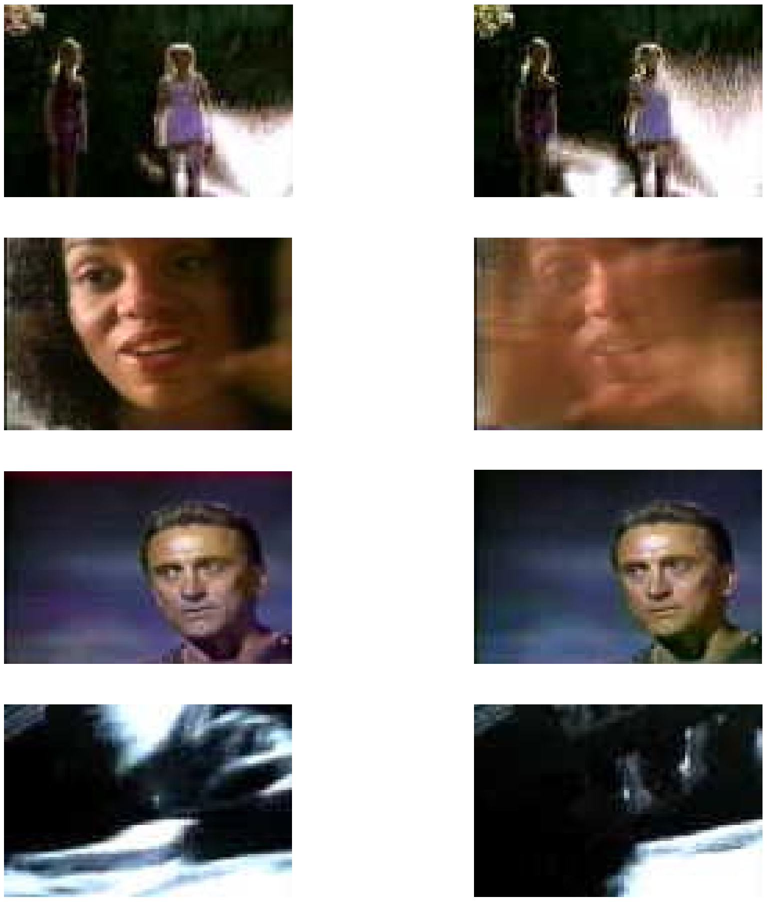


Figure 5.2: Cut detection: False cuts detected by histogram but not by correlogram.

The results of our experiments show that banded autocorrelograms are more accurate for cut detection than histograms despite both having the same amount of information. It is certainly more efficient than dividing an image into 16 subimages [HJW94]. Thus the autocorrelogram seems to be a promising feature for cut detection.

## 5.3 Summary

Cut detection is the initial step to parse a video sequence. Our experiments show that color correlograms perform better results than color histograms for in detecting abrupt scene changes. This is because the correlogram tolerates the large local image change while the histogram fails. This is not surprising because we have shown that correlograms are much more stable to tolerate large image appearance changes in Chapter 3.

Table 5.2: Precision vs. Scope for cut detection. (Larger values are better.)

| Method $\rightarrow$ | hist | | | banded-auto | | |
|---|---|---|---|---|---|---|
| Video $\downarrow$ | $\mathrm{cuts}(\vec{\mathcal{I}})$ | $2\,\mathrm{cuts}(\vec{\mathcal{I}})$ | Avg. Prec. | $\mathrm{cuts}(\vec{\mathcal{I}})$ | $2\,\mathrm{cuts}(\vec{\mathcal{I}})$ | Avg. Prec. |
| 1 (9 cuts) | 0.78 | 0.5 | 0.90 | 1.0 | 0.5 | 1.0 |
| 2 (15 cuts) | 0.87 | 0.5 | 0.98 | 1.0 | 0.5 | 1.0 |
| 3 (16 cuts) | 0.75 | 0.5 | 0.93 | 0.81 | 0.5 | 0.97 |
| $4$ (7 cuts) | 1.0 | 0.5 | 1.0 | 1.0 | 0.5 | 1.0 |
| $5$ (10 cuts) | 0.8 | 0.45 | $< 0.95$ | 0.9 | 0.5 | 0.98 |

# Chapter 6

# Supervised Learning in Content-Based Image Retrieval

We have already shown that the color correlogram is a stable, scalable, effective, and efficient image feature for content-based image retrieval. It is, however, impossible for any image feature to be entirely fool-proof. Therefore, for any query image, not all the images labeled “relevant” by any retrieval method are actually relevant. It is reasonable in such situations to enlist the services of the user by requesting for some relevance feedback. For instance, in practical situations like searching for a particular image on the world-wide web, the system can retrieve a list of images that it considers relevant to the query image. Then, the user can be prompted to mark each retrieved image as “relevant” or “irrelevant”. This information can be used to refine the search, thus potentially improving the quality of retrieval.

In this chapter, we investigate the use of information provided interactively by a user to improve the performance of correlograms. We outline two schemes that use feedback information (in the form of labeled examples). The first scheme is borrowed from the standard technique in information retrieval [Salt89]. Learning is effected by interpolating the query vector with feature vectors of positive examples. The second scheme is based on rearranging the feature space by the query and the positive/negative examples. By learning a weighted metric, it is possible to selectively enhance (resp. suppress) the role of appropriate dimensions to retrieve images of the kind termed positive (resp. negative).

**Organization.** Section 6.1 discusses some related work. Then two learning methods are presented in Section 6.2. The experimental framework (Section 6.3) and results (Section 6.4) are presented next.

## 6.1 Related Work

Some work has been done on how to improve the performance of image retrieval by learning from user feedback. In PicHunter [CMOY96], relevance feedback from the user is used to search the target. This approach is based on a Bayesian framework that incorporates an explicit model of the user selection process. They show that the feedback system works much better than random chance in a queryless setting. They also conduct a user-study of the performance of their system.

Minka and Picard [MP96, PM95] introduce a learning component in their system by using positive and negative examples which lets the system choose image groupings within and across images based on color and texture cues. The user-chosen region defines the positive or negative examples which the system tries to generalize and label various parts of the scene. In [Dela96], positive and negative examples are used to first construct histograms. Then, a scoring function is constructed from a comparison of the extent of overlap between the histograms. This scoring function is aimed to best partition the examples and counter-examples.

Ma [Ma97] uses labeled samples to learn visually similar patterns. Learning is achieved by exploiting neural networks to partition the feature space into clusters of similar patterns, each pattern with a known label. The learning scheme is test on a texture database with $116$ images. The results suggest that the learning scheme improve the recall in the scope of $20 - 60$ images.

Rui *et al.* [RHMO97] use a two-layer model for a multimedia object. The top layer contains different image features and their corresponding distance functions; the bottom layer contains the actual values of these features. Relevance feedback is learned in the two layers separately. For the top layer, the initial retrieval results come from overall results from different features and distance functions. Users are supposed to provide a rank list of the initial results based on their perception criteria. Then the best feature and distance function pair is found by finding the smallest difference between the initial ranks and users' ranks. Based on this best pair of feature and distance function, a new retrieval is returned. Although this approach seems to find the best, it assumes that users judge their ranks based on only one feature. This is not true in general. Therefore the refined retrieval may not be as good as the initial one. For the bottom layer, they used the standard techniques in information retrieval.

## 6.2 Two Learning Methods

In this section, we look at the issue of using relevance feedback from the user to improve the performance of the correlogram. In a real-life scenario, the user can provide this information with very little effort. Our goal is to make use of this information to improve search results, so that the user has an incentive to make this small additional effort.

Let $\mathcal{F} = \mathcal{F}^+ \cup \mathcal{F}^-$ be the set of images for which the user provides relevance judgments, where $\mathcal{F}^+ = \{\mathcal{F}_1^+, \ldots, \mathcal{F}_{|\mathcal{F}^+|}^+\}$ (resp. $\mathcal{F}^- = \{\mathcal{F}_1^-, \ldots, \mathcal{F}_{|\mathcal{F}^-|}^-\}$) is the set of positive (resp. negative) examples.

### 6.2.1 Learning the Query Vector

The motivation for this method (called QLEARN) comes from the fact that feature vectors of similar images are usually geometrically close to each other in the feature space. Hence, a natural approach is to modify the feature vector of the query image $\mathcal{Q}$ to get a new feature vector which serves as a pseudo-query vector. One such approach involves taking a weighted average of the positive examples together with the query image. The modified feature vector is then given by

$$(1-\beta) \cdot \alpha(\mathcal{Q}) + \beta \cdot \frac{1}{|\mathcal{F}^+|} \sum_{i=1}^{|\mathcal{F}^+|} \alpha(\mathcal{F}_i^+)$$

Here, $\beta$ controls the contribution of the feedback images. Note that $\beta$ cannot be too large because there is the danger of "losing" the importance of the original query.

It is not obvious how to augment this method to handle images in $\mathcal{F}^-$. In the feature space, feature vectors of images in $\mathcal{F}^+$ are usually roughly clustered. The modification described above can be interpreted as moving the query vector closer to the cluster of positive examples. The corresponding way to use negative examples would be to move the query away from them. Top-ranked negative examples do not exhibit a similar clustering, however, and are usually scattered in a close neighborhood of the query. Thus, the notion of "moving away" from the negative examples is not well-defined.

### 6.2.2 Learning the Metric

The main idea for this method (called WLEARN) is to learn a weighted $L_1^\omega$ metric instead of the usual $L_1$ metric to compute the distance between feature vectors[1]. In this manner, we would be able to enhance the importance of those dimensions that "assist" in getting the similar images and diminish the importance of those dimensions that "hinder" this process.

More formally, the weighted $L_1^\omega$ metric for a given real weight vector $\omega \in R^{m\times d}$ is given by

$$|\mathcal{I} - \mathcal{I}'|_{\alpha, L_1^\omega} \triangleq \sum_{i\in[m],k\in D} \omega_{i,k} \cdot |\alpha_{c_i}^{(k)}(\mathcal{I}) - \alpha_{c_i}^{(k)}(\mathcal{I}')|$$

If $D^+ \subseteq [m] \times [d]$ is the set of dimensions that are deemed important, then using some weight update scheme, we can increase those weights $\omega_{i,j}$ for which $(i,j) \in D^+$. In an analogous manner, if $D^- = ([m] \times [d])\backslash D^+$, then we can decrease the weights $\omega_{i,j}$ for which $(i,j) \in D^-$. See Figure 6.1 for an example. The dimensions indicated by the solid (resp. dotted) line can be considered to be in $D^+$ (resp. $D^-$). One scheme to implement this idea is given below.

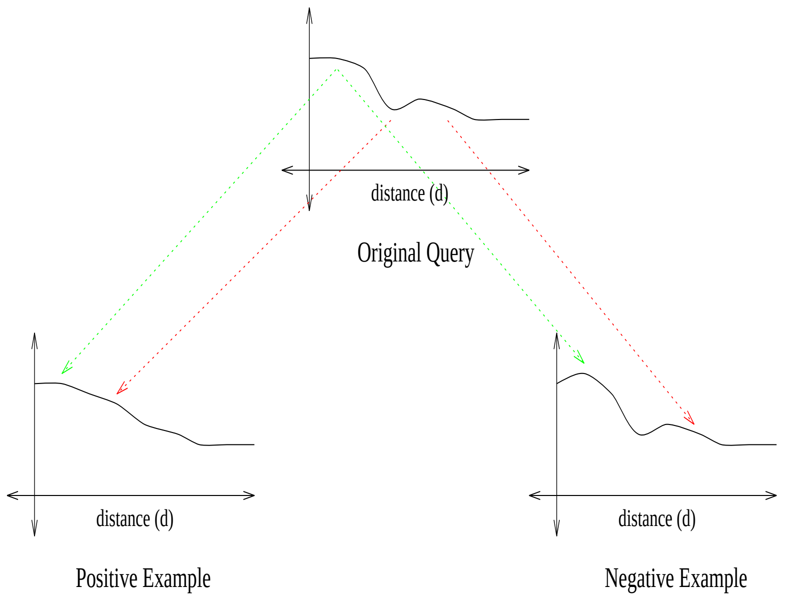


Figure 6.1: Learning the Metric.

Given $\mathcal{F}_i^+$, we analyze the difference vector $\delta = |\alpha(\mathcal{Q}) - \alpha(\mathcal{F}_i^+)|$. Since $\mathcal{F}_i^+$ is labeled positive, the dimensions that contribute larger quantities to $\delta$ can be considered to be less important. Similarly, the dimensions with lower contributions to $\delta$ can be considered to be more important.

[1]Metric learning is also applied to face recognition [CGY96]

This intuition is formalized as a simple heuristic update rule, which is given by

$$\omega_{i,j} = \omega_{i,j} \cdot (1 + \overline{\delta} - \delta_{i,j})$$

where $\overline{\delta}$ is the mean of the components of $\delta$.

In a similar manner, given $\mathcal{F}_i^-$, let $\delta = |\alpha(\mathcal{Q}) - \alpha(\mathcal{F}_i^-)|$. Since $\mathcal{F}_i^-$ is labeled negative, the dimensions that contribute larger quantities to $\delta$ can be considered to be more important. Similarly, the dimensions with lower contributions to $\delta$ can be considered to be less important. A simple update rule that implements this is given by

$$\omega_{i,j} = \omega_{i,j} \cdot (1 - \overline{\delta} + \delta_{i,j})$$

The weight update rule is applied for each example in $\mathcal{F}$. The final weight $\omega$ is used to compute the $L_1^\omega$ metric for subsequent queries.

## 6.3 Experiments

We conduct several experiments to investigate how learning can be used to improve search results. We focus mainly on the image retrieval task, but we also report some preliminary results for the object searching problem. (Another important aspect of studying feedback is doing a user study like PicHunter. The results presented here are from one user, who is an expert in image retrieval. Since the nature of feedback is very elementary in our case, we expect that most users will find it easy to provide such feedback.)

We use a database of 20,026 color JPEG images (*IC20*), obtained by adding $6000$ Corel images to *IC14*, for all our experiments. We consider the RGB color space with quantization into 64 colors. We use the same distance set $D = \{1, 3, 5, 7\}$ (so, $d = 4$) for computing the autocorrelograms.

**Performance Measure.** Let $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_q\}$ be the set of query images. For a query $\mathcal{Q}$, let $\mathcal{A}$ be the set of images that are similar to $\mathcal{Q}$. $\mathcal{A}$ is ordered by the metric on the feature vector. We use the following performance measures:

1. *Scope vs. Recall:* For a scope $s$, we count $|\{\mathcal{I} \in \mathcal{A} \mid \mathrm{rank}(\mathcal{I}) \leq s\}|$, i.e., the number of relevant images ranked below $s$.

2. *Average Precision:* Let $t$ be fixed *a priori*. Define $\mathcal{A}^{(t)} = \{\mathcal{I} \in \mathcal{A} \mid \mathrm{rank}(\mathcal{I}) \leq t\}$. Note that $\mathcal{A}^{(t)}$ is still ordered. Then, the average precision is given by
   $(1/|\mathcal{A}^{(t)}|) \sum_{\mathcal{I}_i \in \mathcal{A}^{(t)}} i/\mathrm{rank}(\mathcal{I}_i)$

For a given scope, the higher the recall, the better. A higher value of average precision also indicates better performance (the maximum value of average precision is one). Note that these definitions of the terms recall and average precision are slightly different from the standard ones. The above measures can be averaged over all the queries $\mathcal{Q}_1, \ldots, \mathcal{Q}_q$ to get a fair estimation of the performance of a method.

To evaluate performance, we use a maximum scope of $s = 50$ for the scope vs. recall measure and $t = 50$ for average precision. We use $\beta = 0.4$ for QLEARN. For the rest of the paper, "auto" refers to the autocorrelogram method with no feedback.

**Image Retrieval.** We first consider a scenario in which the user examines a small but fixed number of top-ranked images retrieved in response to the original query, and provides relevance judgments about them. Our experiments investigate whether this small additional effort on the part of the user can result in improvements in retrieval effectiveness.

We use a set of 29 queries for this experiment. For each query, a set of images is retrieved. The user examines the top 10 images and marks the ones that are relevant. This relevance information is used to modify the feature vector for the initial query or to modify the distance measure used to compare feature vectors. The database is searched again using the modified query vector or distance measure.

This scheme for providing user feedback turns out to be inadequate in some situations, however. If the query is a "difficult" one, then it is possible that few (or even none) of the top 10 images are useful. Thus, the system gets very little new positive information in the above approach. Further, several queries have answer images in which the appearance of the object of interest is significantly different from its appearance in the query. Such answers may not be ranked within the top 10. In order to get more of these answer images within the top-ranks, the user should be able to give positive feedback to the system about such images.

Accordingly, we conduct a second set of experiments in which the user marks the first 5 relevant and first 5 non-relevant images (irrespective of their ranks). Since some of the 29 queries used above had fewer than five answer images, we use a subset of 19 queries for this experiment.

In addition to using the query- and metric-learning techniques individually in this experiment, we study the effect of combining both techniques during the final search. The results for all these experiments are presented in the section on results.

**Object Searching.** We also conduct some preliminary experiments exploring the usefulness of learning techniques for the object-searching problem. An "interesting" object in an image is selected as the query and images are retrieved using the Correlogram Intersection method (see Section 4.2). The user then examines the top-ranked images (typically ten or less), and marks a few (typically three to five) of the useful images. The user also specifies the location of the object in these images (for example, by clicking on a point that approximately represents the center of the object in the image). The system uses this information to modify the query or the distance measure and a new search is conducted. No negative feedback is used in these experiments.

## 6.4 Results

Table 6.1 shows the effectiveness of the initial search (no feedback) and the results obtained using feedback for the top 10 images. For comparison, we also provide the average precision value for the initial search if the histogram is used as the image feature vector instead of the autocorrelogram. Figure 6.2 and Figure 6.3 shows some examples of some query images and ranks of similar images according to different methods.

Table 6.1: Retrieval effectiveness with and without feedback (top 10 images).

| | Histogram | Correlogram | | | |
|---|---|---|---|---|---|
| | hist | auto | QLEARN | WLEARN(+ only) | WLEARN |
| Average Precision | 0.425 | 0.547 | 0.661 | 0.627 | 0.631 |
| %-age Change | – | – | + 20.8 | + 14.6 | + 15.4 |

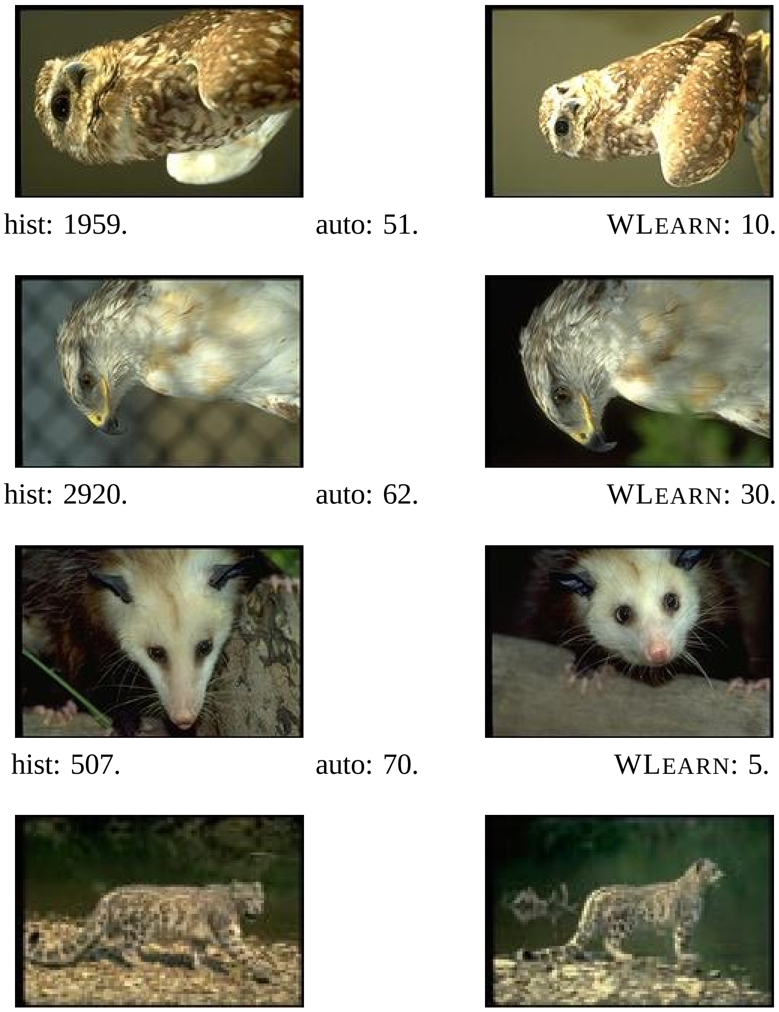


Figure 6.2: Comparison of various methods for image retrieval. (Lower numbers indicate better performance.)

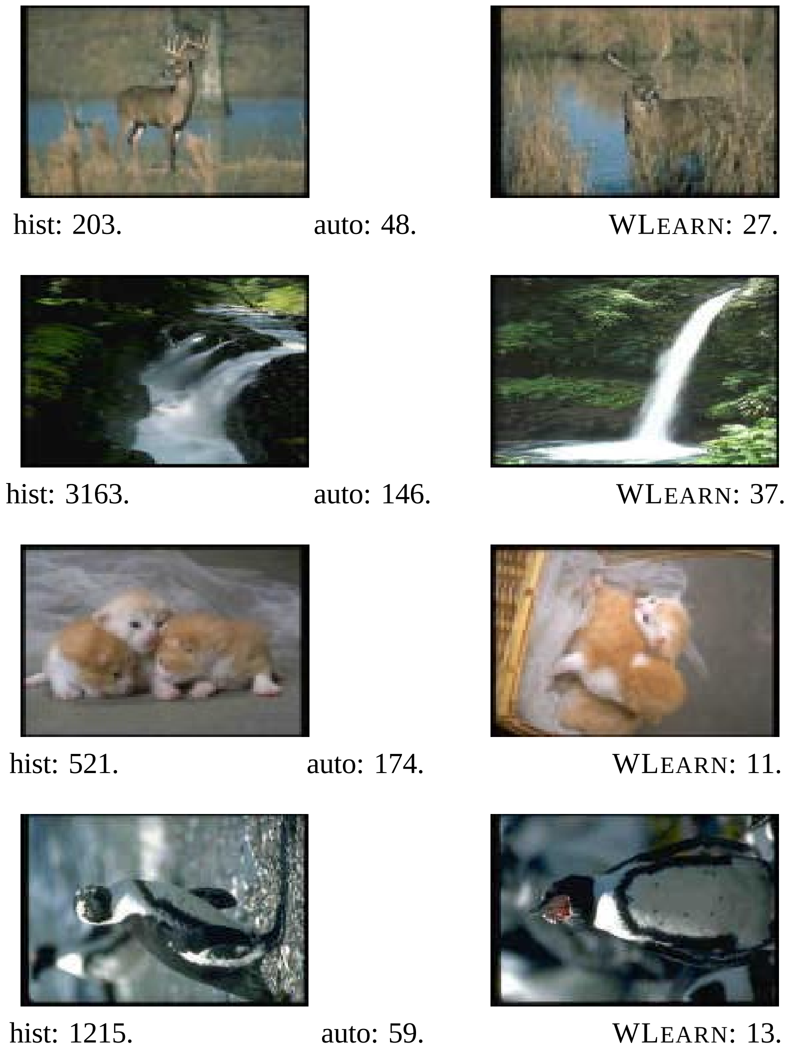


Figure 6.3: Comparison of various methods for image retrieval. (Lower numbers indicate better performance.)

For the initial retrieval, autocorrelograms significantly outperform histograms. This agrees with the observations above. When feedback is used, results improve by 15 to 20% over the initial retrieval[2]. The QLEARN and WLEARN approaches both appear to be effective methods for using feedback information. The improvements obtained using feedback are shown graphically in Figure 6.4. The number of relevant images retrieved by the different methods is plotted against the scope. From the graph, it is clear that the use of feedback improves performance — the curves for QLEARN and WLEARN lie consistently above the curve for the initial retrieval.

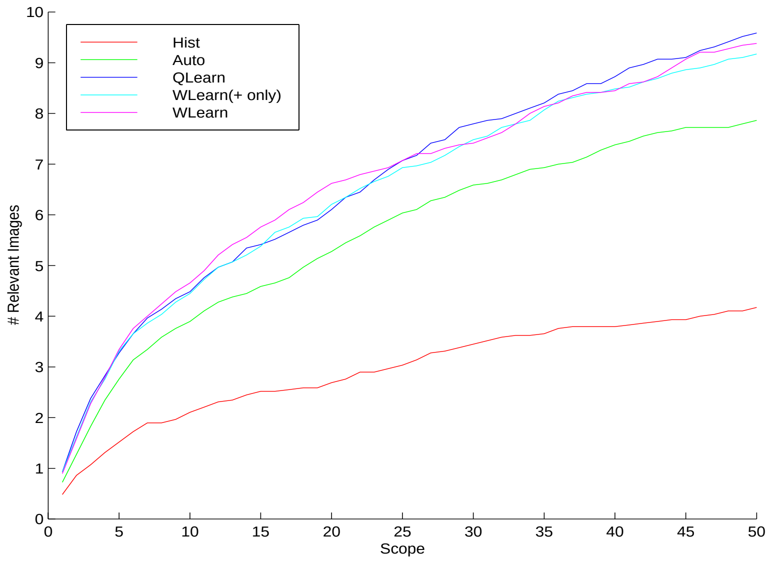


Figure 6.4: Graph showing number of relevant images retrieved at various scopes.

**Negative Examples.** We also investigate the usefulness of negative examples. First, only the images marked positive are used to modify the distance measure in the WLEARN method. Next, the user's feedback about all the ten images is used (i.e., both positive and negative). The results for these two approaches are shown in the last two columns in Table 6.1. From the table, it appears that the use of negative information is marginally more useful on the whole. A more detailed analysis shows, however, that there are queries for which WLEARN(+only) does better than WLEARN. Typically, if only one or two out of the top ten images are judged useful, the large number of negative examples could change the weights in the WLEARN method in undesirable ways, and the final result would deteriorate. The next set of experiments investigates a setting in which an equal number of positive and negative examples are used.

Figure 6.5 and Table 6.2 show the results of the second set of experiments. For this experiment, the user provides feedback about the first 5 relevant and first 5 non-relevant images. Once again, the use of feedback results in significant improvements. Since the QLEARN and WLEARN methods are in some sense independent and both seem to be effective, we investigate a combination of these methods. We call this method QWLEARN. Table 6.2 and Figure 6.5 suggest that

[2]This improvement in average precision is partly due to the improvement in ranks of the images that are marked relevant by the user. The improvement for previously unseen images is really what is more relevant. Figure 6.4 shows that this improvement is also substantial — see the increase in recall for scope $> 10$, for example.

Table 6.2: Retrieval effectiveness with and without feedback (first 5 positive and negative examples).

| | Histogram | Correlogram | | | |
|---|---|---|---|---|---|
| | hist | auto | QLEARN | WLEARN | QWLEARN |
| Average Precision | 0.379 | 0.613 | 0.690 | 0.710 | 0.766 |
| %-age Change | – | – | + 12.6 | + 15.8 | + 25.0 |

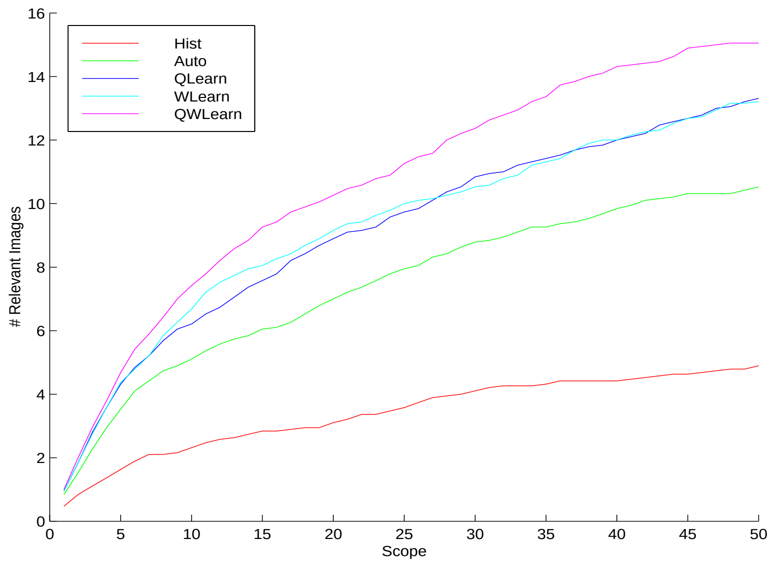


Figure 6.5: Graph showing number of relevant images retrieved at various scopes.

the two methods are indeed complementary, since the results for QWLEARN are significantly better than either method — the average precision for QWLEARN is about 8% better than that for WLEARN, and the curve for QWLEARN is consistently higher than the curves corresponding to QLEARN and WLEARN.

**Object Searching.** Figure 6.6 shows some sample queries and answers and their ranks before and after learning[3]. The examples suggest that a small amount of feedback from the user can improve results substantially. For instance, the image of the owl moves up from the 75th rank to the sixth position. The improvement in the rank of useful images usually implies that the number of useful images at a given scope also increases. The object-searching problem is a difficult one, however, and more experimentation is needed before reliable conclusions can be drawn about using learning methods for this problem.

[3]Note that these answers were not seen by the user at the end of the first pass. Of course, the ranks of the images that are judged relevant by the user are expected to improve.

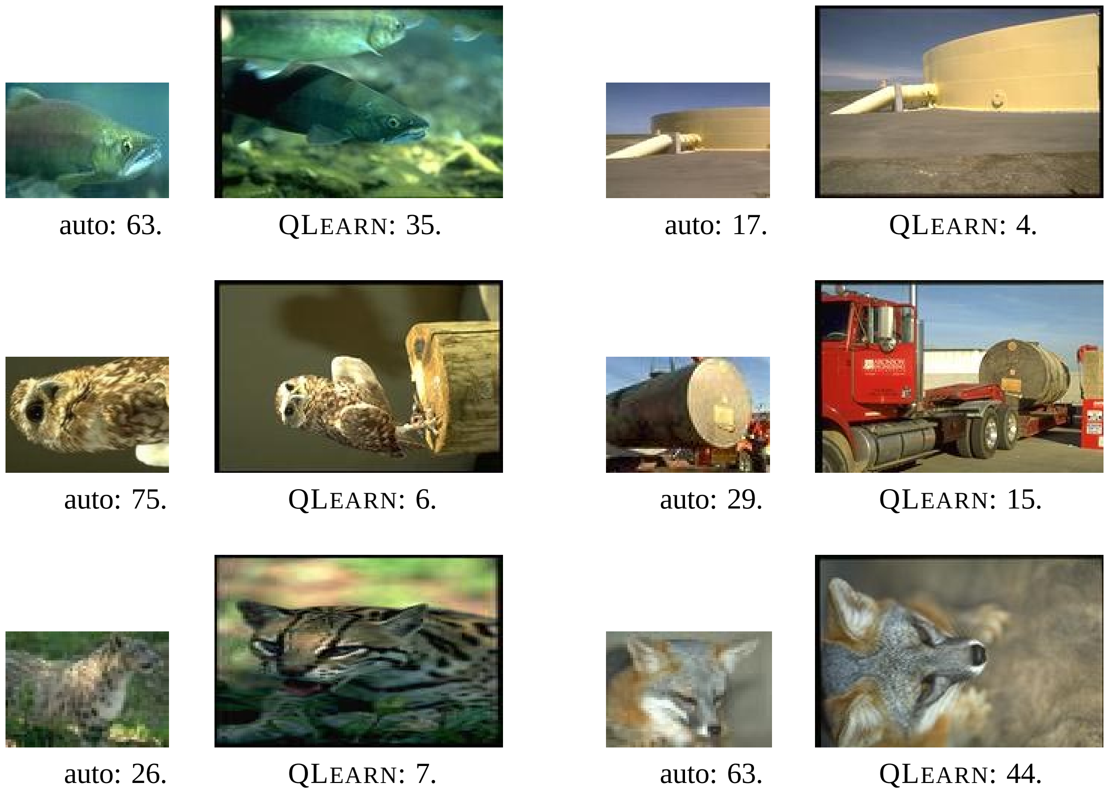


Figure 6.6: Comparison of various methods for object searching. (Lower numbers indicate better performance.)

## 6.5 Summary

We have addressed how relevance feedback can be used to improve the performance of content-based image retrieval. We present two supervised learning methods: *learning the query* and *learning the metric*. We combine the learning methods with *color correlograms* to improve the performance of content-based image retrieval. Our results on a large image database of over $20,000$ images suggest that these learning methods are quite effective for content-based image retrieval and image subregion querying.

# Chapter 7

# Image Classification

We have been mainly focusing on image retrieval issues so far. Users, however, are not only interested in searching for specific images or video shots. They would also like to browse and navigate through the image corpus. This task becomes much easier if the corpus were organized in a meaningful manner. One way to achieve this is via *image classification*. Image classification is the task of classifying images into (semantic) categories based on the available training data. This categorization of images into classes can be helpful both in semantic organizations of digital libraries and in obtaining automatic annotations of images.

The classification of natural imagery is quite hard in general, since real images from the same semantic class may have large variations (see Figure 7.1 and Figure 7.2) and images from different semantic classes might share a common background (such as images from "clouds" and "aviation", images from "waves" and "dolphins and whales"). These issues limit the applicability of object-based and knowledge-based approaches.

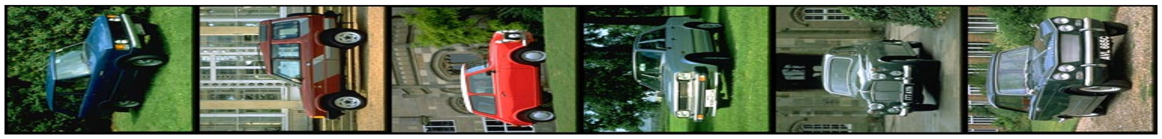
A common approach to image classification involves addressing the following three issues: (i) image features — how to *represent* the image, (ii) organization of feature data — how to *organize* the data, and (iii) classifier — how to *classify* an image. Acquiring "nice" image features and carefully modeling the feature data are vital steps in this approach.

As noted before, image classification can lead to a semantic organization of a digital database. Though this type of organization can be done in several ways, a hierarchical approach has multifold advantages: (i) easy browsing and navigating through the database, (ii) efficient retrieval, and (iii) ergonomically friendly presentation of the database. For instance, WebSEEK, a web image search engine [CSBB97], uses hierarchical semantic structure for collecting and searching images from the web. The image categories and hierarchies are preset by human design. An image is classified into one of the classes by first extracting key words from its html tag and then mapping the key words to classes. This approach requires human assistance and depends on the information in the html tag, which may be insufficient or even inaccurate and misleading.

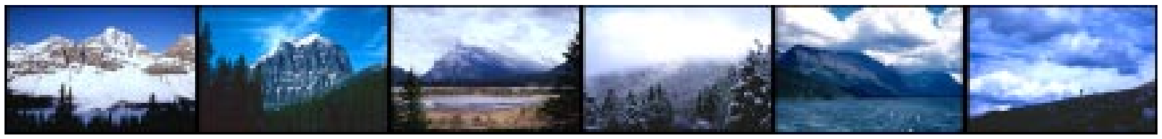
In this chapter, we propose a new scheme for automatic hierarchical image classification. We assume a training set of images with known class labels is available. Using banded color correlograms for the training images, we model the feature data using *singular value decomposition* [GL89] and construct a *classification tree*. Once the classification tree is obtained, any new image can be classified easily.

We test our method on 11 fairly representative classes of Corel images. We test our scheme using banded color correlograms and color histograms as features and compare our method to the nearest-neighbors directly applied to both color features. Our results suggest that this hierarchical

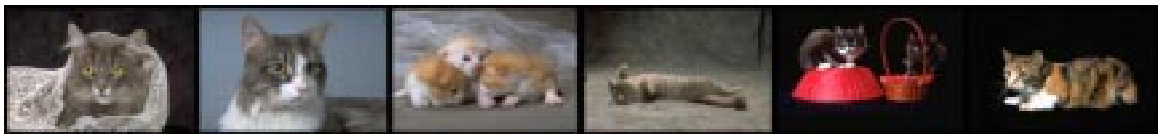
Aviation photography.



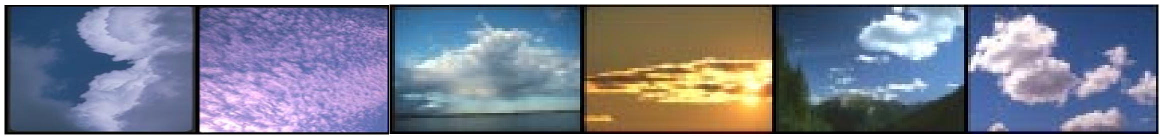
British motor collection.



Canadian Rockies.



Cats and kittens.



Clouds.

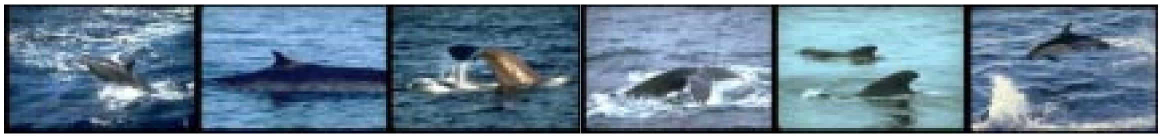
Figure 7.1: Sample images from image classes of aviation photography, British motor collection, Canadian Rockies, cats and kittens and clouds.

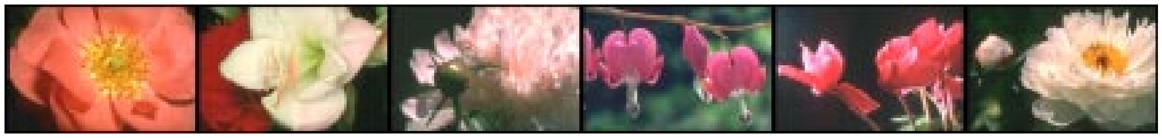
scheme is able to perform consistently better than the already effective nearest-neighbor algorithm (see [CBGM98]). The classification tree we obtain also conforms with the semantic content of the 11 classes. Our results also suggest that correlograms have more *latent semantic* structures (than histograms) that can be extracted by SVD procedure.

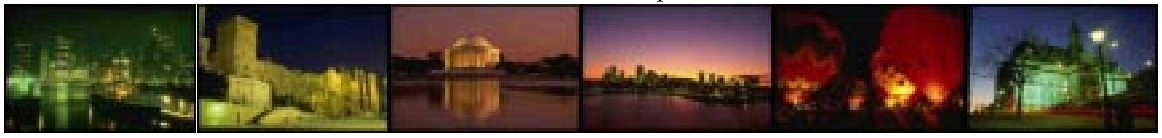
**Organization.** The rest of the chapter is organized as follows: Section 7.1 contains work related to image classification; Section 7.2 outlines how to use singular value decomposition to model feature vectors; and Section 7.3 describes our hierarchical classification method. Section 7.4 contains our experimental results and Section 7.5 concludes our discussions.

## 7.1 Related Work

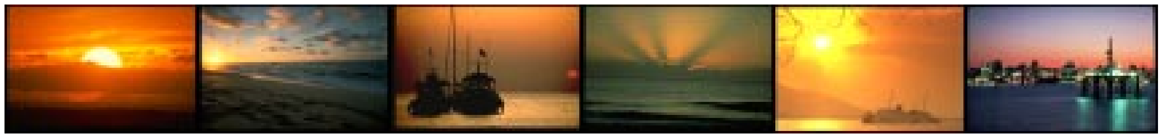
Since classification itself is a long-studied research area, different classifiers can be tried on image classification (e.g., $k$-nearest neighbor, decision trees, Bayesian nets, maximum likelihood analysis, linear discriminant analysis, neural networks, etc.). Not much work has been done on how to


Dolphins and whales.


Flowers close up.


Night scenes.

Spectacular waterfalls.


Sunsets around the world.

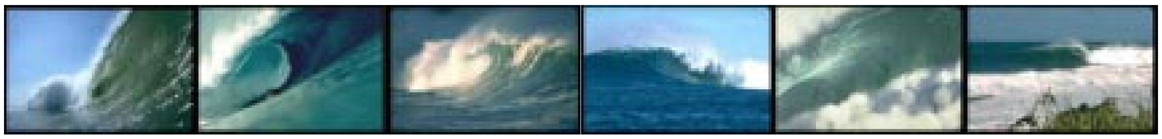

Waves.

Figure 7.2: Sample images from image classes of dolphins and whales, flowers close up, night scenes, spectacular waterfalls, sunsets around the world and waves.

represent images (i.e., features) and how to organize features. In the following, we review some previous work in image classification.

Yu and Wolf [YW95] present a one-dimensional *Hidden Markov Model* (HMM) for indoor/outdoor scene classification. An image is first divided into horizontal (or vertical) segments and each segment is further divided into blocks. Vector quantization technique is applied to model the color histograms of blocks. These color features are used to train HMM's for a preset standard set of clusters, such as a cluster of sky, tree, river, a cluster of sky, tree, grass. The maximum likelihood classifier is then used to classify an image as indoor or outdoor. The overall results of classification depend on the standard set of clusters which describe the indoor scene and outdoor scene. Generally, it is difficult to enumerate a set to cover a general case such as indoor/outdoor.

The *configural recognition* scheme proposed by Lipson *et. al.* [LGS97] is also a knowledge-based scene classification method. A model template, which encodes the common global scene configuration structure using qualitative measurements, is hand-crafted for each category. An image is then classified to the category whose model template that best matches the image by deformable template matching (which requires heavy computation, despite that the images are subsampled to low resolutions) — the nearest neighbor classification. The average percentage of correct classification on $4$ classes of scenery (snowy mountains, snowy mountains with lakes, fields, and waterfalls) is about $64$. To avoid the drawbacks of manual templates, a learning scheme that automatically construct scene templates from a few examples is proposed by [RG97]. The learning scheme was tested on two scene classes and suggested promising results.

Carson *et. al.* [CBGM98] propose a new representation for images. Each image is thought to consist of several *blobs*; each blob is coherent in color and texture space[1]. All the blobs in the training data of $14$ image categories are clustered into about $180$ "canonical" blobs using Gaussian models with diagonal covariance. Each image is then assigned a score vector which measures the nearest distance of each canonical blob to the image. These score vectors are used to train a decision-tree classifier. The results of this method are compared to color histograms with the decision-tree classifier.

Interestingly, the color histograms seem to perform better than blobs. Several explanations were given for this performance degradation: (i) the clustering procedure did a bad job on choosing canonical blobs and the canonical blobs were not interesting enough to distinguish categories; (ii) the trained decision-tree focused too much on irrelevant blobs which did not help in classification; and (iii) the $14$ categories have large amount of overlapping which cause the difficulties, such as fields appear in the categories "fields", "horses", and "elephants".

## 7.2 Singular Value Decomposition

In this section, we briefly review the singular value decomposition that we use for organizing image feature vectors.

Without loss of generality, let $m \geq n$. For an $m \times n$ matrix $A$, the *singular value decomposition* of $A$ is given by $A = U\Sigma V^T$ [GL89], where

(i) $U$ is an $m \times n$ matrix, $\Sigma, V$ are $n \times n$ matrices;

(ii) $U, V$ are column orthonormal i.e., $U^TU = V^TV = I_n$;

[1]This is one kind of color and texture based image segmentation method.

(iii) $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ where $r = \mathrm{rank}(A)$ and the *singular values* are $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_r > 0$.

The first $r$ columns of $U$ and $V$ together with the non-zero singular values actually are the eigenvectors and the $r$ non-zero eigenvalues of $AA^T$ and $A^TA$ respectively. Several efficient algorithms exist to compute the SVD of a matrix, especially if the matrix is known to be sparse.

The SVD of a matrix can be used to obtain lower rank approximations of the matrix. If we take the first $k$ columns of $U$, $V$ (denoted $U_{[k]}, V_{[k]}$) and the leading $k \times k$ submatrix of $\Sigma$ (denoted $\Sigma_{[k]}$), define

$$A_k = U_{[k]}\Sigma_{[k]}V_{[k]}^T = \sum_{i=1}^{k} U_i \Sigma_{i,i} V_i^T.$$

Then, $A_k$ is the best rank $k$ approximation to $A$, i.e.,

$$\min_{\mathrm{rank}(B)=k} |A - B|_2 = |A - A_k|_2 = \sigma_{k+1}.$$

This property of the SVD helps to obtain a good trade-off between the quality of approximation and the size of the approximation (i.e., $k$).

The advantages of SVD are nicely exploited in *latent semantic indexing* (LSI) for document retrieval [DDF$^+$90]. The SVD[2], in some sense, derives the underlying structure that is hidden in $A$. The approximation $A_k$ can be thought of dampening the “noise” and unreliability that is present in the original matrix $A$. When SVD is applied to feature vectors, it not only eliminate the “noise” in the feature vectors, but also reduce the dimension of the feature when $k < m$.

We outline our approach of using SVD with correlograms. Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ denote the set of training images and let $m$ be the number of color quantizations. We define the matrix $A_{i,j}(\mathcal{I}) \triangleq \beta_{c_i}(I_j)$. We compute the SVD of $A(\mathcal{I})$ to be $A(\mathcal{I}) = UDV^T$. Let $A_k = U_{[k]}\Sigma_{[k]}V_{[k]}^T$ be an approximation to $A$. We can choose $U_{[k]}$ as the basis for the new $k$-dimensional feature space. Then $V_{[k]}$ is the new representation for the correlograms in this reduced feature space. When we have a new image that needs to be classified, we first compute its correlogram $q$, then project $q$ onto the reduced feature space by computing

$$q' = q \cdot U_{[k]} \cdot \Sigma_{[k]}^{-1}$$

Now, the question is: how to choose $k$ for the approximation? We use the following heuristic to pick the $k$. Note that we want to find the best approximation $A_k$ such that the SVD representation of correlograms give the best classification results using nearest-neighbor rule. Instead we compute the classification for each $k$ between the number of classes (i.e., $c$) to an upper limit $k^*$ and choose the best $k$ in this range. Now, we show how to choose $k^*$. Notice that the singular values of $A$ correspond to the eigenvalues of $AA^T$, which is the correlation matrix of local color density for the training images. We set $k^*$ be the $k^*$-th biggest eigenvalue that is within $2\%$ of the maximum eigenvalue, i.e., we ignore those correlations whose values are less than $2\%$ of the maximum correlation[3].

Note that in the above SVD method, histograms can be used instead of correlograms. We will see (Section 7.4) that the performance with correlograms is much better than with histograms.

[2]For the difference among SVD, KLT and PCA, see [Gerb81].

[3]There is no good heuristic for choosing $k$. The rule of thumb is finding the $k$ that gives the best performance [BB62].

## 7.3 The Hierarchical Classification Scheme

Let $\mathcal{C} = \{C_1, \ldots, C_c\}$ be the image classes known *a priori*. We assume that we have a set $\mathcal{S}$ of training images whose class membership is known and $\mathcal{T}$ of images that need to be classified. We want build a classification tree from training images. At each level of the classification tree, we aim to choose the best modeling of the training data. We first use SVD to eliminate "noise" from the training data as described in Section 7.2. We then classify each image in the training data using the nearest-neighbor algorithm with the first neighbor dropped (similar to the leave-one-out cross-validation scheme). Based on the performance of this classification, we then split the classes into two subclasses such that the intra-subclass association is maximized while simultaneously the inter-subclass disassociation is minimized. This is accomplished using *normalized cuts* [SM97]. Finally, the subclasses and those training images that were correctly classified with respect to the subclasses are worked upon recursively to obtain the hierarchy in the classification tree, with the hope of improving the classification performance.

### 7.3.1 Confusion Matrix

We construct the matrix $A(\mathcal{S})$ as indicated in Section 7.2 and compute its SVD: $A(\mathcal{S}) = U\Sigma V^T$. Then we choose the approximation $A_k$ that gives the best classification of $\mathcal{S}$ on itself. The details are the following.

For an image $I \in \mathcal{S}$, and $C(I)$ be the class of $I$, let $\beta_k(I)$ denote the $k$-dimensional reduced SVD representation of $I$. We consider each $I \in \mathcal{S}$ as a query and obtain the class $C'(I)$ where

$$C'(I) \triangleq C(\arg \min_{J \in \mathcal{S} \setminus \{I\}} \{|\beta_k(I) - \beta_k(J)|\}).$$

In other words, $C'(I)$ is the class assigned by the nearest neighbor classification when all images other than $I$ itself is considered. Intuitively, this procedure helps to find the best association patterns between classes from SVD.

Now, the $c \times c$ *confusion matrix* $M$ is then defined by

$$M_{i,j} = \text{size of } \{I \mid C(I) = C_i, C'(I) = C_j\}$$

The diagonal entries of $M$ are the number of images that are correctly classified, while the off-diagonal entries are the misclassifications. The average percentage of correct classification is just the sum of the diagonal entries ($\mathrm{trace}\,(M)$) divided by the size of $\mathcal{S}$.

### 7.3.2 Normalized Cuts

In this section, we show how to partition the confusion matrix $M$ on basis of maximizing the inter-class association. First, we review some basic definitions from graph theory.

Given a weighted graph $G = \langle V, E \rangle$ with $w(u, v)$ being the weight of an edge $(u, v)$, the *mincut* is defined to be a partition of $V = V_1 \cup V_2$ such that

$$\mathrm{cut}_w(V_1, V_2) \triangleq \sum_{(u,v) \in V_1 \times V_2} w(u, v)$$

is minimized. Mincuts can be computed in polynomial time using network flow techniques.

The confusion matrix $M$ defines a natural directed graph. The mincut in this graph corresponds to a partition of the classes into $M_1$ and $M_2$ such that the number of misclassification among these classes is minimized. A partition of $M$ according to the mincut, however sometime favors cutting small sets [WL93], i.e., one of $V_1, V_2$ is very small. This problem is considered in [SM97] where *normalized cuts* are introduced.

Formally, the normalized cut is given by the best partition of $V = V_1 \cup V_2$ that minimizes

$$\mathrm{ncut}_w(V_1, V_2) \triangleq \mathrm{cut}_w(V_1, V_2)\left(\frac{1}{\mathrm{cut}_w(V_1, V)} + \frac{1}{\mathrm{cut}_w(V_2, V)}\right).$$

The partition based on normalized cut is shown to have the property that is minimizes the disassociation between the groups and maximizes the association within the group.

Define the diagonal matrix $M'_{i,i} = \sum_j M_{i,j}$. Normalized cuts can be computed reasonably well and efficiently by computing the second smallest eigenvalue of the system defined by $(M - M')x = \lambda M' x$ and using some additional heuristics. The details can be looked up in [SM97].

We use normalized cuts to partition $M$ into $M_1$ and $M_2$, in effect obtaining a partition of the classes $\mathcal{C}$ into $\mathcal{C}_1$ and $\mathcal{C}_2$.

### 7.3.3 Classification Tree

Using the normalized cuts, we can build the classification tree recursively. Given the original set of classes $\mathcal{C}$, we compute the partition $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ based on normalized cuts. We define $\mathcal{S}_1 = \{I \in \mathcal{S} \mid C'(I) \in \mathcal{C}_1 \text{and } C(I) \in \mathcal{C}_1\}$ and $\mathcal{S}_2 = \{I \in \mathcal{S} \mid C'(I) \in \mathcal{C}_2 \text{and } C(I) \in \mathcal{C}_2\}$. In this way, the images that are misclassified across $\mathcal{S}_1$ and $\mathcal{S}_2$ are not considered from now on. We then recursively work on classifying $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) with $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) as the set of classes.

The algorithm is described in its entirety below:

```
BUILD-TREE($\mathcal{S}$,$\mathcal{C}$)
  let $\mathcal{S} = \{I_1, \ldots, I_n\}$
  let $\mathcal{C} = \{C_1, \ldots, C_c\}$
  $A_i(\mathcal{S}) = \beta(I_i), 1 \le i \le n$
  compute SVD of $A(\mathcal{S}) = U\Sigma V^T$
  let $k^* = |\{\sigma_i \mid \sigma_i \ge 0.2\sigma_1\}|$
  for $k = c$ to $k^*$ do
    use reduced SVD representation to compute
      $C_k(I) = C(\arg\min_{J\in\mathcal{S}\setminus\{I\}}\{|\beta_k(I) - \beta_k(J)|\})$
    form the confusion matrix $M^{(k)}$ as
      $M^{(k)}_{i,j} = |\{I \mid C(I) = C_i, C_k(I) = C_j\}|$, $1 \le i,j \le c$
    compute the average correct classification $\gamma_k$ as
      $\gamma_k = \sum_{i=1}^{c} M^{(k)}_{i,i}/|\mathcal{S}|$
    choose $M^* = M^{(k)}$ such that $\gamma_k$ is maximal
  compute $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ such that $\mathrm{ncut}_{M^*}(\mathcal{C}_1, \mathcal{C}_2)$ is minimized
  let $\mathcal{S}_1 = \{I \in \mathcal{S} : C'(I) \in \mathcal{C}_1\}$
  let $\mathcal{S}_2 = \{I \in \mathcal{S} : C'(I) \in \mathcal{C}_2\}$
  BUILD-TREE($\mathcal{S}_1$,$\mathcal{C}_1$)
  BUILD-TREE($\mathcal{S}_2$,$\mathcal{C}_2$)
```

**Trimming.** Sometimes, the performance of the classification on the training data does not always improve level by level using reduced SVD representations. This is because some variations that are important to a set of classes may be removed by the SVD reduction. In this case, it does not pay-off to recursively split such classes. We perform a trimming procedure on the tree we obtain from the algorithm to remove subtrees that do not improve the classification correctness.

## 7.4 Experiments and Results

### 7.4.1 Experiments

We use $11$ image classes from Corel collections: aviation photography (A1), British motor car collection (B1), Canadian Rockies (C1), cats and kittens (C2), clouds (C3), dolphins and whales (D1), flowers (F1), night scenes (N1), spectacular waterfalls (S1), sunsets around the world (S2), and waves (W1). For simplicity, we use the abbreviations. See Figure 7.1 and Figure 7.2 for some sample images from these classes. These images contain a wide range of content (scenery, animals, objects, etc.), colors, and lighting conditions. We delete some images in each class which are not consistent with the rest of the class (as in [CBGM98]) and leave $90$ images in each class. Since we use the nearest-neighbor rule with the classification tree, we want to make sure that the color distributions of training images and test images are more or less the same. Therefore, we randomly shuffle the images in each class and take $70$ images as the training set and the rest $20$ images as the test set. By doing so three times (to ensure fairness), we obtain $3$ sets of training data and test data.

To compute color histograms and color correlograms, we quantize the RGB color space[4] into $8 \times 8 \times 8 = 512$ colors (3 bits for each color channel). This level of quantization is good enough for the SVD to extract the underlying structure, while not being too big (unlike $6912$ colors used in [CBGM98]) so as to affect efficiency.

### 7.4.2 Results

We test both color correlograms and color histograms on the hierarchical classification approach and compare the hierarchical approach with the nearest-neighbor classification. The three classification trees from three training sets are more or less the same and are consistent with the color content of the $11$ classes. We only present the tree from the first data set (Figure 7.3). From the classification tree, we see that A1 and C3 share the same parent because of the same sky background; similarly, W1, S1, and D1 are grouped together because of the same water background.

The confusion matrices for different methods are shown in Table 7.1, Table 7.2, and Table 7.3. The classification behavior of the classification tree is quite different from the nearest-neighbor. The classification tree is better than the nearest-neighbor in that

1 the overall number of misclassification between classes is smaller;

2 the overall number of correct classification is larger.

The average percentage of correctness of the three test sets are summarized in the table 7.4. The results show that the hierarchical method (with trimming) is consistently better than the

[4]The results don't change much if HSV color space is used.

nearest-neighbor classification, and the color correlogram is consistently better than the color histogram.

Using the simple nearest-neighbor classification, the correlogram performs $3\%$ better than the histogram; using the classification tree, the correlogram performs $21\%$ better than the histogram. Using the classification tree, the correlogram improves $3\%$ over the nearest-neighbor; it improves $7\%$ over the nearest-neighbor on the histogram. Note that the average data size of the SVD representations is about $15,3\%$ of the original size. The average number of non-leaf nodes in the classification trees is five after trimming. Therefore the computation and storage of the data for the classification saves about $85\%$, a significant number.

**Remark.** We notice from the results that the color histogram performs consistently worse with the classification tree than with the nearest-neighbor, while the color correlogram performs consistently better with the classification tree than with the nearest-neighbor. This suggests that correlograms have an underlying "latent semantic" structure (local color density). Color histograms do not seem to have such a property.

Table 7.1: Class confusion matrix for trimmed classification tree (correlogram).

| | A1 | C3 | C1 | D1 | W1 | B1 | C2 | S1 | F1 | N1 | S2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 | **17** | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C3 | 2 | **15** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 |
| C1 | 0 | 2 | **16** | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D1 | 0 | 1 | 0 | **17** | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| W1 | 0 | 1 | 1 | 4 | **13** | 0 | 0 | 1 | 0 | 0 | 0 |
| B1 | 0 | 0 | 0 | 0 | 0 | **20** | 0 | 0 | 0 | 0 | 0 |
| C2 | 0 | 0 | 0 | 0 | 0 | 0 | **20** | 0 | 0 | 0 | 0 |
| S1 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | **17** | 0 | 0 | 0 |
| F1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | **17** | 0 | 0 |
| N1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **18** | 1 |
| S2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | **17** |

## 7.5 Summary

We propose using the singular value decomposition with banded color correlogram to extract a "latent semantic" structure of images for classification into categories. Our tests on 11 image classes show that our method using this scheme and a classification tree not only performs better than the nearest-neighbor classification but also saves much computation and data storage (about $85\%$). In addition, the results also suggest that the correlogram is more suitable for the image classification task that the color histogram.

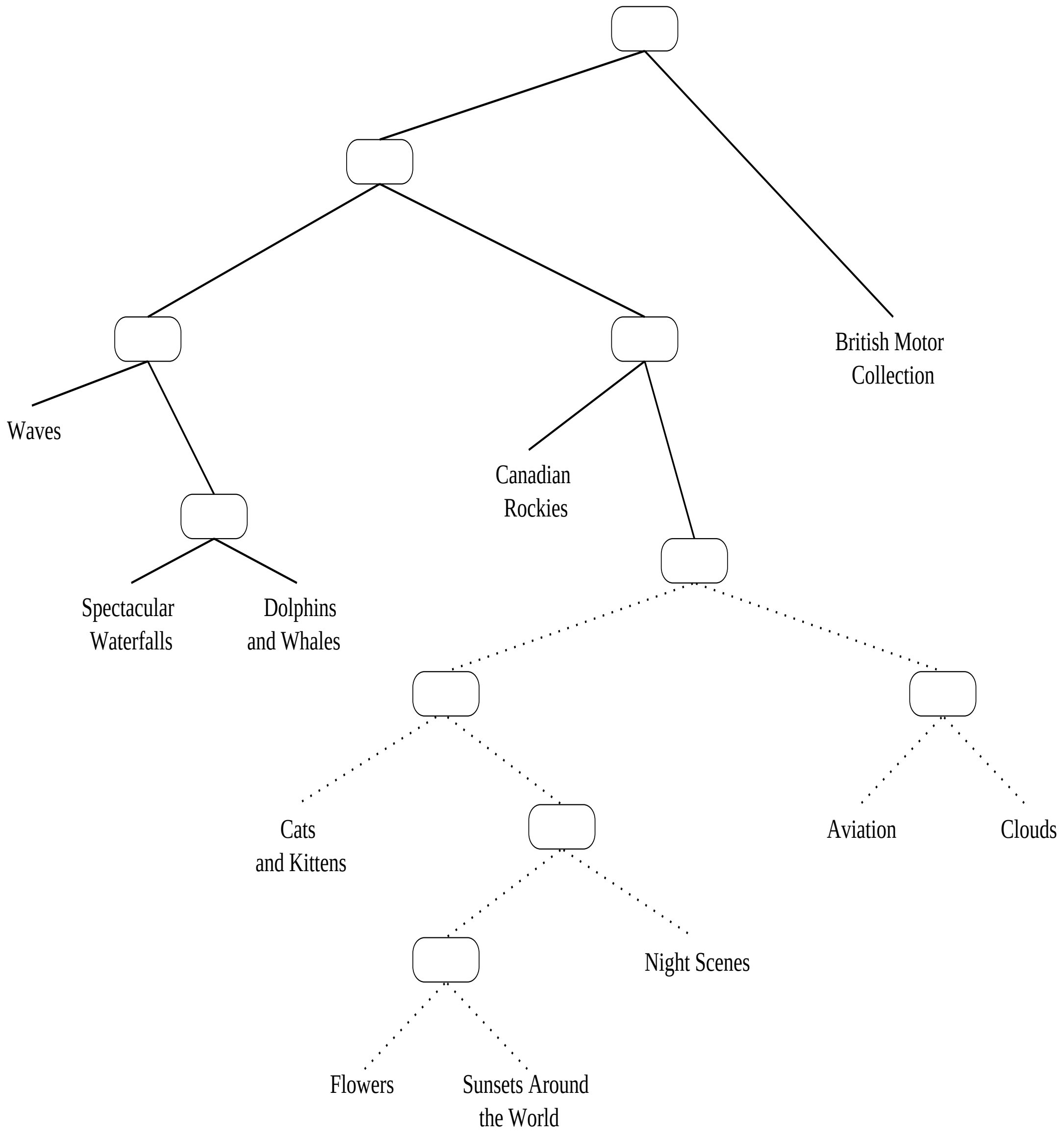


Figure 7.3: The classification tree obtained from the first training set using correlograms. The dotted lines indicate the trimmed portions.

Table 7.2: Class confusion matrix for the nearest-neighbor classification (correlogram).

| | A1 | C3 | C1 | D1 | W1 | B1 | C2 | S1 | F1 | N1 | S2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 | **16** | 1 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| C3 | 1 | **14** | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 1 |
| C1 | 0 | 0 | **15** | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D1 | 0 | 1 | 0 | **19** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W1 | 0 | 0 | 0 | 2 | **17** | 0 | 0 | 1 | 0 | 0 | 0 |
| B1 | 0 | 0 | 0 | 0 | 0 | **14** | 4 | 2 | 0 | 0 | 0 |
| C2 | 0 | 0 | 0 | 0 | 0 | 0 | **20** | 0 | 0 | 0 | 0 |
| S1 | 0 | 1 | 0 | 2 | 1 | 0 | 0 | **16** | 0 | 0 | 0 |
| F1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | **14** | 1 | 2 |
| N1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **19** | 1 |
| S2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | **16** |

Table 7.3: Class confusion matrix for the nearest-neighbor classification (histogram).

| | A1 | C3 | C1 | D1 | W1 | B1 | C2 | S1 | F1 | N1 | S2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 | **14** | 0 | 2 | 0 | 1 | 0 | 2 | 1 | 0 | 0 | 0 |
| C3 | 0 | **13** | 1 | 2 | 0 | 0 | 3 | 1 | 0 | 0 | 1 |
| C1 | 0 | 1 | **16** | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
| D1 | 0 | 0 | 0 | **17** | 2 | 0 | 1 | 0 | 0 | 0 | 0 |
| W1 | 0 | 0 | 0 | 3 | **16** | 0 | 0 | 1 | 0 | 0 | 0 |
| B1 | 0 | 0 | 0 | 0 | 0 | **17** | 1 | 1 | 0 | 1 | 0 |
| C2 | 0 | 0 | 0 | 0 | 0 | 0 | **20** | 0 | 0 | 0 | 0 |
| S1 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | **16** | 0 | 0 | 0 |
| F1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | **18** | 0 | 0 |
| N1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **13** | 6 |
| S2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 2 | **13** |

Table 7.4: Correctness classification on three data sets.

| | Nearest-Neighbor | | | Classification Tree (Trim) | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 |
| Hist | 0.786 | 0.746 | 0.786 | 0.696 | 0.677 | 0.668 |
| Corr | 0.818 | 0.800 | 0.786 | **0.850** | **0.805** | **0.823** |

# Chapter 8

# Conclusions and Future Work

The main contribution of this thesis is the use of color correlogram — a new image feature — for solving several problems that arise in content-based image retrieval and video browsing. The novelty in this feature is the characterization of images in terms of the spatial correlation of colors instead of merely the colors *per se*. Briefly, a color correlogram expresses how the spatial correlation of color changes with distance. This information seem to both discriminate "different" images and identify "similar" images very well. We show that correlograms can be computed, processed, and stored at almost no extra cost, thereby justifying using this instead of many other features to get more bang for the buck.

The most important application of correlograms is in content-based image retrieval systems. Viewed in this context, correlogram is neither a region-based nor a histogram-based method. Unlike purely local properties, such as pixel position, gradient direction, or purely global properties, such as color distribution, correlogram takes into account the local color spatial correlation as well as the global distribution of this spatial correlation. While any scheme that is based on purely local properties is likely to be sensitive to large appearance changes, correlogram is more stable to tolerate these changes and while any scheme that is based on purely global properties is susceptible to false positive matches, correlogram seems to be scalable for CBIR. This is corroborated by our extensive experiments on large image collections, where we demonstrate that correlograms are very promising for CBIR.

One issue that still needs to be resolved satisfactorily is the following: in general, illumination changes are very hard to handle in color-based CBIR systems [Gong98, SC96, FF95, SH95]. During our experiments, we encountered this problem occasionally. Though correlograms perform better on a relative scale, its absolute performance is not fully satisfactory. The question is, can correlograms, with some additional embellishments, be made to address this specific problem?

On a related note, it also remains to investigated if correlograms, in conjunction with other features, can enhance retrieval performance. For instance, how will the correlogram perform if shape information is used additionally? This brings up the question of object-level retrieval using correlograms. More work needs to be done in this regard as to finding a better representation for objects.

In the process of using correlograms for CBIR, we use a new metric — the relative distance metric. On average, this metric performs much better than the traditional $L_1$ metric for histogram-based features. In rare cases, however, it fails miserably. This behavior needs to be pinned down exactly. Taking a step back further, one can ask if a measure needs to be a metric at all [JWG98]? Is there a non-metrical measure that can boost the performance of the correlograms?

Further applications of color correlograms are image subregion querying and localization, which are indispensable feature of any image management system. Our notions of correlogram intersection and correlogram correction seems to perform well in practice. But, certainly there is room for improvement and these needed to investigated in greater detail. We also apply correlograms to the problem of detecting cuts in video sequences. An interesting question that arises here is, can this operation be done in the compressed domain [YL95]? This would cut down the computation time drastically and make real-time processing feasible.

We also suggest two supervised learning methods to incorporate feedback information to enhance the quality of image retrieval using correlograms. Our learning methods are in fact independent of feature vectors and could conceivably be used for other feature vectors as well. Initial experiments indicate that the proposed learning methods significantly improve retrieval quality using little effort on the part of the user. The application of these techniques to object searching needs to be studied in greater detail. In particular, it is not obvious if one of the methods actually outperforms the other, though they seem to have a synergic effect when combined. A user-study of the performance of these methods needs to be performed in the future.

In an effort to exploit correlograms in supervised image classification, we propose using the singular value decomposition with banded color correlogram to extract a "latent semantic" structure of images for classification into categories. Our tests show that our method using this scheme and a classification tree not only performs better than the nearest-neighbor classification but also saves much computation and data storage. It will be interesting to use techniques like *feature weighting* [WAM97] to further assist SVD to discover latent semantic structures from training data.

In general, the algorithms we propose for various problems are not only very simple and inexpensive but also tie-in well if the underlying indexing scheme is correlogram-based. It pays off more in general if there were a uniform feature vector that is universally applicable to providing various functionalities expected of a CBIR system (like histograms advocated in [SB91]). It is unreasonable to expect any CBIR system to be absolutely fool-proof; it is needless to state that correlogram is not the panacea. The goal, however, is to build relatively better CBIR systems and in this direction, based on various experiments, we feel that there is a compelling reason to use correlograms as one of basic building blocks in such systems.

# Appendix A

# Probabilistic Formulation of Correlograms

Let $\mathcal{I}$ be an $n \times n$ image. (For simplicity of exposition, we assume that the image is square.) The colors in $\mathcal{I}$ are quantized into $m$ colors $c_1, \ldots, c_m$.

Randomly pick a pixel $p = (x, y)$ from $\mathcal{I}$. Let a random variable $c$ represent the color value of $p$, i.e., the sample space for $c$ is $\{c_1, c_2, ..., c_m\}$. The probability distribution of $c$ is the following:

$$\begin{aligned}\Pr(c = c_i) &= \frac{\text{number of pixels of color } c_i}{\text{total number of pixels in image } \mathcal{I}} \\ &= \frac{H_{c_i}(\mathcal{I})}{n^2}\end{aligned}$$

which is just the *histogram* we defined in Section 2.1.1.

Now consider another event. Randomly pick a pixel $p_1 = (x_1, y_1)$ from the image $\mathcal{I}$, and randomly pick another pixel $p_2 = (x_2, y_2)$. Let random variables $c_1, c_2$ represent the color values of $p_1$ and $p_2$ respectively. If we ignore the boundary conditions, then

$$\Pr(c_1 = c_i, c_2 = c_j, |p_1 - p_2| = k) = \frac{\Gamma^{(k)}_{c_i,c_j}(\mathcal{I})}{n^2 \times 8k}$$

The numerator is the number of pairs of pixels which are $k$ distance away in the image $\mathcal{I}$ and the colors are $c_i, c_j$, the denominator is the total number of pairs of pixels which are $k$ distance away (here we ignore the boundary conditions). The factor $8k$ is due to the eight neighbors which are $k$ distance (in $L_\infty$-norm) away from the center pixel.

If we fix the color of pixel $p_1$ to be $c_i$, by the definition of *conditional probability* ([Fell66]),

$$\begin{aligned}\Pr(c_2 = c_j, |p_1 - p_2| = k \mid c_1 = c_i) &= \frac{\Pr(c_1 = c_i, c_2 = c_j, |p_1 - p_1| = k)}{\Pr(c_1 = c_i)} \\ &= \frac{n^2}{H_{c_i}} \times \frac{\Gamma^{(k)}_{c_i,c_j}(\mathcal{I})}{n^2 \times 8k} \\ &= \frac{\Gamma^{(k)}_{c_i,c_j}(\mathcal{I})}{H_{c_i} \times 8k}\end{aligned}$$

Therefore, the correlogram describes how the conditional probability distribution of pairs of colors changing with distance $k$.

# Bibliography

[ABF$^+$95] J. Ashley, R. Barber, M. D. Flickner, J. L. Hafner, D. Lee, W. Niblack and D. Petkovic. Automatic and semiautomatic methods for image annotation and retrieval in QBIC. *Proc. SPIE, Storage and Retrieval for Image and Video Databases* III, pp. 24–35, 1995.

[BFG$^+$96] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain and C. Shu. Virage image search engine: an open framework for image management. *Storage & Retrieval for Image and Video Databases IV, Proc. SPIE 2670*, 1996.

[BN96] Simon Baker and Shree Nayar. Pattern rejection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 544–549, 1996.

[BCGM98] Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Color- and texture-based image Segmentation using EM and its application to content-based image retrieval. *Proc. 8th Intl. Conference on Computer Vision*, 1998.

[BR96] J. S. Boresczky and L. A. Rowe. A comparison of video shot boundary detection techniques. *Storage & Retrieval for Image and Video Databases IV, Proc. SPIE 2670*, pp. 170–179, 1996.

[BB62] H. Borko and M. Bernick. Automatic document classification. *Journal of the ACM 9*, pp. 512–521, 1962.

[BE92] S. A. Brock-Gunn and T. J. Ellis. Using color templates for target identification and tracking. *Proc. British Machine Vision Conference*, pp. 207–216, 1992.

[CBGM97] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. *Porc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp42–51, 1997.

[CBGM98] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using EM and its application to image querying and classification. Manuscript, submitted to *Pattern Analysis and Machine Intelligence*.

[CCWS$^+$97] S.-F. Chang, W. Chen, H. Weng, H. Sundaram and D. Zhong. VideoQ: an automatic content-based video search system using visual cues. *Proc. ACM Multimedia*, pp.313–324, 1997.

[CSBB97] S-F. Chang, J.R. Smith, M. Beigi and A. Benitez. Visual information retreival from large distributed online repositories. *Communications of the ACM 40*, No.12, December 1997.

[Comp95] Content-based image retrieval systems. *IEEE Computer*, September 1995.

[CMOY96] I. J. Cox, M.L. Miller, S.M. Omohundro and P. N. Yianilos. PicHunter: Bayesian relevance feedback for image retrieval. *Intl. Conf. on Pattern Recognition*, Vienna, Austria, 1996.

[CGY96] I.J. Cox, J. Ghosn and P.N. Yianilos. Feature-Based Face Recognition Using Mixture-Distance. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 209-216, 1996.

[CPG$^+$97] I.J. Cox, T.V. Papathomas, J. Ghosh, P.N. Yianilos and M.L. Miller. Hidden annotation for content-based image retrieval. *Porc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp76–81, 1997.

[DRD97] M. Das, E. M. Riseman and B. A. Draper. FOCUS: searching for multi-colored objects in a diverse image database. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 756-761, 1997.

[DDF$^+$90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*, pp. 391–407, 1990.

[Dela96] R. L.Delanoy. Supervised learning of tools for content-based search of image databases. *SPIE proceedings*, 2670:194–205, 1996.

[DH73] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. John Wiley & Sons Inc., 1973.

[EM95] F. Ennesser and G. Medioni. Finding waldo, or focus of attention using local color information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8), 1995.

[Fell66] W. Feller. *An introduction to probability theory and its applications.* New York, Wiley, 1966.

[FFB96] M. M. Fleck, D. A. Forsyth, and C. Bregler. Finding naked people. *European Conf. on Computer Vision*, volume II, pp. 590–602, 1996.

[FSN$^+$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. *IEEE Computer*, 28(9):23–32, September 1995.

[FMM$^+$96] 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. *Proc. Intl. Workshop on Object Recognition*, Cambridge, April 1996

[Fuku90] K. Fukunaga. *Introduction to statistical pattern recognition.* Second Edition, Acedemic Press Inc, 1990.

[FF95] B. Funt and G. Finlayson. Color constant color indexing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.17, pp. 522–529, 1995.

[Furh96] *Multimedia tools and applications*. Kluwer Academic Publishers, edited by Borko Furht, 1996.

[Gerb81] J. J. Gerbrands. On the relationships between SVD, KLT and PCA. *Pattern Recognition*, 14(1), pp375–381.

[GL89] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1989.

[GZCS94] Y. Gong, H. Zhang, H.C. Chuan and M. Sakauchi. An image database system with content capturing and fast image indexing abilities. *Intl. Conf. on Multimedia Comp & Systems*, pp. 121–130, 1994.

[Gong98] Yihong Gong. *Intelligent image databases: towards advanced image retrieval*. Kluwer Academic Publishers, 1998.

[GPF98] Y. Gong, G. Proietti and C. Faloutsos. Image indexing and retrieval based on human perceptual color clustering. *Proc. 17th IEEE Conf. on Computer Vision and Pattern Recognition*, to appear, 1998.

[GL 1987] W. L. Grimson and T. Lozano-Pérez. Localising overlapping parts by searching the interpretation tree. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9:469–482, 1987.

[Gros97] W. Grosky. Managing multimedia information in database systems. *Communications of the ACM 40*, No.12, december 1997.

[GJM97] W. Grosky, R. Jain and R. Mehrotra. *The handbook of multimedia information management*. Prentice-Hall, Inc. 1997.

[GTB97] B. Günsel, A. M. Tekalp and P.J.L. van Beek. Object-based Video Indexing for Virtual Studio Productions. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 769–774, 1997.

[HSE95] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7):729–736, July 1995.

[HJW94] A. Hampapur, R. Jain, and T. Weymouth. Digital video indexing in multimedia systems. *Proc. AAAI-94 Workshop on Indexing and Reuse in Multimedia Systems*, August, 1994.

[Hara79] R. M. Haralick. Statistical and structural approaches to texture. *Proc. of IEEE*, 67(5):786–804, 1979.

[Haus92] D. Haussler. Decision theoretic generalization of the PAC model for neural net and other learning applications. *Information and Computation*, 100:78–150, 1992.

[HCP95] W. Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. *3rd ACM Multimedia Conf.*, pp. 305–313, November 1995.

[HKM$^+$97] J. Huang, S. R. Kumar, M. Mitra, W. J. Zhu, and R. Zabih. Image indexing using color correlograms. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 762–768, 1997.

[HKM97] J. Huang, S. R. Kumar, and M. Mitra. Combining supervised learning with color correlograms for content-based image retrieval. *Proceedings of the Fifth ACM Multimedia Conference*, pp.325–334, 1997.

[HKMZ98] J. Huang, S. R. Kumar, M. Mitra, and W. J. Zhu. Spatial color indexing and applications. *Proc. 8th Intl. Conference on Computer Vision*, 1998.

[HKZ98] J. Huang, S. R. Kumar, R. Zabih. Automatic and hierarchical color image classification. *Proceedings of the Sixth ACM Multimedia Conference*, to appear, 1998.

[HU86] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. *Proc. Intl. Conf. on Computer Vision*, pp. 102–111, 1986.

[HKR93] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:850–863, 1993.

[HLO96] D. P. Huttenlocher, R. H. Lilien, and C. F. Olson. Object recognition using subspace methods. *Proc. European Conf. on Computer Vision*, pp. 536–545, 1996.

[IAH95] M. Irani, P. Anandan and S. Hsu. Mosaic based representations of video sequences and their applications. Proc. ICCV, pp. 605–611, 1995.

[IWYS96] K. Ikeuchi, M.D. Wheeler, T. Yamazaki, and T. Shakunaga. Invariant Histograms and Deformable Template Matching for SAR Target Recognition. *Proc. 15th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 100-105, 1996.

[JWG98] D. W. Jacobs, D. Weinshall and Y. Gdalyahu. Condensing image databases when retrieval is based on non-metric distances *Proc. Intl. Conference on Computer Vision*, 1998. To appear.

[JZL96] A. Jain, Y. Zhong and S. Lakshamanan. Object Matching Using Deformable Templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 18, pp. 267-278, 1996

[JDL96] A. Jain, M.P. Dubuisson and S. Lakshamanan. Vehicle Segmentation Using Deformable Templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 18, pp. 293-308, 1996.

[JV96] A. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8), 1996.

[Jain97] R. Jain. Visual information management. *Communications of the ACM 40*, No.12, December 1997.

[KKS91] T. Kato, T. Kurita, and H. Shimogaki. Intelligent visual interaction with image databases. *Journal of Information Processing of Japan*, 12(2):134 – 143, 1991.

[KCH95] P. Kelly, M. Cannon and D. Hush. Query by image example: the comparison algorithm for navigating digital image databases (CANDID) approach. *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pp. 238–249, 1995.

[KD97] V. Kobla and D. Doermann. Video trails: representing and visualzing structure in video sequences. *Proc. ACM Multimedia*, pp.335–346, 1997

[Lee92] H. C. Lee. A physics-based color encoding model for images of natural scenes. Proceedings of the Conference on Modern Engineering and Technology, Electro-Optics session, pages 25–52, Taipei, Taiwan, 1992.

[LGS97] P. Lipson, E. Grimson, and P. Sinha. Configuration based scene classification and image indexing. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp.1007–1013, 1997.

[LP96] F. Liu and R. Picard. Periodicity, directionality and randomness: Wold features for image modeling and retrieval. *IEEE Trans. Patt. Anal. Mach. Intel. 18*, 7, pp722-733, 1996.

[MM97] W. Ma and B. Manjunath. Edge flow: a framework of boundary detection and image segmentation. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 744–749, 1997.

[Ma97] W. Ma. NETRA: a toolbox for navigating large image databases. Ph.D dissertation, Dept. of Electrical and Computer Engineering, Univ. of California at Santa Barbara, 1997.

[MM96] B. Manjunath and W. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on PAMI, Special Issue on Digital Libraries*, November 1996.

[MN78] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. R. Soc. Lond. B.*, 200:269–294, 1978.

[MMK95] J. Matas, R. Marik, and J. Kittler. On representation and matching of multi-colored objects. *Proc. IEEE 5th Intl. Conf. on Computer Vision*, pp. 726–732, 1995.

[MG 1993] R. Mehrotra and J. E. Gary. Feature-based retrieval of similar shapes. *Proc. 9th Data Engineering Conference*, pp. 108–115, Vienna, Austria, 1993.

[MP96] T. Minka and R. Picard. Interactive learning using a "society of models". *Proc. Computer Vision and Pattern Recognition*, 1996.

[NT92] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearences. In E. Knuth and L. Wegner, editors, *Visual Database Systems II*, pp. 127–173, SPIE, Elsevier, 1992.

[NB93] S. K. Nayar and R. M. Bolle. Reflectance ratio: a photometric invariant for object recogniton. *Proc. Fourth International Conference on Computer Vision*, pp. 280–285, 1993.

[VMH96] V. V. Vinod, H. Murase, and C. Hashizume. Focused color intersection with efficient searching for object detection and image retrieval. *IEEE Proc. Multimedia*, pp. 229–233, 1996.

[MN95] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *Intl. Journal of Computer Vision*, 14:5–24, 1995.

[CS95] V. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, September 1995.

[OPH96] T. Ojala, M. Pietikainen and D. Harwood. A comparative study of texture measures with classification based feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[ORC$^+$97] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra and T. Huang. Supporting similarity queries in MARS. *Proceedings of the Fifth ACM Multimedia Conference*, pp.403–412, 1997.

[PYL97] I. K. Park, I. D. Yun and S. U. Lee. Models and algorithms for efficient color image indexing. *Porc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp.36–49, 1997.

[PZM96] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. *4th ACM Multimedia Conf.*, November, 1996.

[PZ96] G. Pass and R. Zabih. Histogram refinement for content-based image retrieval. *IEEE Workshop on Applications of Computer Vision*, pp. 96–102, 1996.

[PZ98] G. Pass and R. Zabih. Comparing images using joint histograms. *ACM Multimedia Systems*, to appear, 1998.

[PPS96] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *Intl. Journal of Computer Vision*, 18(3):233–254, 1996.

[PM95] R. W. Picard and T. P. Minka. Vision texture for annotation. *Journal of Multimedia Systems*, Vol.3, pp3–14, 1995.

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical recipes in C*. Cambridge University Press, second edition, 1992.

[PHB97] J. Puzicha, T. Hofmann and J. M. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 267–272, 1997.

[RB95] R. P. Rao and D. Ballard. Object indexing using an iconic sparse distributed memory. *Proc. IEEE 5th Intl. Conf. on Computer Vision*, pp. 24–31, 1995.

[RL93] A. R. Rao and G. L. Lohse. Towards a texture naming system: identifying relevant dimensions of texture. *IEEE Conf. on Visualization*, pp220–227, San Jose, CA 1993.

[RG97] A. L. Ratan and W. E. L. Grimson. Training templates for scene classification using a few examples. *Proc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 90–97, 1997.

[RM97] S. Ravela and R. Manmatha. Retrieving images by similarity of visual appearance. *Porc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp67–74, 1997.

[RS96] R. Rickman and J. Stonham. Content-based image retrieval using color tuple histograms. *SPIE Proc.*, 2670:2–7, February 1996.

[MR90] A. Margalit and A. Rosenfeld. Using probabilistic domain knowledge to reduce the expected computational cost of template matching. *Computer Vision, Graphics, and Image Processing*, 51:219–234, 1990.

[Robe65] L. G. Roberts. Machine perception of three-dimensional solids. In *Optical and Electro-Optical Information Processing*. MIT Press, 1965.

[RL87] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection.* John Wiley & Sons, 1987.

[RGT97] Y. Rubner, L. Guibas and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. em Proc. *DARPA* Image Understanding Workshop, T. Strat, Ed., pp. 661–668, 1997.

[RTG98] Y. Rubner, C. Tomasi and L. Guibas. A metric for distributions with applications to image databases. *Proc. 8th Intl. Conference on Computer Vision*, 1998.

[RHMO97] Y. Rui, T.S. Huang, S. Mehrotra and M. Ortega. A relevance feedback architecture for content-based multimedia information retrieval systems. *Porc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp82–89, 1997.

[Salt89] G. Salton. *Automatic Text Processing—the Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley Publishing Co., Reading, MA, 1989.

[SAG95] H. S. Sawhney, S. Ayer and M. Gorkani. Model-based 2D&3D dominant motion estimation for mosaicing and video representation. *Proc. ICCV*, pp. 583–590, 1995.

[STC97] S. Sclaroff, L. Taycher and M.La Cascia. ImageRover: a content-based image browser for the world wide web. *Porc. of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp2–9, 1997.

[SM97] J. Shi and J. Malik. Normalized cuts and image segmentation. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 731–737, 1997.

[SH95] D. Slater and G. Healey. Combining color and geometric information for the illumination invariant recognition of 3-D objects. *Proc. IEEE 5th Intl. Conf. on Computer Vision*, pp. 563–568, 1995.

[SC96a] J. Smith and S-F. Chang. Tools and techniques for color image retrieval. *SPIE Proc.*, 2670:1630–1639, 1996.

[SC96b] J. Smith and S-F. Chang. VisualSEEK: a fully automated content-based image query system. *Proc. ACM Multimedia*, pp87–98, 1996.

[SC96c] J. Smith and S-F. Chang. Visually searching the Web for content. *IEEE Multimedia 4*, 3, pp12–20, 1997.

[Smit97] J. Smith. Integrated spatial and feature image systems: retreival, analysis and compression. PhD dissertation, Department of Computer Science, Columbia University, 1997.

[SK97] M. Smith and T. Kanade. Video skimming and characterization through the combination of image and language understanding technique. *Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 775–781, 1997.

[SZ96] S. Smoliar and H. Zhang. Video indexing and retrieval. In B. Furth, editor, *Multimedia Systems and Techniques*, KAP, 1996.

[SS96] A. Soffer and H. Samet. Retrieval by content in symbolic-image databases. *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, vol. 2670, pp144–155, IS&T/SPIE, 1996.

[SD96] M. Stricker and A. Dimai. Color indexing with weak spatial constraints. *SPIE Proc.*, 2670:29–40, 1996.

[SS94] M. Stricker and M. Swain. The capacity of color histogram indexing. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 704–708, June 1994.

[SP95] K.K. Sung and T. Poggio. Example based learning for view-based human face detection. MIT technical report.

[SB91] M. Swain and D. Ballard. Color indexing. *Intl. Journal of Computer Vision*, 7(1):11–32, 1991.

[Swai93] M. Swain. Interactive indexing into image databases. *SPIE Proceedings*, volume 1908, 1993.

[SC96] T. F. Syeda-Mahmood and Y.-Q. Cheng. Indexing colored surfaces in images. *ICPR*'96.

[Syed97] T. F. SyeSa-Mahmood. Data and model-driven selection using color regions. *Intl. Journal of Computer Vision*, 21(1/2):9–36, 1997.

[Tego94] D. Tegolo. Shape analysis for image retrieval. *Proc. of SPIE Storage and Retrieval for Image and Video Databases II*, No.2185, pp.59–69, 1994.

[UF85] G. J. Upton and B. Fingleton. *Spatial data analysis by example. Vol I.* John Wiley & Sons, 1985.

[WKSS96] H. Wactlar, T. Kanade, M. Smith, and S.M. Stevens. Intelligent access to digital video: the infomedia project. *IEEE Computer*, vol. 29(5), May 1996.

[WK96] X. Wan and C.-C. Jay Kuo. Color distribution analysis and quantization for image retrieval. *SPIE Proceedings*, 2670:8–16, February 1996.

[WAM97] D. Wettschereck, D. W. Aha, and T. Mohri. A review and evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, Special issue on lazy learning algorithms, 1997.

[WL93] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *Pattern Analysis and Machine Learning*, 11:1101–1113, 1993.

[YL95] B.-L. Yeo and B. Liu. Rapid scene analysis on compressed videos. em IEEE Trans. Circuits Syst. Video Technol. 5, 6, pp533–544, 1995.

[YW95] H. Yu and W. Wolf. Scenic classification methods for image and video databases. *SPIE International Conference on Digital Image Storage and Archiving Systems*, 2606:363–371, 1995.

[ZMM95] R. Zabih, J. Miller and K. Mai. A feature-based algorithms for detecting and classifying scene breaks. *Proc. ACM Multimedia*, pp189–200, 1995.

[ZKS93] H. Zhang, A. Kankanhalli and S. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems 1*, 1, pp10–28, 1993.

[ZSW95] H. Zhang, S. Smoliar and J. Wu. Content-based video browsing tools. In A. Rodriguez and J. Maitan, editors, *Symposium on Electronic Imaging Science and Technology: Multimedia Computing and Networking*, pp. 389–398, SPIE vol. 2417, 1995.