Analysis and Representations for Automatic Comparison, Classification, and Retrieval of Digital Images

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

Nicholas Read Howe

May 2001

Analysis and Representations for Automatic Comparison, Classification, and Retrieval of Digital Images

Nicholas Read Howe, Ph.D.
Cornell University 2001

Humans beings can easily make abstract judgments of similarity, but current techniques for algorithmically measuring the similarity between two images do so at a very concrete level, measuring simple statistics computed from the raw image pixels. This dissertation develops and evaluates an evolvable framework for computing image similarity that moves toward more abstract forms of similarity, particularly by allowing the comparison of images based only upon certain significant portions. We begin by formulating and stating the area-matching assumption for concrete visual similarity: Two images are likely to be similar to the extent that they comprise equally matched areas of visually similar materials. We develop an infrastructure to test and explore this approach, and extend it to applications such as classification, image retrieval, and object retrieval. The infrastructure extends from early phases of image processing and analysis, through to multiple-image comparisons and frameworks for applying sophisticated learning algorithms. Throughout we apply the best available tests to evaluate the new techniques and compare them to existing methods.

In order to identify areas of uniform visual properties, we develop an algorithm that combines very small regions in a bottom-up fashion to arrive at a segmentation of the image. The algorithm builds models of each region describing its properties, and combines those areas that can be adequately described by a single model. It outputs a multi-tiered segmentation, dividing the image at three levels of refinement to afford a gradation of choices for modules using its results.

The core of the image comparison infrastructure is a flexible representation system that records in a uniform framework the characteristic feature co-occurrences found within any image. Combined with a smoothing mechanism that allows closely related but non-identical feature matches to be recorded and scored, the representation forms the basis of a new similarity metric on images. We compare this similarity metric to other techniques on multiple tests, and find that in many circumstances, although not all, it performs on par with or better than existing methods.

We further develop variations of the original metric on images to allow searches for matches to portions of images. We do this without requiring any additional stored data structures beyond the basic image description for the images searched. Again, we compare our technique to existing methods, showing that one form of it is at least as powerful as any of the other current methods implemented as comparisons. However, no presently known technique does well enough at this task for reliable use in most applications.

Finally, we look farther afield and draw connections between this work and the field of machine learning. First we show that the infrastructure developed for use with images has broader application to other machine learning problems. Conversely, we also demonstrate that boosting, a popular machine learning approach, can significantly improve the performance of our image infrastructure on a classification task.

Biographical Sketch

Born in Stony Brook, New York on December 30, 1970, Nicholas Read Howe escaped from Long Island in time to avoid acquiring any trace of the local accent. From the age of 3 he lived in Hamden, Connecticut, where he attended the public schools. Upon graduation from Hamden High School in 1989, he matriculated to Princeton University, where he spent the next four years earning an A.B. in physics.

Leaving the university in 1993, he began a two year stint through the Teach for America program teaching math, physics, and French to high school students in Kingsland, Arkanasas. During this time he applied and was accepted to the graduate program in computer science at Cornell University. During the course of a six year stay at Cornell, he met another doctoral student in civil engineering named Susannah Hobbs. They plan to marry in the fall of 2001 and settle in western Massachusetts.

Nicholas Howe plays ice hockey in the winter, volleyball in the summer, and ultimate frisbee in the fall and spring. He enjoys swimming, hiking, and bicycle riding. When not engaged in academic or physical pursuits, he enjoys cooking, reading, and working in clay, fabric, leather, wood, and metal. He also dances, sings, and ties his shoes in a bow.

The author at the start of his graduate studies.

Acknowledgements

Any work of prose must necessarily reflect the life and circumstances of its author. Thinking of the many people whose support and positive encouragement have influenced and guided me in the preparation of this dissertation, I must count myself blessed to have had such friends. I will attempt on this page to demonstrate some measure of my gratitude for their aid and guidance.

Everything begins with my parents and family. Without their gentle nurturing and firm guidance, I would not be who and where I am today. They've been with me all along, and will always have my love and gratitude. I hope that they can always be proud of the son they have sent out into the world.

I have been fortunate to receive guidance from two advisors during my stay at Cornell. Claire Cardie shepherded me through my first attempts at scholarly research and introduced me to the study of machine learning. Later she provided guidance and advice when I needed somewhere to turn. Dan Huttenlocher kindly took me on when my interests shifted to concern themselves with images and computer vision. I could not have completed this work without their help and input, and I thank them both for what they have done for me. I also thank Michael Spivey for his service on my doctoral committee, and for enjoyable conversation at many functions sponsored by the cognitive studies program.

No acknowledgments could be complete without mentioning the many people who have made my life in Ithaca not only bearable but pleasant and memorable. These include the native Ithacans who organized the dances, sings, and other athletic and cultural events that make Tompkins County such a vibrant place to live. They include the students, faculty and staff of the computer science department, who have managed to take a professional association and make it friendly and personal. I will always remember my hockey teammates, fellow Friday evening researchers in artificial intelligence, and the other good people who put off their work to enrich social events with their presence. I would particularly like to thank four people: my longtime housemates Amanda Holland-Minkley and Dan Grossman, who watched the home front with me, and my two officemates of longest tenure, Dave Walker and Stephanie Weirich, who were always there to explain the finer points of type theory to me whenever I desired it.

I have reserved the final note of indebtedness for my fiancée Susannah Hobbs, who came on the scene in the midst of things but has had the most profound impact since then. She has provided me with inspiration, encouragement, and good cheer in boundless measure without the least complaint, and has seen this project through to the end with me.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The digital computer surely ranks as one of the most revolutionary devices ever invented by humankind. After little more than a half-century of existence, computers play a crucial role in the modern economy, transforming both our work and leisure time with the unique services they provide. Computers assist us with secretarial tasks, mediate our communications, entertain us with assorted diversions, and serve as handy extensions to our every endeavor. Nevertheless, the computer still functions as an effective *idiot savant*, able to channel vast amounts of information without any true understanding. Computers can archive lengthy volumes of text, store millions of digitized images, and manipulate endless streams of numbers, all without comprehension, recognition, or the attachment of meaning. Close human supervision remains necessary to ensure that our electronic protégés do not run astray.

In truth, one cannot ignore the tantalizing possibility that computers could do far more, if only they could mimic human judgments without supervision. This observation motivates research in artificial intelligence, natural language understanding, computer vision, and related fields. Each of these fields of research involves interpreting and assigning appropriate meaning to stimuli from the world. Work in computer vision, in particular, might perhaps be better termed *image understanding* in order to emphasize the parallels with speech and natural language understanding. Each takes a raw material, be it text or image data, and seeks to extract semantic knowledge of the material. Both are extremely difficult tasks, approaching the entirety of artificial intelligence in their full scope.

What should a computer be able to do when faced with an image or video as input? At a minimum, answering the simple questions "What is it about?" or even "What is there?" seems a reasonable prerequisite for making deeper judgments based upon the content. Yet even these seemingly simple queries require a great deal of work to produce a full answer. The computer must identify and isolate entities present in the scene. It must determine their boundaries and relationships, and make judgments of relevance. In video, the computer must determine correspondence between entities in individual frames. Thus, within these basic questions lie embedded problems of object recognition, segmentation, distribution of attention, tracking, and comparison. The major lines of research in computer vision are all concerned in some way with the problem of image understanding.

Image understanding is difficult and complex, and unqualified success is not likely in the short term. Happily, a simpler problem exists that includes many of the same issues but is more amenable to partial solutions: *image retrieval*. In general terms, image retrieval entails finding a set of images relevant to a given query, or ranking a set of images by their relevance to the query. If the query is itself an image, then the challenge becomes one of measuring similarity between images. Crude forms of image retrieval are easier to achieve than image understanding because relevance can often be inferred from a small subset of image properties, yet perfect retrieval requires perfect understanding.

This work will focus on image retrieval as a window on the larger problems of image understanding. Nevertheless, because many issues in computer vision are related, a full treatment of the subject requires a somewhat larger scope. In the course of developing better tools for retrieval, the text will also examine techniques for object recognition, segmentation, and other problems. These issues will be addressed as they arise.

1.1 Research Program

We temporarily defer a formal statement of the problem to discuss current issues in image retrieval research and the areas where this work makes a contribution. Nearly all algorithms for image retrieval reduce the problem to the objective measurement of similarity between pairs of images. Accordingly, although the details will be given later in Section 1.2.1, this discussion assumes as a goal the algorithmic implementation of a function that takes two images as arguments and produces a (real) numeric measure of their similarity based upon some set of internal calculations. To be specific, assume further that higher numbers correspond to greater similarity.

What, then, should form the internal basis for calculations of similarity? Researchers have long noted that many levels of similarity may be defined, corresponding to different levels of abstraction [6, 52]. These may range from the most concrete (e.g., red pictures, pictures of circles), through increasing levels of abstraction (e.g., pictures containing some objects or objects, pictures illustrating a particular function), through highly abstract notions (e.g., pictures of France, pictures that make one happy). Though establishing cutoff points along the continuum of abstraction may be difficult, we may nonetheless broadly term these ranges respectively *featural similarity*, *visual similarity*, and *abstract similarity*. While the more complex sorts of similarity appeal most to humans used to making abstract judgments, the sophisticated cognitive processing they require pose formidable difficulties in implementation. In short, interpreting highly abstract notions of similarity seems equal in complexity to the host of general challenges in artificial intelligence that have stubbornly resisted advances for thirty or more years. On the other hand, the most concrete forms of similarity can be implemented much more easily but with correspondingly little reward. The most progress to date has come by following a middle course: use the most abstract form of similarity that can be feasibly implemented. This has typically resulted in systems that operate on some level of visual similarity, meaning that they look for images that contain material with similar sets of directly measurable features.

To summarize the current state of the art, statistical methods that compare entire images currently achieve the best retrieval performance on comparative benchmarks. In particular, a statistic called the correlogram seems superior to the other methods to which it has been compared [41]. Nevertheless, statistics describing full images are inherently unable to capture sub-image details, and attempts to modify them to do so have not been particularly successful. Approaches based on object modeling and search have shown some promise, but not in a general way [61, 49]. One challenge in image retrieval, therefore, is to bridge the gap by developing general statistics that can accurately retrieve images based on an arbitrary portion of interest. Alternately, the challenge can be seen as a move up the hierarchy toward more abstract notions of similarity sensitive to objects and other independent image components.

1.1.1 Motivations and Goals

In the movement towards implementing more abstract forms of similarity, building systems that explicitly account for and allow the representation of distinct entities or regions within an image represents an important step forward. Images are not monolithic structures, and human beings often base similarity judgments upon components that only fill one region in the image (for example, the identity of some object in the foreground). Consideration of the requirements of a system that will handle comparisons between arbitrary regions suggests several conditions that should be met. First, the system should be capable of handling as many possible segmentations into regions as possible. This avoids the difficulties encountered by some region-based approaches, where regions not identified during the initial segmentation phase are not available during retrieval. One way to address the issue is to adopt an agglomerative representation, where the description of each region consists of the descriptions of each of its component parts. This is the approach introduced in Chapter 3, where the component parts are small patches expected to fall below the resolution of objects of interest.

Another consideration concerns the treatment of regions as opposed to entire images. In order to do region-based comparisons seamlessly, the representation of an image must fit in the same framework as that of any region. This consideration points again towards the use of an agglomerative representation, where the representation of any image is simply the combination of the representations of its component regions, however they are defined. Similarly, each region is represented as a combination of subregions, and so on.

Finally, the results from object recognition suggest that success will require a relatively detailed description of each region, in order to successfully discern and classify it. Average color, texture, and aspect ratio are far from sufficient when a collection includes many thousands of images in varying lighting conditions. Unfortunately, the language of general descriptions that will not prove overwhelming to implement is limited. One way to address this concern is to make the region descriptions expandable, so that new information can be included as needed.

This thesis develops a flexible representation in accordance with the above considerations. Designed as an agglomerative structure, it can apply equally to images and image fragments. The representation of the combination of two image subregions into one larger region can be accomplished simply by adding their representations, which are high-dimensional vectors. Varied regions of interest can be composed out of small pieces, thus circumventing the limitations of existing approaches that use predefined regions. The format is flexible enough to accommodate complex object descriptions, including color proximity data.

1.1.2 The Area-Matching Approach

The work described in this thesis fits into the middle of the hierarchy of similarities outlined above. Specifically, it bases judgments firmly on its measurement of visual similarity, seeking out components in the two images that look the same according to a specified set of criteria. This approach is based on a particular assumption about how visual similarity should be defined; the assumption and the approach it inspires will be referred to as the *area-matching assumption* and the *area-matching approach*, respectively.

The area-matching assumption holds that visual similarity between two images is based upon an implied area-preserving mapping between the two image planes. The quality of a particular mapping is proportional to the quality of the match between mapped areas, integrated over the entire image surface. ("Quality of match" here refers to the pointwise similarity between images with respect to a set of local image features deemed to be of importance, e.g., color, shape, etc.) The similarity between two images is defined as the quality of the best mapping between them. Such a mapping may be discontinuous and exhibit other subtleties. For example, the lower right corner of one image may map to the upper left in the other, perhaps because both regions contain images of the same object. Perfect similarity between two images would imply a mapping that gives a perfect score on all relevant features at all points in the image plane. Less than perfect similarity may result either from a much lower score at a few points or equivalently from a slightly lower score at many points. Thus the area-matching approach assesses both qualitative and quantitative factors.

Formulation of the area-matching assumption immediately raises several research questions. What is an appropriate set of local features with which to score candidate mappings? Chapter 3 investigates some basic features that may be measured from the local image data, while Chapter 4 investigates slightly more abstract features representing the surrounding context. Is the area-matching assumption correct in holding to one particular definition of visual similarity? Other approaches to image retrieval, such as the correlogram statistic, do not share the commitment to area preservation required by area matching. Comparative results throughout this work will consider performance relative to such methods; a summary of these findings appears in Chapter 7.

The area-matching approach inspires and meshes with the agglomerative format described in the previous section. This format therefore provides an environment in which to investigate the area matching approach and its implications. By using a variety of different features to describe the image areas, the system can evaluate the effects of literal matching on those features. In particular, by moving away from concrete features like color and texture, the proposed framework can provide a means to test retrieval at a higher levels of abstraction.

1.2 The State of the Art

A number of sources have surveyed work on image retrieval at various points in time over the past decade or two [21, 64]. However, a fairly thorough summary is given here for those unfamiliar with the field. The section begins with a formal definition of the problem in its various forms, considers historical approaches, and then summarizes recent work and the current state of the art.

1.2.1 The Problem

Formally, suppose we are given a set $\mathcal{I}$ of $n_{\mathcal{I}}$ images, $\mathcal{I} = \{I_1, I_2, ..., I_{n_{\mathcal{I}}}\}$, and a query, $q \in \mathcal{Q}$. An image retrieval system $\mathcal{S}$ may be defined as a function from the space of possible queries $\mathcal{Q}$ onto the space of subsets of $\mathcal{I}$:

$$\mathcal{S}(q) = \mathcal{I}_q,\ \mathcal{I}_q \subset \mathcal{I} \tag{1.1}$$

Alternately, many image retrieval systems rank the images in $\mathcal{I}$ according to their presumed relevance to the query. In this case, the image retrieval system is a function from the space of possible queries onto the space of ordered permutations of the integers from 1 to $n_{\mathcal{I}}$.

$$\mathcal{S}(q) = \pi_q(\langle 1, 2, ..., n_{\mathcal{I}} \rangle) \tag{1.2}$$

where π_q indicates a permutation operator. This latter definition is perhaps more common than the former. With the addition of a threshold that selects a portion of the top-ranked images, a system built on Equation 1.2 also encompasses the earlier definition. For this reason, ranking systems give more flexibility and should thus be preferred.

The space of possible queries $\mathcal{Q}$ must be carefully chosen, for the range and flexibility of the queries allowed by any retrieval system help to define its value to any potential user. Perhaps the most natural way for users to interact with such a system is through the use of natural language. Unfortunately, above and beyond the challenges in understanding natural language itself, the translation of natural language into a format suitable for making comparisons with visual data has proven especially difficult. For this reason, most systems define $\mathcal{Q}$ as the space of possible images. In other words, a query takes the form of an image, and the system retrieves related images in response. Such systems have proven far easier to build than those with a natural-language interface. However, there has recently been some research, not yet published, to bridge the gap by using a suitably large database of word-image pairs [69]. This issue will be taken up in greater detail in Chapter 5.

Most image retrieval systems build upon an algorithmic measurement of similarity between images. A similarity function σ that maps pairs of images onto real numeric scores may be trivially converted to a ranking retrieval system by computing the similarity between the query image and the library images, then sorting.

$$\mathcal{S}_\sigma(q) = \mathbf{sorting}\left(\langle \sigma(q, I_1), \sigma(q, I_2), ..., \sigma(q, I_{n_\mathcal{I}})\rangle\right) \tag{1.3}$$

The present work adopts this approach to image retrieval.

1.2.2 Historical Approaches

The problem of image retrieval was originally taken into consideration by the database community, and early solutions to the problem reflect this history. Pointers to images could be stored in a relational database, and retrieved when the associated records were retrieved. This approach may work well if the stored images are needed only in very specific contexts (probably in conjunction with a record to which they are attached) or if each image is attached to a sufficiently rich set of descriptive data that it can be found in a variety of contexts. Unfortunately, these constraints are not acceptable in all applications, so people have continued to search for other indexing methods.

Some organizations (such as stock photo houses and certain government agencies) possess large collections that consist exclusively of photographic data without associated information. Before the advent of computers, these large collections of photographs were only loosely organized. Anecdotally, access to these collections was managed mostly by a few individuals intimately familiar with their contents [57]. The useful size of a collection was therefore limited to what could be encompassed and recalled by a single human mind, and the indexing information was in a particularly volatile form that made duplication and wide dissemination impossible. The limitations of the human mind as an indexing system also drove the search for better methods.

Hand Tagging

With the advent of the use of computers to organize image collections, people attempted to create more useful indices by entering a short descriptive phrase to accompany each image. This procedure, known as *hand tagging*, represents a significant advance over the sole reliance upon human memory. It draws upon expertise already in existence for managing text in databases, and appealed to researchers with experience in this area. Unfortunately, as others have already noted [64], the method is fraught with several significant drawbacks, not all of which may be apparent at first glance.

To begin with, hand tagging requires substantial human effort. While this may be justified in cases where a collection will see substantial use, there will naturally arise situations where users wish to retrieve images from collections where the expense of hand tagging is prohibitive. For example, one might wish to search through archives of video footage for specific frames, but labeling such archives by hand would be extremely arduous. Furthermore, any human endeavor is potentially fraught with error, and experience shows that keywords will be entered incorrectly. One research group reports finding fifteen different misspellings of the keyword "building" in a commercially available collection of human-annotated images [76].

The greatest difficulty with hand tagging lies in the arbitrary nature of the decision to assign a specific tag to an image. Any such decision may not be appropriate in the context of possible future queries. Most images contain many objects and other entities (sky, grass, etc.), raising vexing questions. Should the entities in an image be described individually, collectively, or both? What words should be used to describe them? Consider Figure 1.1 and the multiple possible tags for this image. A system would have to rely on some significant natural language processing and possible knowledge engineering to match any of these descriptions with a query like "blossoms in an alpine meadow," or even "a group of colorful flowers". Such problems are pervasive with hand-tagged descriptions.

Figure 1.1: Problems with hand tagging.
Tags that might be used to describe this image include "flowers", "tulips", "red and yellow tulips", "pink and yellow tulips", "garden", and "sky, mountains, flowers, grass".

Statistical Approaches

In the early 1990s, researchers in the computer vision community became interested in automated approaches to image retrieval. By developing algorithms capable of detecting whether two images are related, they aimed to eliminate the need for human involvement (and thus human error) in the tagging process. This movement coincided with a rise in the use of images as queries instead of words, due to the difficulty in automatically converting a natural-language query into a form comparable to an image.

The first automated approaches to image retrieval tended to be rather simplistic, yet nevertheless achieved a surprising degree of success. A large number of techniques appeared that compute a statistic about an image and use a distance metric on the space of statistics (typically the L_1 or L_2 distance) as a proxy for image similarity. Using this scheme, images in a collection can be ranked according to their similarity to the query image. Examples of this sort of algorithm include those based on color [74, 5, 53, 73], texture [13, 5] and other properties such as shape [5]. Perhaps the most widely successful of these methods is the color histogram method, which has been incorporated in various forms into many systems.

At length, it has become clear to researchers working on image retrieval that reliance on a single feature such as color or texture alone will not distinguish all images sufficiently clearly. In response, developers have created integrated systems that offer users a choice of features [5, 57, 45, 59]. While allowing somewhat more flexibility and the simplicity of a single interface, each subcomponent of such systems still uses a single feature for retrieval. More recently, other systems have been developed that go farther in fashioning single statistics based upon multiple cues in the image [38, 48, 51, 18, 65]. Perhaps the best example of a combined statistic is the autocorrelogram (or correlogram for short) [41], which is relatively simple to compute yet remains closely coupled to both the color and texture in the image. For readers unfamiliar with the histogram and correlogram statistic, a short summary appears in Section 3.4.

Object Recognition

Meanwhile, a parallel line of research in computer vision has developed techniques to detect and recognize objects within images [37]. One family of such algorithms uses object models and a search paradigm to locate candidate objects. For example, object models may be defined as prototypical edge maps and searches performed through an efficient multi-resolution Hausdorff matching [42]. Unfortunately, even with highly efficient search algorithms, the combination of a large number of candidate object models with a large number of possible positions, scales, and orientations within a large number of images makes the full-scale application of such approaches impractical in most cases.

One way to avoid some of the combinatorial problems with model-based search is to focus on finding only certain specialized objects. A notable situation where a model-search framework has been successfully applied is in face detection [63], where the models used are relatively simple. (Interestingly, there is psychological evidence that the human brain also has specialized circuits that perform face detection [4].) Specialized models have also been built for finding other types of objects, such as people [58], cars [61], and fish [49]. Such efforts may eventually bear fruit for the general image retrieval effort, perhaps by using context to choose which models to apply. However, this problem has not seen any satisfactory solutions to date.

More promising from the point of view of general image retrieval is work that seeks to identify the presence of target objects in an image without explicitly searching all possible positions and orientations. For example, one system uses a metric called the earth-mover's distance to determine whether the colors of the target object are present in an image, and at what scale [22]. Images that do not contain the right mix of colors can be ruled out *a priori*. This scheme can be quite successful, but appears to work best when the target objects contain unusual combinations of colors. It fails in cases where the color of the target object is not fixed. Furthermore, all color-based approaches must deal with a lack of color constancy between images created with different cameras and under different lighting conditions.

Region-Based Approaches

Often the semantic connection between two images results from the presence of the same object or objects in both. The leading approaches to object recognition appear unsuitable for image retrieval due to combinatorial factors. On the other hand, statistics based upon entire images cannot discern or recognize individual objects independent of their context in the image. This observation has led to development of image retrieval systems based upon the identification and description of coherent regions in each image. Prominent examples include Blobworld [20] and Netra [23, 50].

Region-based approaches may seem on the surface to be clear winners, but on closer examination they present a few problems. Typically, the region-based system will segment each image as it is added to the collection. Ideally, this segmentation will isolate individual objects and other regions of note, such as sky. By pre-segmenting the image, the need to search all possible locations for candidate objects is avoided. Assuming the ideal case where each region corresponds to an individual object, one need simply classify each region to fully describe the image. Of course, difficult cases can arise where some sort of hierarchical clustering is necessary to make sense of a scene, and the segmentation must decide between several alternate views. More seriously, outright errors in the original segmentation can confound later computations on an image. For example, an object that is accidentally split between several image regions is unlikely to be detected in later comparisons. Even when the segmentation is relatively good in terms of assigning one segment per object, current methods rarely produce segments so clearly delineated that comparisons can be made based on shape.

Although a simple segmentation scheme may be used, such as color quantization [71], the desire for segments that closely correspond to entities of semantic interest has driven the development of more elaborate algorithms. Two that have been recently proposed are JSEG [24] (used in Netra) and a method based upon normalized cuts [70] (used in Blobworld). More will be said on segmentation and its importance in Chapter 2.

Regardless of the segmentation algorithm used, region-based approaches to image retrieval have proven somewhat disappointing. The creators of Blobworld and Netra have not provided comparisons between region-based algorithms and the best statistical approaches. However, there is some indication that the statistical approaches are superior [38]. The reasons for the shortfall are unclear. Any automatic segmentation is imperfect, and this doubtless affects the performance of methods based upon it. However, more subtle difficulties also play a role. For one, the description of each region is typically limited to aggregate properties such as average color, average texture, position, and aspect ratio. These lack the detail found in most object modeling schemes, and thus are more prone to confusion of unrelated regions. Additionally, dividing an image into regions tends to lose the information that one region lends concerning the identity of the others. For example, the identification of one region as a body of water increases the probability that nearby regions might be boats. This effect should be familiar to anyone familiar with Bayesian reasoning, and is implicitly captured in the statistics of full images. However, current region-based systems do not take it into account.

1.2.3 Challenges

The current state of the art in image retrieval shows some successes, but also a number of areas ripe for improvement. This section outlines some of the issues that will surface throughout the rest of this thesis. These are difficult issues, and some will be only partially resolved, leaving further explorations for future work. The discussion below elaborates on some of the issues raised in Section 1.1.

Objects

Perhaps as a first priority, future techniques for image retrieval have to acknowledge and accept the possible presence of multiple distinct entities in an image. Treating each image as a monolithic whole has produced highly successful algorithms, including color histograms and correlograms [74, 41]. Continued work may further improve statistical techniques operating in a similar vein, and there are some applications for which such techniques are highly suited. Yet regardless of their success, these techniques ultimately represent a detour on the path to deeper understanding of images. Focusing exclusively on statistical full-image approaches would hamper development of a deeper understanding in the long run, even if attempts at achieving such an understanding appear less successful in the short term.

Human beings regularly base judgments about images on specific parts of the picture, while disregarding other parts. The very notions of "foreground" and "background" illustrate the fundamental nature of this process. Lacking the concepts of foreground and background (as a full-image retrieval system does by definition) may cause a system to needlessly include parts of an image that distract from the real target. Yet beyond the simple introduction of noise, the inability to separate foreground and background means that certain types of questions can't begin to be addressed. Does a picture contain one tree or three? Is a person standing or sitting? Is it the man's pants that are blue, or his shirt? A system based upon full images can detect at best properties like "some trees," "a person," and "something blue." Making better judgments about these sorts of questions (for example, retrieving images that match a desired answer) cries out for a system that can address and manipulate discrete parts of an image. Thus one major challenge is to develop algorithms that can handle arbitrary portions of images in an appropriate way, while maintaining a sufficiently high level of performance.

Literalism

A second but potentially even more important challenge is linked to the known limitations of most image retrieval systems to date. From the introduction of color histograms to the present, popular techniques using the query-by-example (QBE) approach all share a form of over-optimism that may be termed *literalism.* Specifically, they assume that the images to be found will have certain visual characteristics (colors, textures, shapes, etc.) in common with the query image. In a sense, this limitation is closely linked to the power of the QBE paradigm: in giving an example of the image to be found, QBE provides strong cues that are suggestive but not necessarily comprehensive. In terms of the abstraction hierarchy of similarity described in Section 1.1, literalism implies close ties with the most concrete end of the spectrum. Getting away from literalism would represent a major step towards more abstract retrieval.

The problem with literalism appears when the example provided by the query image includes cues that are unreliable or misleading. Since most retrieval techniques rely heavily on color, for example, misinformation about the colors to be retrieved can be disastrous. Unfortunately, this situation arises often with subjects of great interest: most man-made objects have no consistent coloring, and people themselves often wear clothes of widely varying hue. Probably these impediments explain why researchers have widely favored testing their algorithms on images of natural scenes and animals with more consistent coloring. However natural, such dodges avoid important issues and weaken the push to solve the more difficult problems. Demand exists for retrieval systems that can deal with man-made environments, where artifacts of inconsistent color are common. Developing systems that begin to address this issue is a second major challenge.

While literalism is constraining in the general case, it does provide considerable power when applied appropriately. Sometimes reliable retrieval of closely similar images may be all that an application requires. Given the difficulties inherent in developing systems that treat query images in a non-literal way, we must exploit all available options. As evidenced by the partially successful systems implemented to date, the literal approach is worth exploring in spite of its limitations.

Upon reflection, one sees that different levels of literalism are possible. The simplest form, and one rejected in all but the most constrained applications, is a straightforward comparison of pixel values using a vector-space metric. Most images will fall in some bell curve of distances under such a metric, and the distances computed will be essentially meaningless. However, pictures that are substantially similar to the query image in all respects (perhaps shots of the same scene from the same camera angle and under similar lighting conditions) will stand out as highly significant outliers from the general trend. Thus the first level of literalism serves only to find images that are near duplicates of the query.

Given that one desires to retrieve more than just near-duplicate images, researchers have investigated techniques that allow the relaxation of strictly literal matching. Typically these involve choosing some subset of the query image's features to compare exactly, while allowing other features to vary. For example, using color histograms implicitly assumes that related images will have identical mixes of colors but allows them to be arranged arbitrarily in space. Shape-based techniques assume that certain geometric features will be consistent, but allow color to vary. Other techniques focus on different features, such as texture. Each of these can be useful under different circumstances. For example, Figure 1.2 shows how color, shape, and texture can all be the most useful feature in finding pictures of wolves. Certain categories of images, particularly thematic ones such as "pictures of Paris" may require the complete abandonment of literalism, as two images in this category may have no visual properties in common at all. Such thematic categories are probably beyond the reach of techniques based on image processing alone, at least until significant problems in artificial intelligence and knowledge engineering can be solved.

Human beings can switch easily and flexibly between different modes of literalism, but today's computer systems cannot perform so skillfully. One laudable goal is to work towards rectifying this situation. However, presuming that some sort of literalism will be a fact of life for the foreseeable future, work is also needed to investigate and determine the form it should take.

1.3 Road Map

The remaining chapters in this thesis explore various aspects of image retrieval within the broader context of image understanding. Chapter 2 begins with a survey of segmentation, which is vitally important to region-based retrieval algorithms but also useful in other contexts. It surveys existing work on image segmentation, with special note of algorithms designed for use with retrieval methods. Finding none specifically suited for the requirements of the work in later chapters, it delineates a novel segmentation algorithm for later use. The behavior and performance of this algorithm are analyzed.

Chapters 3 and 4 describe the novel image retrieval framework developed in this thesis. The basic framework and background are described in Chapter 3, together with a description

Figure 1.2: Four pictures of wolves.
Although these pictures are related, no single simple feature (such as color, texture, or shape) is shared in common by all.

of experiments evaluating the approach. Chapter 4 extends the basic framework to include a more complicated image description incorporating regional context.

Chapter 5 focuses specifically on the problem of sub-image or regional retrieval. In particular, it considers the problem of finding images containing an arbitrary object (for which a learned model may or may not be available) in a large collection. This chapter applies many of the algorithms developed in the previous chapters to the modified problem. It also considers how other popular algorithms, particularly those designed to deal with full images, work when applied to this problem.

In Chapter 6, the algorithms developed for image retrieval are shown to be more generally applicable than is at first apparent. The architecture developed in Chapter 3 can be adapted to a class of general problems of machine learning. This chapter describes the adaptation process and some results on a non-image domain.

Conclusions and a summary of the work performed appear in Chapter 7.

Chapter 2

Segmentation

An early decision that the designers of any image retrieval system must face is whether and how to segment the images. Systems not based upon segmentation do not distinguish objects from background or other objects, and thus face difficulty in performing any sort of object-based retrieval. On the other hand, given that automatic segmentation is likely to be the only feasible solution whenever large collections of images are involved, systems that do rely on segmentation must be tolerant of errors made by the segmentation algorithm.

This chapter explores image segmentation with an eye to its usefulness in image retrieval. Beginning with a critical look at the needs of an image retrieval system and previous work in combining the two technologies, it proceeds to examine a few approaches to segmentation before focusing on a some specific segmentation algorithms that will be used in later chapters.

2.1 Benefits of Segmentation

Raw images do not contain much useful information to support further visual processing. Comprising a simple array of pixel color component intensities, a raw image does not explicitly encode regions, objects, boundaries, or any other information that begins to summarize the semantic content. Nevertheless, because people would like to retrieve images based upon semantic properties, a successful image retrieval system must somehow address these higher-level issues. Automatic segmentation helps to provide a higher level capability because properties of image regions can be computed and used as a basis for retrieval.

While useful, segmentation alone is not the key to image retrieval. A recent generation of retrieval systems [20, 50] attempts to provide a solely region-based approach to the problem. The Netra system [50] provides an instructive example. Each image known to the system is segmented into as many as a dozen regions, and each region is then characterized separately. The user can query the system by selecting any number of regions from the query images, and the system returns images that show a close match with the selected regions.

Unfortunately, region-based retrieval systems show disappointing results [8]. There seem to be two reasons for this failure. The first is the failure of the automatic segmentation algorithm itself, which may not return regions that match those the human user might desire. Particularly in cluttered scenes, it is difficult to determine automatically which areas to group together. Thus the region a person wished to find might not exist in the segmentation of the target image. Figure 2.1 shows one image where the segmentations used by both the leading region-based retrieval algorithms fail to give ideal results. A second, more subtle failing is that region-based systems do not take into account the relationships between different regions, which often prove important in determining the identity of an image. For example, a bright red region on a gray background might be a car on a road, whereas a bright red region on a green background might be a flower in foliage. Regardless of which region interests the user, other regions present in the picture may influence its interpretation. This applies especially when the region description is relatively impoverished, not including the subtle features that would otherwise distinguish a car from a flower. In such cases, simple approaches based upon full images can outperform solely region-based mechanisms.

(a) (b) (c)

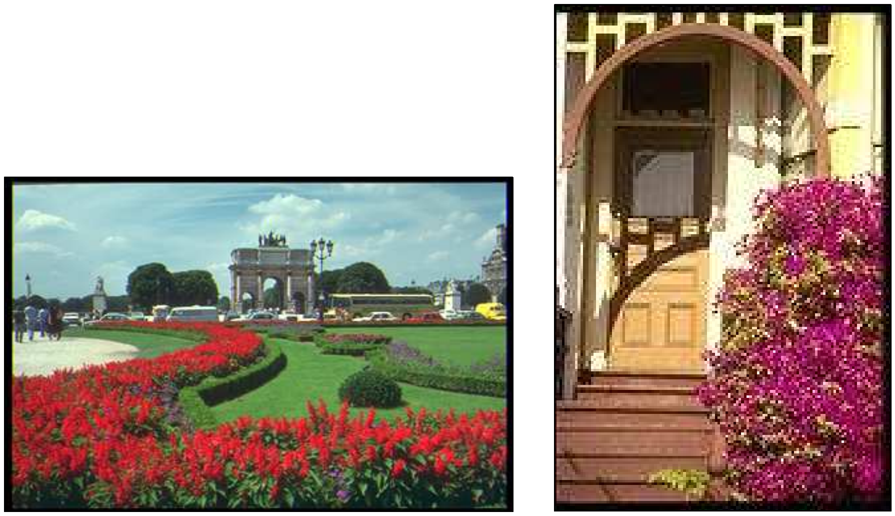
Figure 2.1: Segmentation results for algorithms used by two region-based retrieval systems. (a) Original image. (b) Normalized cuts, used by Blobworld. (c) JSEG, used by Netra.

2.2 Segmentation Algorithms

Many segmentation algorithms have been proposed, but they may be very roughly divided into three main families. Some early segmentation algorithms were *edge-based* procedures that first compute an edge image and then extend and connect edges to separate the image into distinct regions [79]. The disadvantage of this approach is that not all significant region boundaries may correspond to sharp intensity edges. For example, one region may merge gradually into another, as with clouds and clear sky or mingled foliage of different types. On the other hand, strong intensity changes may be evidence of texture within a single homogeneous region, rather than a boundary between two regions. Figure 2.2 shows examples of each of these phenomena.

The JSEG algorithm [24], although not based specifically on edges, falls nominally into this category because it uses gradients to segment an image. Regions are grown from seeds, proceeding in the direction of flat gradients in a *J-image* that is derived from the local color distributions. Using the J-image allows the technique to treat textured regions as homogeneous and to locate diffuse boundaries. JSEG performs the automatic segmentation used in the Netra-2 image retrieval system [23].

A family of *entropy-based* segmentation algorithms seeks to find segmentations that minimized entropy according to one definition or another. The primary example of this sort of



(a) (b)

Figure 2.2: Examples of problem boundaries.
(a) Gradual transitions in clouds and foliage. (b) Sharp lines caused by shadows.

work is the *minimum description length* (MDL) segmentation algorithm developed by Leclerc [47]. Typically, the entropy to be minimized has one term that measures the homogeneity of the segments, and another that measures the fragmentation of the segmentation. High quality segmentations balance the two terms, achieving reasonable scores in each without sacrificing the other. Unfortunately, such methods can be slow, and there is no guarantee that an optimal segmentation will be found.

More recently, research attention has focused on graph-based segmentation techniques, including graph minimum cut, minimum-spanning-tree (MST) clustering, and spectral methods based on the normalized cut [14, 28, 70]. Ishikawa and Geiger propose an algorithm for segmenting grayscale images based upon maximum flows [43]. More recently, Felzenszwalb and Huttenlocher have applied the same graph clustering algorithm to high-level segmentation, in a way that provides an interesting contrast with the technique presented here [27]. These algorithms are united by the fact that they represent an image as some sort of graph, and use fast graph-based algorithms to segment it. This approach has resulted in some notable successes, although a graph formulation can sometimes impose artificial biases on the segmentation process. One advantage of graph-based approaches is that they can have quite good running times: many run in quadratic time [16], and one even in linear time [28]. However, depending on the graph algorithm used, some techniques may not be able to deal with large images at full resolution. For example, some versions of the normalized cut algorithms typically must downsample images in order to achieve reasonable running times [70]. A normalized cut algorithm forms the basis of the Blobworld retrieval system [20].

2.3 A Medium-Grade Segmentation

Generally speaking, segmentation algorithms differ by design in the resolution of the features they will segment. At one end of the scale, the MST segmentation scheme of Felzenszwalb and Huttenlocher [28] can produce a very fine segmentation, with many small discrete regions. (With reasonable parameter settings, it produces on average more than 150 segments from a 128×192 image.) This sort of segmentation is useful when one wishes to calculate local image properties (such as texture, color, etc.) only across homogeneous areas, but isn't concerned with finding complete structures. At the other end of the spectrum are algorithms that produce very coarse segmentations, such as JSEG and normalized cuts. These are useful in finding a few (typically fewer than one dozen) primary regions that comprise an image, but often a single such region will contain considerable diversity. JSEG in particular, with parameter settings that work well for a majority of images, will nevertheless occasionally fail to segment an image at all into more than one segment.

Coarse segmentations have been used for image retrieval in part because they are presumed to segment images on the level of objects. However, any purely intensity-based segmentation will fail to properly group the pieces of any multicolored object (such as humans wearing clothes). For this reason, it perhaps makes sense to step back from the goal of segmenting objects and to attempt to segment pieces of objects instead. This suggests a level of refinement between the extremes: a medium-grade segmentation that produces several dozen segments per image. The remainder of this section describes an algorithm that achieves such a segmentation.

2.3.1 Algorithm Design

A combination of features taken from different existing algorithms leads to an effective moderately coarse segmentation algorithm as developed in the following sections. The algorithm employs energy minimization as the underlying paradigm. However, in order to operate efficiently, the energy model is embedded in a graph algorithm, so that it can be solved using efficient graph minimum-cut approximation algorithms [16]. Furthermore, it adopts a tiered approach by clustering the output of a finer segmentation based upon minimum spanning trees. This cuts down significantly on the work that must be done by the algorithm, without impairing the results. A final tier can merge the medium-grade segments to produce a final coarse segmentation more analogous to that produced by JSEG or normalized cuts.

Throughout the process, the algorithms uses models of the regions it finds to guide its decisions. Model building provides a natural way to motivate segmentation, because a single paradigm encapsulates exactly both the properties that characterize a particular region and how the region differs from its neighbors. Furthermore, model building is a natural partner to graph-based algorithms, because it motivates the weights placed upon links in the graph. To achieve a segmentation, the algorithm builds a graph with two types of nodes, representing both models and regions in the image. The degree to which the model fits the region determines the strength of the links between them.

2.3.2 Models

In principle, the family of models potentially useful to describe different regions is virtually limitless. Regions can be modeled according to their color, shade, texture, depth, or other properties or combinations. Models can further differ in their level of detail, with more detailed models capable of finer distinctions. The choice of a family of models will in large part determine the performance of the algorithm, and therefore deserves careful consideration.

Choice of Models

It might be desirable to combine families of models with different properties. For example, to segment a picture of an African savannah, one might want a simple planar color model for the sky, and several different texture models for the grass, trees, and animals present. Unfortunately, it can be difficult to compare models of widely differing types. We compromise by using a large family of models which capture aspects of both color and texture.

In the segmentation process developed herein, region models consist of two parts: a base intensity and a residual variance profile. The base intensity is a simple (constant, linear, or quadratic) function over the x and y coordinates of the image. Fitting the function that minimizes the sum of the squared error over all the points of a given region yields the best model for the base intensity in that region. The residual variance profile is simply the distribution of residual deviations from the base, as computed from the sample of points in the region. For any given model, there are three separate base intensity functions, one for each color component. Similarly, there are three components to the residual variance profile.

In computing with color images, design choices must be made about what sort of space will be used to represent colors. This work uses two simple color spaces, one based upon red, green, and blue (RGB) components, and the other based upon hue, saturation, and brightness or gray value (HSV) components. Although detailed information on a variety of color spaces can be found elsewhere [80], a brief summary of the relevant aspects of these two spaces appears here. RGB space may be visualized as a cube with simple Euclidean geometry, making it easy to compute the base color functions in the region models. HSV space, in contrast, may be visualized as a cone, with gray value along the central axis tapering towards the darker colors, saturation along the radial axis, and hue represented in the angular component. HSV space is thought to better model human notions of the perceptual difference between colors, and is therefore used to calculate the residual variance profile in the region models. In fact, a slight variation on the standard HSV is used, as described in Section 3.1.1. (Other color spaces, such as the Munsell space, may model human perception better than HSV [56, 80], but they are more difficult to compute with. Using HSV therefore represents a compromise between accuracy and ease of computation.)

Our choice of model encompasses many common types of regions found in natural images, although certainly not all. Constant and linear models of the base intensity can describe the planes of color created by approximately flat surfaces under uniform illumination (with or without small-scale texture), such as sides of buildings and some types of terrain. They may also model some types of sky under diffuse illumination. In addition to the entities described by constant and linear models, quadratic models can describe the shading found on uniformly surfaced simple solids such as cones, ovoids, and the like. More complex regions cannot be captured so well; in practice they are described either by breaking the shape into pieces, using a different model in each portion, or by using one ill-fitting model and sweeping the bad fit into the residual. For example, the shading on a human face cannot be modeled by a quadratic function, but individual portions of it can. Contrarily, the same face might be approximately modeled by a quadratic function, with shading around the nose, eyes, mouth, and other facial features considered "texture". Developing a more flexible class of base intensity functions might ultimately lead to better segmentations, but the current options were chosen as a solid base for evaluation purposes.

We use a texture profile that records the shape of the distribution of residual pixel variances in each component. Thus it includes more information than the mean variance alone. It excludes correlations between components because this would make the space required to store the profile too large. As a result, a region containing a mixture such as bright blue with dark red would have a profile very similar to that of another region containing equal amounts of dark blue and bright red. Fortunately, such combinations are uncommon in natural images. More seriously, the profile also excludes any spatial information about the texture. Thus large spots may be indistinguishable from smaller spots if the residual pixel-level variances are the same. This is a flaw that should be the target of future research. Figure 2.3 shows examples of a few regions and their texture profiles.

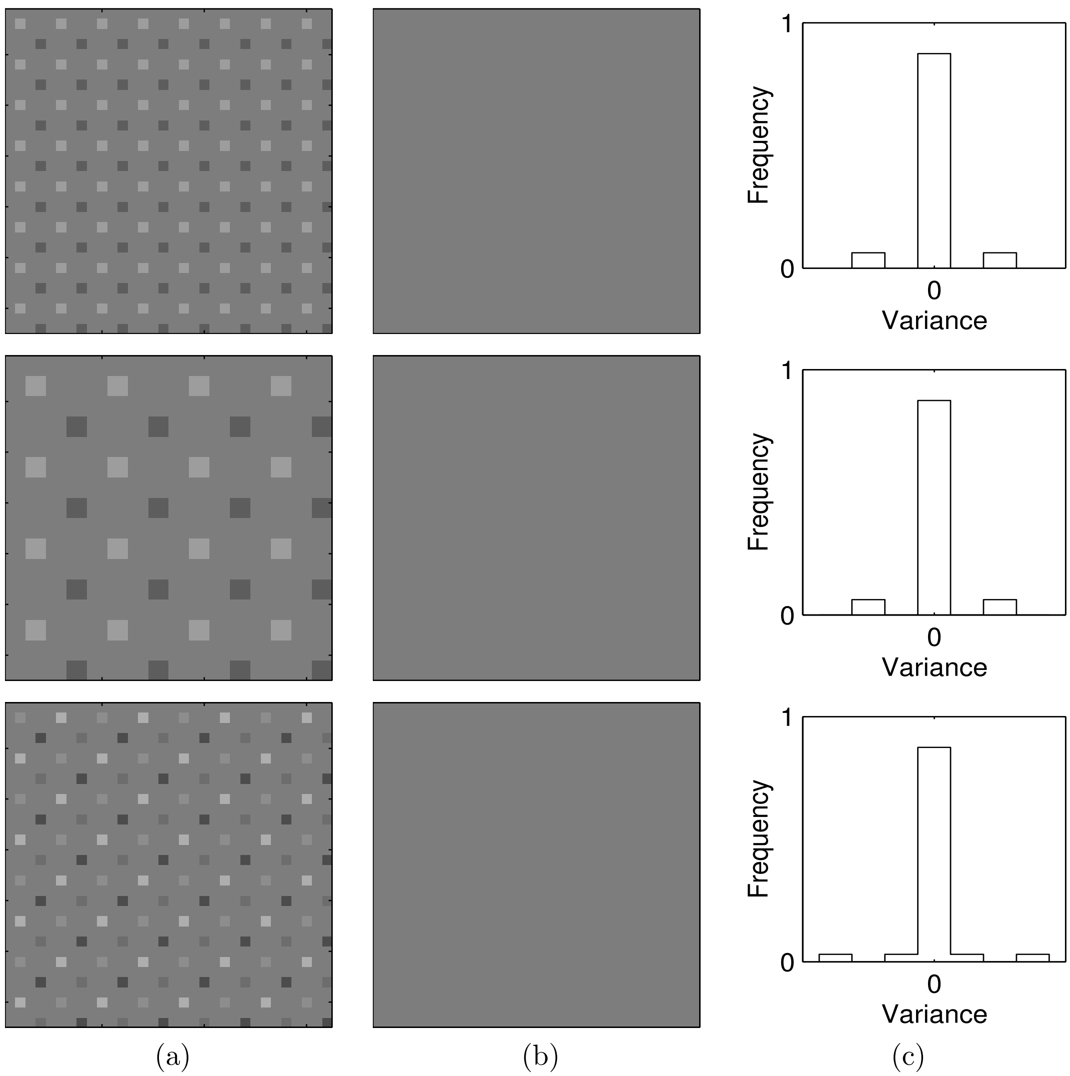
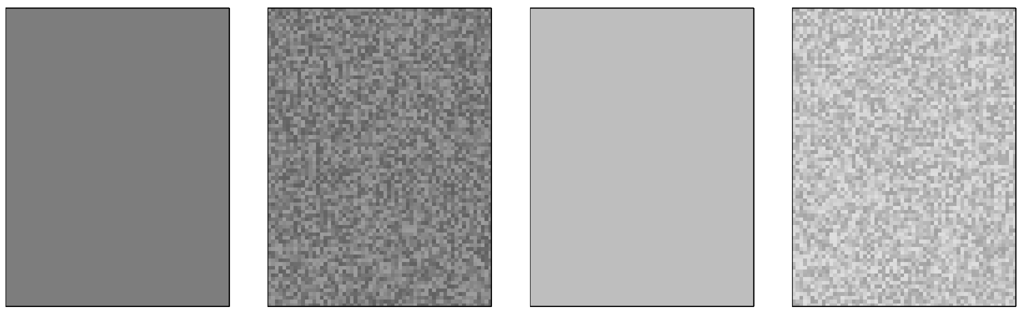


Figure 2.3: Examples of region models for synthetically generated patches.
(a) The region described is split into (b) a base intensity and (c) a profile of the residual pixel variances. Note that all the base intensity models are the same in this case. The mean absolute variance is also the same, but the profile varies between the first two examples and the third.

Model Fitting

The use of models for segmentation depends upon carefully gauging how well a given model fits a given region. With our chosen family of models, we wish to rate the match of both the base intensity and the residual variance profile in a single, combined measure. Thus a smooth dark gray region should match only moderately well to a mottled dark gray region, or to a smooth light gray. It should match even less well to a mottled light gray region. (See Figure 2.4 for an illustration.)



(a) (b) (c) (d)

Figure 2.4: Matching with texture and shading variation.
(a) The smooth dark gray region partially matches (b) the mottled dark gray and (c) the smooth light gray regions. It matches (d) the mottled light gray region less well.

As a first step in calculating the fit of a model to a region, the base intensity model is used to predict the background color of the region. Comparing this prediction to the actual pixel values gives an empirical measure of the residual variance between the region and the base intensity, which can be viewed as a probability distribution over the space of possible residuals. The model itself predicts a particular shape for such a probability distribution. Therefore, the problem of computing the quality of fit between a model and a region boils down to the choice of a metric for comparing probability distributions.

One candidate metric is the Kolmogorov-Smirnoff distance, which is commonly used to compare two probability distributions. This may be computed by converting the probability distributions to cumulative distributions and finding the maximum deviation between the two functions. Unfortunately, because many sorts of changes to the distributions under comparison will not change their maximum deviation, the metric is insensitive to such events. In particular, it is possible to alter nearly all parts of a distribution, representing the majority of the area of the region, without changing its Kolmogorov-Smirnoff distance with respect to another distribution.

For this reason, we choose a related metric that accounts for the entire shape of the probability distribution. Expressing both profiles as cumulative probability distributions v_R and v_M respectively, the distance between the two is defined as follows:

$$D(v_R, v_M) = \int_{-\infty}^{\infty} |v_R - v_M| \tag{2.1}$$

This distance function is also a metric, and it exhibits the desirable properties discussed at the beginning of this section. Of course, the exact probability distributions are not available, and thus a discrete approximation to Equation 2.1 is actually computed using the region's pixels as samples. The fit of a model to a region is the sum of the profile distances for each of the three color components, with lower numbers indicating a better fit.

From the definitions above, two properties become apparent. The fit of a model to the region that generated it is zero by definition, because the variance profiles are the same. Given two regions and their models, the fit from one to the other is not symmetric, even though Equation 2.1 is a metric, because the base intensity functions of the two models may differ. The latter is important because it will allow certain models to dominate in the algorithm described in the next section. The most successful models form the basis of the finished segmentation.

2.3.3 Energy Minimization

One of the advantages of a model-based approach is that it can be understood in Bayesian terms. For any image, we would like to find the most probable segmentation given the details of the image itself and our preconceptions about what a good segmentation should look like. More formally, suppose that I is an image, S is a segmentation comprised of regions R_S, and each such region $r \in R_S$ has a corresponding model M_r. Then the segmentation with the maximum a posteriori (MAP) probability is

$$S_{MAP} = \arg\max P(S|I) = \arg\max P(I|S)P(S) \tag{2.2}$$

The data or observation term, $P(I|S)$ is a function of the match of the image pixels assigned to each region with the model of that region. We choose a negative exponential function of the fit.

$$P(I|S) = \prod_{r \in R_S} e^{-F(r(I),M_r)} \tag{2.3}$$

Here, $F(r(I), M_r)$ denotes the fit of the image pixels in region r to the region model M_r, computed using Equation 2.1 from the preceding section.

$$F(r(I), M_r) = D(v_r(I, M_r), v_{M_r}) = \int_{-\infty}^{\infty} |v_r(I, M_r) - v_{M_r}| \tag{2.4}$$

Here $v_r(I, M_r)$ represents the cumulative distribution of the residuals in region r, relative to the base intensity predicted for the region by M_r, and v_{M_r} represents the cumulative distribution expected for model M_r. For simplicity of notation, Equation 2.4 applies to grayscale images; for color images there should be three similar terms, one for each of the hue, saturation, and value components.

The likelihood term, or prior probability of a segmentation, $P(S)$, involves the number of regions and perhaps the smoothness of their borders. For high-level segmentation, it should tend to favor a very small number of segments. One plausible way to express this is with a penalty applied between separate regions.

$$P(S) = \prod_{r_1 \in R_S} \prod_{r_2 \in R_S} e^{-\Delta(r_1,r_2)} \tag{2.5}$$

Here $\Delta(r_1, r_2)$ represents the penalty for keeping r_1 and r_2 separate in the segmentation S. By definition, $\Delta(r, r) = 0$. Typically, the value of $\Delta(r_1, r_2)$ in the prior probability will based upon the sizes of the two regions, their positions relative to one another, and any common border they may share. More discussion on Δ appears in the following section.

Assuming the forms given above for the probabilities in Equation 2.2 and taking the negative log converts the problem to one of energy minimization. The energy function to be minimized is thus

$$E = \sum_{r \in S} F(r(I), M_r) + \sum_{r_1 \in S} \sum_{r_2 \in S} \Delta(r_1, r_2) \tag{2.6}$$

Energy minimization is a well-established technique in computer vision, and a number of researchers have investigated the solution of problems of this form [33, 79]. One useful approach is to model the minimization as a multi-way graph cut. Potential labels correspond to contending models, and form terminal nodes in the graph. Potentially discrete regions of the image form interior nodes in the graph. Each terminal is connected to every interior node with weights representing $F(r(I), M_r)$, and the interior nodes are connected to each other with weights representing $\Delta(r_1, r_2)$. Given the appropriate weights on the links, finding the minimum multi-way cut is equivalent to minimizing the energy E. Although the minimum multi-way cut problem is difficult to compute (NP-hard), under certain assumptions on Δ there exist approximation techniques that produce results within a factor of the optimal [16]. In practice, these methods often perform still better than the theoretical worst case. However, some care must be taken in order to come up with an appropriate set of models. The models we use actually come directly from the image itself, as described in the next section.

2.3.4 Algorithm Structure

Our segmentation algorithm consists of three main stages that generate a hierarchical tier of segmentations: a bottom-tier or local segmentation consisting of small homogeneous patches, a middle-tier segmentation consisting of larger regions that might be objects or parts of objects, and a top or final-tier segmentation consisting of the most prominent regions of the image. Figure 2.5 summarizes the full segmentation algorithm.

1. Segment image into small patches.
2. Build model for each patch and measure fit between all patches and models.
3. Combine patches into middle-tier segmentation using graph algorithm with energy function that encourages local aggregation.
4. Build new models based upon middle-tier segmentation, and measure their fit.
5. Combine middle-tier segments using gradient-descent algorithm with energy function that encourages growth of extended structure.

Figure 2.5: Steps in our segmentation algorithm.

Bottom-Tier Segmentation

Pixels make a poor starting point for high-level segmentation. It is difficult to build a regional model from a single pixel, or to evaluate the fit of a regional model based on a single pixel.

Thus we begin with small homogeneous patches of the image, averaging about 50 pixels apiece. This size is smaller than the expected size of the final segments, but large enough to calculate regional properties such as texture. Felzenszwalb and Huttenlocher's fast local segmentation algorithm [28] finds the initial patches, using conservative settings to get many small regions. (We set $k = 200$ as opposed to $k = 300$ as described in the original work.) In some cases the segmentation still contains regions much larger than the 50 pixel target size, in which case they may be split arbitrarily into smaller patches. However, this process adds to the algorithm's running time (which is quadratic in the number of bottom-tier regions) and seems to make little difference to the final segmentation, since the split regions quickly recombine in the following stage. Therefore we omit the step in the final algorithm.

Middle-Tier Segmentation

The initial local segmentation forms the lowest tier of the segmentation pyramid. The next tier is found via energy minimization set up as a multi-way graph cut, as described in the preceding section. The algorithm generates a set of potential region models, one from each patch in the initial segmentation. A least-squares fit yields the base intensity model, and the comparison of this model with the original image gives the residual variance. In practice, the profile of the distribution of this residual is stored as the individual pixel samples. This creates a pool of perhaps some 500 models that potentially describe parts of the image.

Using the pool of models, the algorithm builds a graph with one terminal node per model, and one internal node for each local region. (Thus if the initial segmentation had n local regions, the constructed graph will have $2n$ nodes in all.) Each terminal is connected to each internal node with a link strength proportional to the model fit $F(r(I), M_r)$, as calculated from Equation 2.4. Internal nodes are interconnected with link strength $\Delta(r_1, r_2)$, reflecting the prior probability that two regions should be joined.

The choice of Δ will determine in part what sort of segmentation the algorithm produces. One constraint is that the graph-based energy minimization technique we employ is only valid if the second (likelihood or smoothness) term in Equation 2.6 can be expressed involving only individual pairs of the base regions [15]. Furthermore, the initial segmentation provided by the fine-grained segmenter may be unstable, such that two regions that are split in one image may be merged in a very similar image. These considerations lead to the conclusion that Δ should be additive.

$$\Delta(merge(r_1, r_1'), r_2) = \Delta(r_1, r_2) + \Delta(r_1', r_2) \tag{2.7}$$

Accordingly, in this work, we define Δ as the weighted sum of two terms that both exhibit the additivity constraint.

$$\Delta(r_1, r_2) = \lambda_{LOC}\Delta_{LOC}(r_1, r_2) + \lambda_{FAR}\Delta_{FAR}(r_1, r_2) \tag{2.8}$$

Here Δ_{LOC} specifies the number of border pixels shared by r_1 and r_2, and Δ_{FAR} equals the product of the areas of r_1 and r_2. Δ_{LOC} acts as a form of "surface tension" that favors local aggregation into relatively compact segments. Spatially separated areas of similar appearance will not be joined together. In contrast, Δ_{FAR} tends to encourage a qualitatively different sort of segmentation, joining similar regions regardless of position and allowing the formation of complicated, extended structures. For the middle tier of the segmentation, we use $\lambda_{LOC} \approx 0.02$, and λ_{FAR} is set to zero.

Final-Tier Segmentation

Solving the multi-way graph cut yields the middle-tier segmentation, with each set of internal nodes that remain connected after the cut forming a single region. This stage combines many small local segments into a handful (perhaps 50) of regional segments, each fitting some model. Fifty segments probably represents too large a number for most images, yet we deliberately do not set the parameters in the graph model to try to achieve further combinations. The reason is that it is impossible to represent an appropriate stopping criterion in the graph formulation. We would like to terminate the merging process when the cost of the best available merge becomes too high, indicating that no single model would fit the merged regions well. Effectively, this criterion looks at the derivative of E_{fit} rather than its absolute value. The graph formulation cannot represent the derivative, basing its termination instead on the value of E_{fit} itself. One might suppose that using the value directly would provide a better criterion in the first place, but this is not the case. Depending on the quality of the models and the particulars of the image, the value of E_{fit} for an ideal segmentation varies greatly. Thus any fixed threshold on E_{fit} oversegments some images and undersegments others. Using the derivative of E_{fit} works better because it represents the marginal cost of one additional merge. When that merge becomes too expensive (poor in quality), then it is time to terminate the algorithm.

Accordingly, the final-tier segmentation uses a different approach. After finding the middle-tier segmentation, the algorithm recalculates new models for each surviving region, again using the least-squares fit. The final stage further combines the regions using a stochastic gradient-descent approach. The only criterion used in this stage is the fitting energy: at each step, the merge with the lowest fitting energy takes place. In terms of Equation 2.8, this corresponds to setting λ_{LOC} to zero and using a simplified Δ_{FAR} that returns 1 if the two regions have not been merged and 0 otherwise. Thus the last stage of the algorithm tends to build up more extended structures than the previous stage, combining regions that can be described well by a single model, but may be spatially separated.

The gradient-descent technique consists of two steps applied in alternation. At each point during its execution, it maintains a working set of models that describe the image. Out of this set, each region is assigned the model that fits it best. The sum of the quality of these fits (as computed by Equation 2.4) over all the regions is the current score of the algorithm.

$$E_{fit} = \sum_{r \in S_{mid}} F(r(I), M_{W(r)}) \tag{2.9}$$

where $M_{W(r)}$ is the model in the working set assigned to r.

The first step the algorithm tries is to replace one of the models in the working set with a different model (out of the original set generated from the middle-tier segmentation). If it finds a replacement that lowers the score, it makes the change and tries again. Otherwise, it proceeds to the second step, which is to reduce the size of the working set by evicting one model. The regions that had been assigned to that model are assigned new models from the smaller working set, and the algorithm continues looking for replacements.

Evicting a model from the working set actually increases the score because regions must be reassigned to models that do not fit them as well. The rise in E_{fit} therefore corresponds to the penalty incurred by the reassignment. The algorithm proceeds until the cost of the eviction step exceeds a predetermined threshold. The final segmentation has as many regions as there are models in the working set at this point, and each region consists of the middle-tier regions assigned to a particular model in the working set.

Initially the technique just described was applied starting with all the middle-tier models in the working set. However, it turns out that similar results can be achieved starting with a randomly generated working set about twice the size of the final number of regions desired. This speeds up the calculation and, through repeated random initializations, helps to avoid local minima. The results presented here use the lowest score generated on 10 randomized final-stage initializations. Because further repetitions (beyond 10 or so) rarely seem to result in better scores, the greedy stochastic approach may in fact be finding the global minimum solution for most images. However, we have not performed the exhaustive calculations required to check this.

Figure 2.6 shows the three tiers of the segmentation process for one image (Figure 2.6a). The initial segmentation (Figure 2.6b) includes many small regions, although some very smooth patches of sky are large. In the middle step (Figure 2.6c), the smallest regions are aggregated into larger areas, but the "surface tension" of Δ_{LOC} discourages the growth of spindly structures like the full tower. The final stage (Figure 2.6d) puts together the extended structure of the image. Isolated patches of sky connect, and the shape of the tower coalesces into several segments representing the lit and unlit portions.

Discussion of Design Choices

Local aggregation and long-distance matching are both important. A segmentation with simple divisions looks more plausible to the eye than a highly convoluted one. On the other hand, two separated patches of the same material should be somehow linked in the final segmentation. It might seem more straightforward to use a single pass through the graph cut routine, using a linear combination of Δ_{LOC} and Δ_{FAR} to achieve both local aggregation and long-distance matching. Preliminary studies suggested that this often resulted in a segmentation that did neither very well. Thus the process is split into the two separate segmentation tiers. Using Δ_{LOC} first alone aggregates local regions of similarity and finds smooth boundaries. The final stage joins spatially separated regions that are similar. (Effectively, this is like using a nonzero Δ_{FAR}, although the greedy approach described above does not explicitly consider such a smoothness term.)

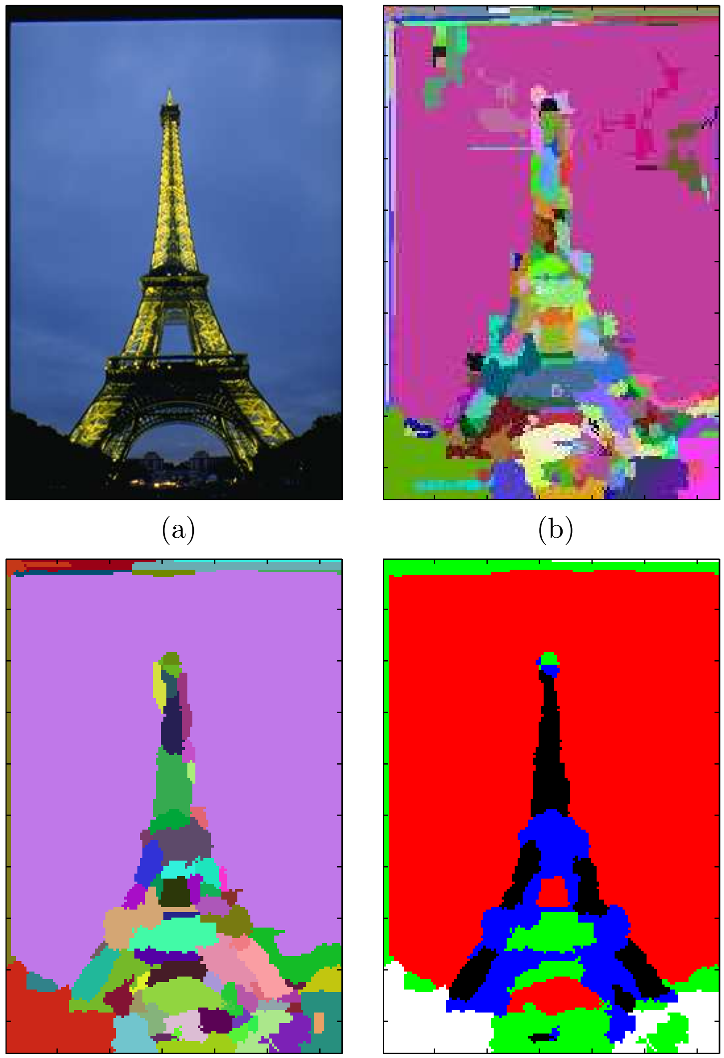
As an alternative, one might imagine using the stochastic final-stage technique for the entire segmentation and eschew the graph-based method entirely. This works reasonably well, and the results can be computed quickly. However, the segmentations produced appear qualitatively to be very slightly less desirable than those generated by the hybrid system.

2.3.5 Results

We ran our segmentation algorithm on several hundred images, after testing to determine suitable parameter settings (constant multiplier on Δ_{LOC}, merging threshold for the final stage, etc.) All the segmentations presented in this paper use linear models, with the same parameter settings. (Values for the parameters are as specified in the preceding section.)

Successes

On most images, the algorithm does quite well. Figures 2.7 and 2.8 show several examples, giving both the original image and the final segmentation in false color. Notice that each distinct region in the image corresponds to a region in the final segmentation, and spatially discontinuous regions are recognized. In some cases, the algorithm detects boundaries that are difficult for the eye to discern. This shows one of the advantages of modeling the regions: a good regional model can help to make decisions in areas that are locally ambiguous. Even on images that are segmented well, there are occasionally areas that don't fit any model well, usually on



(a) (b)

(c) (d)

Figure 2.6: Three levels of segmentation.
(a) The original image is first split into (b) small-scale segments, which are clustered into (c) local regions and finally linked with (d) long-range connections. Note that in (d), the sky visible through the tower has been merged with the sky on the outside.

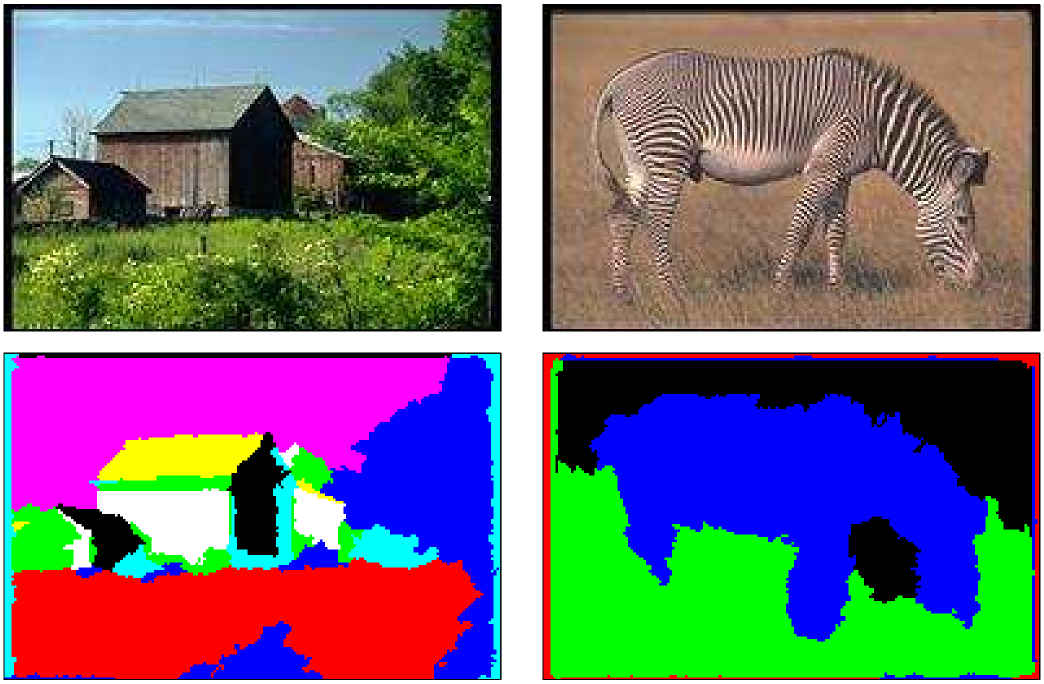
the boundaries between regions. These tend to get aggregated into a single region with high variance. Identifying such high-variance regions may prove important for some applications.

(a) (b) (c)

Figure 2.7: Examples of good segmentations.

Challenges

Some images are difficult to segment properly, either because distinct regions tend to merge, or because single regions tend to be split. These are not necessarily failures of the segmentation algorithm *per se*; rather they indicate that the choice of models was not sophisticated enough. For example, in Figure 2.9a, the sky is merged with part of the sail. As it turns out, both are fit well by the same model in this image. Conversely, in Figure 2.9b, the background is split into several pieces, because no single model fits the whole area well. Clearly there are cases where base intensity plus variance does not tell the whole story. In particular, it would be interesting to look at more complex texture models. Figure 2.9c is a more subtle case. A region has developed in the center of the image which has a very wide variance. The result is that



(a) (b)

Figure 2.8: More examples of good segmentations.

this region includes everything that doesn't fit any of the other region models well. Depending upon the application, this could be desirable or undesirable.

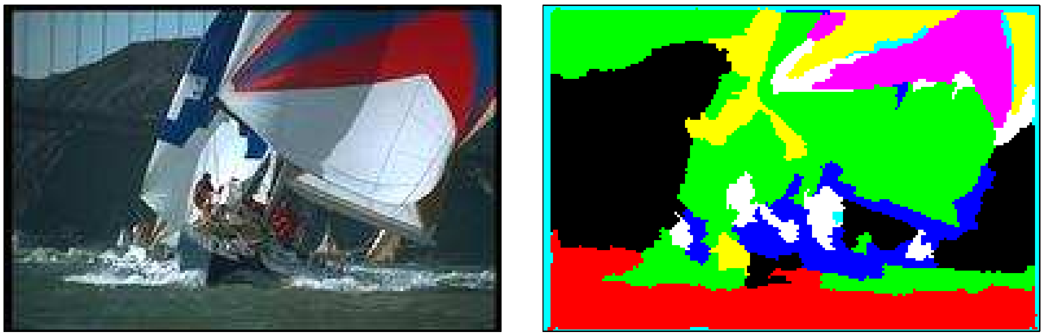
For an instructive contrast, consider the application of Felzenszwalb and Huttenlocher's graph clustering algorithm to the high-level segmentation problem [27]. This approach deals with the image at the pixel level, and creates clusters of pixels that are relatively similar in color and position. The clusters are quick to compute, and the results often agree with those of our algorithm. On the other hand, the segmented regions do not necessarily have the clear interpretation a model provides, and therefore areas that seem unrelated are sometimes connected. Furthermore, there is no mechanism for joining spatially disconnected areas.

2.3.6 Summary

This chapter develops a segmentation algorithm that produces multiple tiers of segmentation. Each tier has a different interpretation and potential use. The coarsest tier is a high-level segmentation designed to pick out the most significant regions in the image. The middle tier is a neighborhood segmentation designed to pick out large patches that can be modeled by a single model. The bottom tier is a local segmentation designed to isolate small coherent patches.

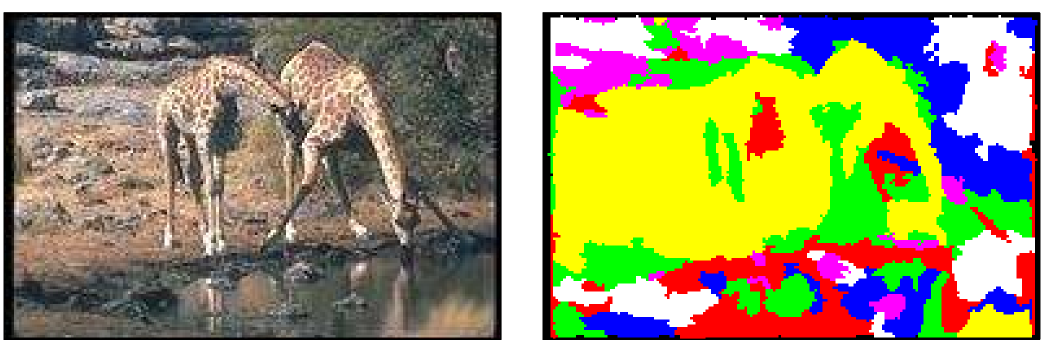
In the end, the test of any segmentation algorithm lies in its potential applications. We developed our high-level segmentation with image analysis in mind, for tasks like classification and retrieval. The middle tier is potentially useful in building geometric models of a scene or a similar application. The bottom tier is an off-the-shelf algorithm and has many applications in low-level vision.

The use of graph-based techniques and region models characterizes the approach detailed



(a)

(b)



(c)

Figure 2.9: Examples of problematic segmentations.

herein, and both affect the design of the algorithm. Although the graph algorithms we use are not new, considerable adaptation was required to apply them to our problem. In order to do so, we developed a flexible class of region models and a novel way of evaluating their fit to the image. In the end, this may have been the most important piece of this research, as the same methods proved useful when applied with other optimization approaches. Nevertheless, as sources of motivation and inspiration, this work is heavily grounded in graph algorithms.

Reliable high-level segmentation that works on a wide range of images is an important result. The techniques outlined in this chapter provide a good start toward that goal, and lay a strong framework for further work through the development of more powerful models. Good high level segmentation is not an unattainable dream.

2.4 Storage of Segmentation Maps

The course of this work required the calculation and storage of segmentation maps for over 20,000 images. In order to reduce the space required on disk and in memory, the segmentation maps were stored in compressed form and decompressed before use. In the process, some interesting results were uncovered concerning the most efficient compression method.

2.4.1 Background

The segmentation map, or *image partition*, for an $m \times n$ image may be most simply expressed as an $m \times n$ array of labels, where each label $s_{ij} \in \{1 ... n_s\}$ denotes the segment of the corresponding pixel, identified by numbers ranging from 1 to n_s, the total number of segments in the image. This label array may therefore be stored as a gray-level image, and standard lossless compression algorithms for such images may be used. However, most segmentation maps have additional properties which allow for increased compression when exploited. For example, all the pixels of a particular segment are often contiguous. Furthermore, the exact label on a particular segment is unimportant, as long as each segment is uniquely identified.

One representation for segmentation maps that has been around for a long time is the Freeman chain code and its variants [29, 47]. In this format, the segmentation boundaries are described in terms of their turnings and branchings. Once the boundaries have been encoded, an appropriate labeling can be generated by a simple sequential scan. (If any of the segments are disjoint, then the labeling will also have to be stored to indicate which disconnected pieces belong together.) The merits of the chain code have been debated at times, but it nevertheless stands as the representation of choice for applications where the encoded size matters [47]. The results presented here show that there is a range of segment densities where it is not always the most efficient representation for image partitions.

Given an $m \times n$ image, there are $(m-1) \times (n-1)$ points located at the corners of four pixels that are potential junctions of segment boundary lines. Given that boundaries do not dead-end, there are twelve possible ways of filling these points, illustrated in Figure 2.10. Using the chain code, an arbitrary junction on the chain has seven distinct possibilities: continue straight, turn left or right, form one of three possible tee junctions, or make a four-way junction. On the other hand, simply scanning the junction points in a diagonal fashion limits the possibilities to at most four in each case. Assuming that the neighboring points above and to the left are known (from the previous scan line), only the connections below and to the right are unknown. The chain code avoids explicitly encoding the blank space between the segments. However, if the average segment size is small so that there is not much blank space, the scanning approach can prove more efficient.

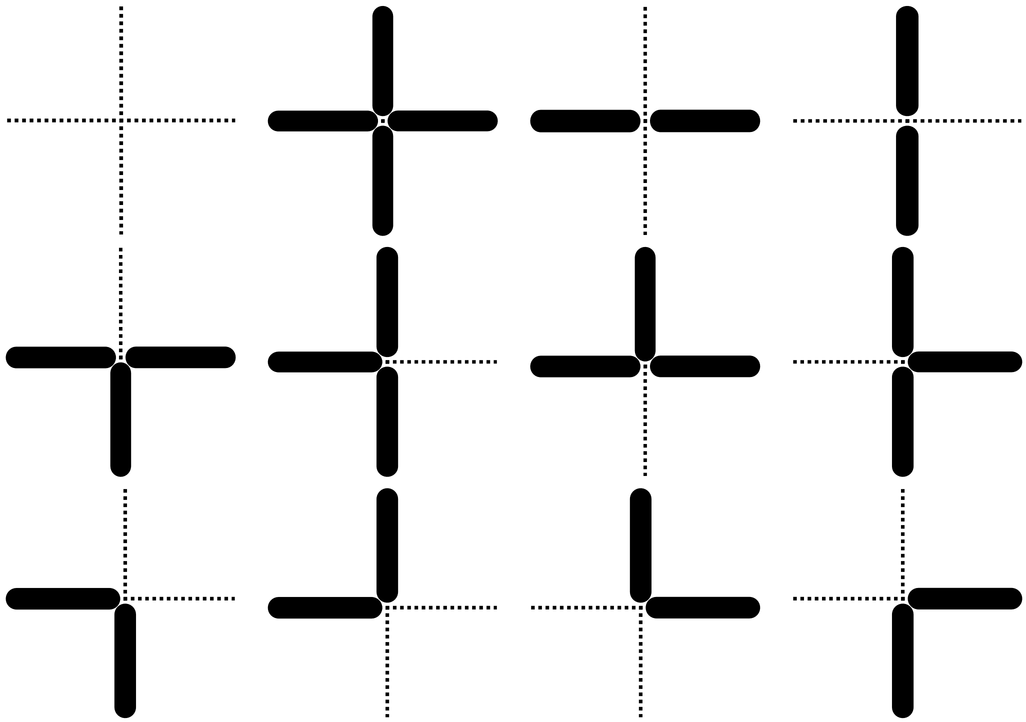


Figure 2.10: Twelve possible junction types.

2.4.2 The Algorithms

Both algorithms were implemented in C, according to the details given in this section. Algorithm A is the chain code, and Algorithm B is the scanning code. Each was tested on a range of segmentations ranging from coarse to fine. Since both algorithms rely on a separate labeling mechanism to handle disjoint segments, only partitions with connected segments were used.

Algorithm A

Our chain code implementation differs in its details from Freeman's original specification [29], in particular because the chains may branch and rejoin. Algorithm A assumes that all the segment boundaries are contiguous, although it may be trivially altered to include multiple disjoint boundaries. The encoding begins by specifying the dimensions of the image, using four bytes, and the starting junction, using another three. It then proceeds to crawl over the remaining boundary structure, describing it as it proceeds and keeping a history of where it has moved. From the current location, it looks for unexplored territory to the left, up, right, and down. If it finds an unexplored edge, it moves in the appropriate direction and records a code specifying the type of junction found (see Table 2.1). The lengths of these codes were chosen according to the empirical probability of each type of junction, also given in Table 2.1. Empirical values were determined by simple counting in a random selection of segmentations.

If no unexplored territory is found at the current position, the algorithm retreats one position back towards its starting point and looks again. It thus methodically fills in all the side chains along the route. When it finally returns to the starting point, it has described the entire boundary structure and can terminate. A decoding algorithm can follow the same sequence of

Table 2.1: Codes used by Chain-Code Algorithm.

Code	Meaning	Frequency
00	Straight	42.6%
01	Right Turn	23.6%
10	Left Turn	22.8%
1100	Four-Way	1.2%
1101	Straight & Right	3.4%
1110	Straight & Left	3.4%
1111	Left & Right	3.1%

Table 2.2: Codes used by Diagonal-Scan Algorithm.

Incoming	Outgoing	Code	Frequency
Top & Left	Bottom & Right	10	6.9%
	Bottom	110	15.5%
	Right	111	12.7%
	None	0	64.9%
Top	Bottom & Right	11	7.7%
	Bottom	0	55.6%
	Right	10	36.7%
Left	Bottom & Right	11	6.1%
	Bottom	10	34.8%
	Right	0	59.1%

moves and reconstruct the segmentation.

Algorithm B

Algorithm B begins by specifying the dimensions of the image. It then locates all of the boundaries that reach the top and left edges of the image, by specifying the breadth of the empty spaces between them. Once this is done, the encoding of diagonal scan lines can proceed, with the assurance that the junctions to the top and to the left of the scanned junction will always be known. If either the upper or left-hand neighbor indicates that a boundary line passes through the current junction point, then the contents of that point are encoded according to Table 2.2. These points may be termed *force-coded*, since they must be filled with some non-empty type of boundary line. Stretches where no boundary lines enter from the top or left are treated differently, since they represent the interiors of segments and it is rare for them to be filled. (Note that the only possible way a line could appear in such a point is to enter from the bottom and turn to the right. Empirical measurements suggest that this happens in only a small number of junction points, fewer than 5%.) A single bit indicates whether any given stretch of apparently empty junctions has an unexpected boundary line appearing. If it does, the location is specified and the remaining empty stretch is considered again. If it does not, then encoding proceeds with the next force-coded junction.

Table 2.3: Comparative Performance of Segmentation Compression Algorithms

Level	Mean # Seg.	Alg.	Code Time (μs)	Decode Time (μs)	Bytes
Fine	495	A	14.4	25.1	2617
Medium	156	A	10.2	23.0	1616
Coarse	47	A	8.2	20.5	718
Fine	495	B	13.0	26.4	2484
Medium	156	B	13.1	24.6	1705
Coarse	47	B	11.1	22.2	847

2.5 Experiments

Two hundred assorted images were selected to compare the two algorithms. All were 192×128 pixels. The images were segmented by three different algorithms that tend to produce varying numbers of segments. The results, shown in Table 2.3, reveal that both are competitive under different conditions. The chain code algorithm produces a smaller encoding for the coarse and medium segmentations, as expected. For the fine segmentation, however, the diagonal scan algorithm produces a smaller encoding. Encoding and decoding times only vary by a few milliseconds between the two algorithms, and tend to correlate with the size of the encoded representation. It appears that when the segment boundaries reach a critical density, the chain code algorithm is less efficient. Overall, both algorithms work well, since at any grade of segmentation the worse algorithm never produced a segmentation more than 20% larger than that of the better.

Chapter 3

Full-Image Retrieval: The Stairs Engine

This dissertation develops an approach to image retrieval based upon automatic segmentation techniques. To be more specific, the description of an image is built up from a collection of primitive image elements, such as segmented patches of color and texture, along with their spatial relationships. These primitive elements combine with each other to create the semantic content of the image in much the same way that words combine to create the semantic content of a piece of text. Each piece carries little meaning by itself, but in combination with other similar elements can form a meaningful whole. The system developed here deliberately draws on successful approaches to text retrieval by treating images analogously to text documents. Thus the system may be described as a Semantic-Token-Based Automatic Image Retrieval System, henceforth abbreviated as Stairs.

In text, words are a clear choice as the primitive semantic token. The image equivalent will be referred to as an *image token*. The question of how exactly to define an image token is not so easy, and indeed the best primitive unit may differ depending on the task. For the basic version of Stairs, each token corresponds to a small homogeneous segment of the image. The token's description consists of the segment's mean color, texture, and location in the image. These are calculated as described in succeeding sections.

The basic version of Stairs may be seen as directly implementing an approximation to the area-matching approach. Since the contribution of each small homogeneous segment is proportional to its area in the final representation, the strength of a given component represents the area of the image with a given property. The lessons learned about the area-matching assumption through experiments with Stairs are explored in more detail in later chapters.

3.1 Preliminary Image Processing

Before describing the way Stairs compares images, we must digress by explaining how the system processes images. Accordingly, this section describes the way Stairs measures color, texture, and location features in an image, prior to combining these into a standard representation. The following section picks up the central narrative with a discussion of the mathematical underpinnings of the Stairs image representation.

All images proceed through identical initial processing steps. First they are scaled to approximately 24,576 pixels (i.e., 128×192 at a 2:3 aspect ratio). They are segmented into about 500 pieces in a two-step process. First, the Felzenschzwalb-Huttenlocher segmentation [28] produces an initial low-level segmentation. Segments of fewer than 25 pixels are merged with their neighbors, while segments of more than 75 pixels are split arbitrarily, resulting in an average segment size of about 50 pixels. (While this may seem somewhat ad hoc, the details of the segmentation process seem more or less irrelevant to the final result. Even using a rectangular 16×32 grid segmentation results in final image representations that is closely related to the representation generated from the custom segmentation, and can be easily distinguished from those of other images. Probably this is because most significant image components are much larger than 50 pixels, so that the average token only spans one such component.)

Once an appropriate image segmentation has been determined, the properties of each individual segment must be computed. In the comparison algorithm which follows, the segment properties must take on only a finite set of discrete values rather than a continuous range. Therefore, there are two steps in describing a segment appropriately for the system. First, one must measure the properties in question. Second, one must discretize the measurements by assigning them the closest value among a (small) finite set of possibilities. In order to minimize the distortion inherent in any discretization, it is desirable that the finite set cover the continuous range in a uniform manner. The sections that follow detail both the measurement of segment features and the discretization of these values.

3.1.1 Color

The color of a segment may be calculated simply by taking the mean RGB values of all of its pixels. This mean color is then converted into a modified HSV color space for all remaining computation. The modification, illustrated in Figure 3.1, expands the range of red to yellow hues while contracting the green to blue range, in order to better match human notions of color distance. (Because the cathode ray tube (CRT) uses red, green, and blue color guns, recent work typically places these colors 120 deg apart in the standard HSV space. In contrast, older texts on color not influenced by the CRT tend to place blue and green closer than 120 deg [62]. The modified HSV space attempts to be more faithful to the traditional notions of color.) Although several specialized, perceptually uniform color spaces have been devised based upon physiological measurements, they tend to be complicated to work with computationally. The modified HSV color space was selected because it is computationally simple to work with and provides ample precision for a prototype application.

HSV space may be visualized as a cone, with hue encoded around the circumference of the cone, saturation along the radial axis, and brightness values along the central axis, tapering towards the darker end. Perceptual distance between two colors corresponds roughly to distance in this space. Most researchers do not report whether they use a Euclidean metric within the cone, or a different metric that preserves the identity of each separate component. This work chooses the latter course, using the absolute sum of the distances in each component. This corresponds to an L_1 metric, but it also implies that the distance between two colors of identical saturation and value is the arc length between them, not the chord length. This makes intuitive sense, because one would not expect a trip along the shortest path between two such colors to involve any change in saturation. Taking into account the conical geometry, the distance formula for comparing any two HSV color triples (h_1, s_1, v_1) and (h_2, s_2, v_2) appears below. The equations assume that all components are scaled to range between 0 and 1.

$$d_c = d_\theta(2\pi h_1, 2\pi h_2) \cdot \min(s_1 v_1, s_2 v_2) + |s_1 v_1 - s_2 v_2| + |v_1 - v_2| \tag{3.1}$$

where d_θ is simply the difference between two angles:

$$d_\theta = |[(\theta_1 - \theta_2 + \pi) \textbf{ mod } 2\pi] - \pi| \tag{3.2}$$

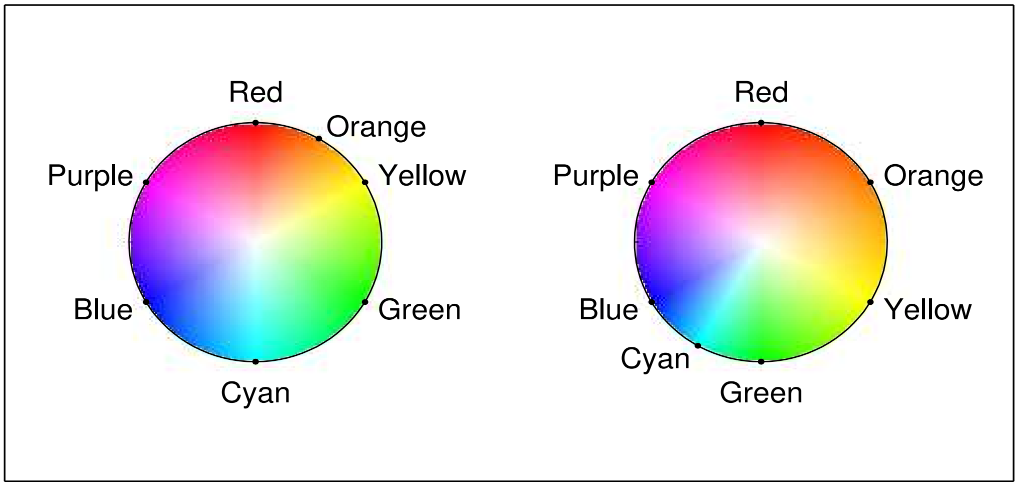


Figure 3.1: Modified HSV color space used in this work.

The HSV color space is discretized by choosing well-spaced seed values and performing a Voronoi decomposition. Stairs uses 28 seeds arranged in a spiral cone, as illustrated in Figure 3.2. This small number of bins produces a relatively coarse covering of color space, but images recolored using this palette are still easily recognizable, as shown in Figure 3.3(a-b). For some applications requiring greater discretion, a 128-bin discretization is used, and a recoloring using this palette (shown in Figure 3.3(c)) shows noticeably higher fidelity than the 28-bin discretization. Distances between two color bins are calculated by comparing their centers, using Equation 3.1.

3.1.2 Texture

The notion of texture encompasses variation at multiple scales, orientations, and degrees of contrast. A full account of the texture in a segment of an image therefore would require excruciating detail: measurement of anisotropy, orientation, and contrast for each color component, potentially at multiple scales [9, 78]. Such descriptions could nevertheless be implemented, using for example the *texture cone* framework and a set of Voronoi cells similar to those used in the color feature [10]. However, current techniques for segmenting different textures and measuring their properties are not completely reliable, and even if they were it is not clear that this wealth of detail would be helpful in the task of general image recognition that Stairs is designed to address. Therefore, the system uses a much simpler notion of texture that simply attempts to measure the total amplitude of small-scale variations present in the segment. ("Small-scale" in this case roughly means smaller than the dimensions of the segment, as explained below.)

Recall that Chapter 2 developed a segmentation algorithm that models an image segment as a quadratic function plus a noise component. The quadratic model captures the large-scale variation of the segment. Any remaining variation (the noise component) is, by definition, evidence of texture. Figure 3.4 shows an image, its quadratic piecewise approximation, and the magnitude of the residual. Notice that all the regions of texture in the original are visible in the residual image, including the giraffe's spots and the leaves of the tree. Unlike many more traditional filter-based approaches to texture, this treatment avoids a strong response at

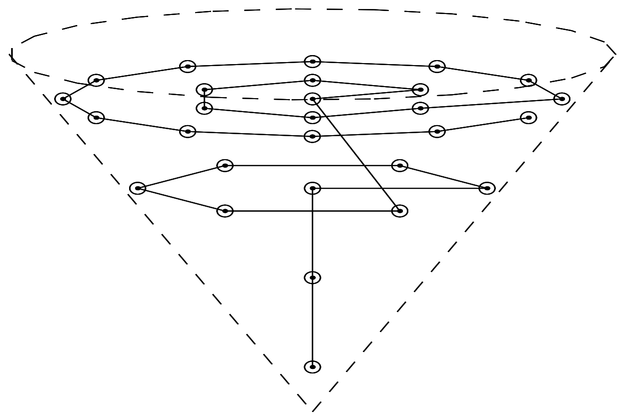


Figure 3.2: Arrangement of Voronoi seeds for color discretization in HSV space.

boundaries between regions of contrasting color, such as the earth and sky. While such edges may be considered texture from one point of view, they arise from a very different processes in the physical world than do the spots on a giraffe, and have less to do with what is in the picture than how it is arranged. Furthermore, object boundaries may contain the strongest contrast present in the image, swamping out more subtle surface variation. Thus for purposes of image retrieval, a texture measure that does not emphasize the edges between regions is more useful than one that includes such edges.

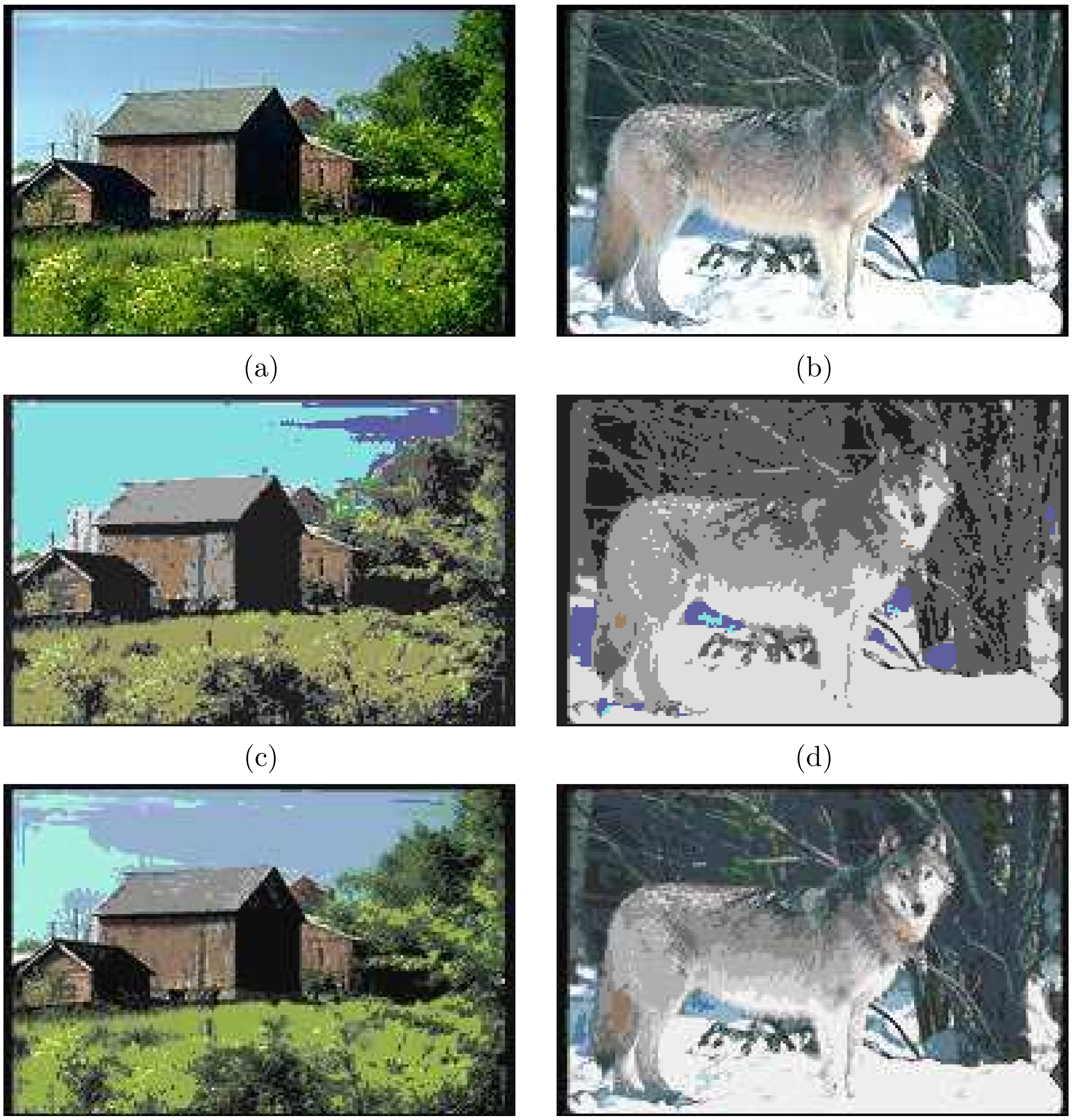
Once the residual *texture image* is produced, there still remains a question of how to measure the texture in a region. The most basic (early) versions of Stairs simply use the mean strength of the texture image across all pixels belonging to the segment. This approach, though crude and somewhat vulnerable to outliers, yields adequate results. However, a much more precise measure of a segment's texture can be had through a somewhat more complicated procedure.

A histogram of the texture image values for any given segment typically shows a peak around zero, with decreasing numbers of pixels at greater absolute values of texture. Empirically, for homogeneous regions in natural photographic images the number of pixels n_τ found at a given texture decays approximately exponentially with the absolute texture value τ.

$$n_\tau \approx \frac{n_s k_\tau}{2} e^{-k_\tau |\tau|} \tag{3.3}$$

where n_s is the number of pixels in the segment. Figure 3.5(a) shows the distribution of the texture components over all the pixels in a region of sky taken from the image shown in Figure 3.4.

The decay parameter k_τ in Equation 3.3 varies from segment to segment, and is a reliable indicator of the texture of the segment. Its value can be estimated for each segment by sampling n_τ at a number of texture values and finding the least-squares fit to the function. It is best to ignore some percentage of the most textured pixels (20% is a reasonable number). These are more likely to include outliers, such as pixels from neighboring regions in cases where the

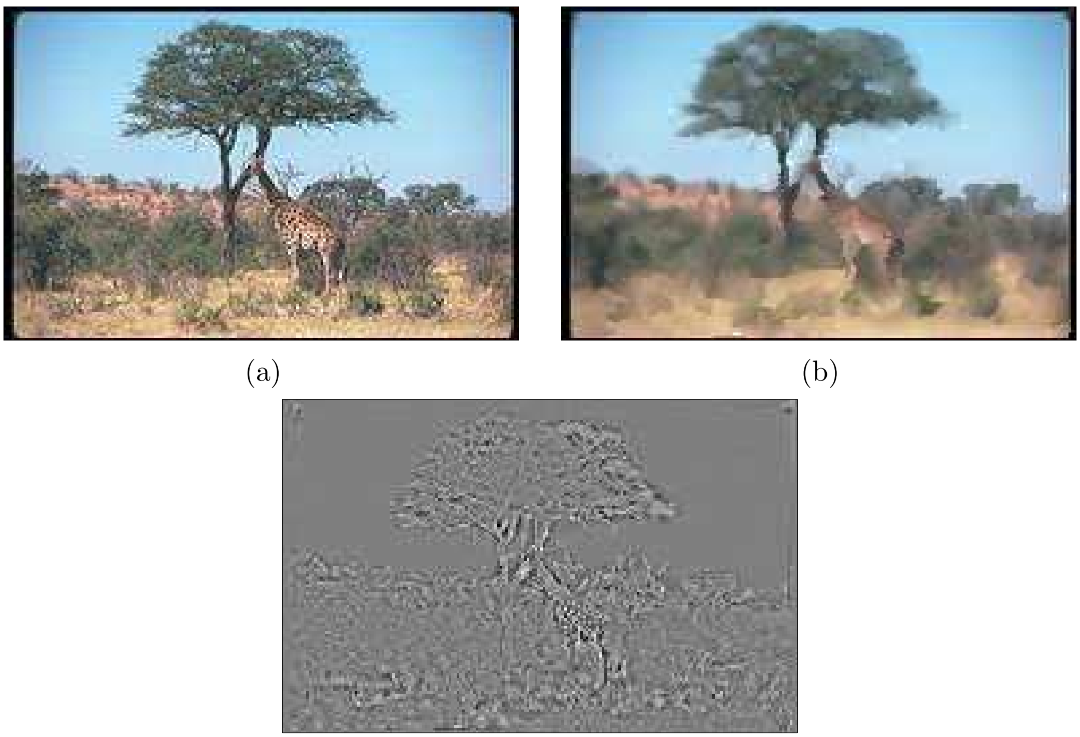


(a) (b)

(c) (d)

(e) (f)

Figure 3.3: Images recolored using quantized palette.
(a-b) Original images. (c-d) Recoloring with 28 colors. (e-f) Recoloring with 128 colors.



(a) (b)

(c)

Figure 3.4: The texture separation process.
(a) An image. (b) A smooth piecewise approximation. (c) The separated texture.

segmentation is imperfect. In practice, the least-squares fit is found using the squared pixel textures, as this gives a slightly more reliable measurement. Figure 3.5(b) shows $\ln(n_\tau)$ plotted versus τ for two regions, a patch of sky and a piece of the giraffe's back. The fitted least-square lines reveal that the sky region has a much steeper slope, corresponding to a larger value of k_τ. (Note that in practice the slopes for natural images are virtually always negative, but k_τ is defined to be positive.) Figure 3.6 shows four example images and the textures computed for their different regions. (For display purposes, the intensity scale has been set so that values of k_τ saturate at levels greater than 100.)

Texture is discretized into just three bins with boundaries at $k_\tau \in \{30, 50\}$, approximately corresponding to smooth regions, somewhat rough regions, and highly textured regions. Distance is calculated between texture bins as though their centers were on a line, arranged from lowest to highest texture.

3.1.3 Location

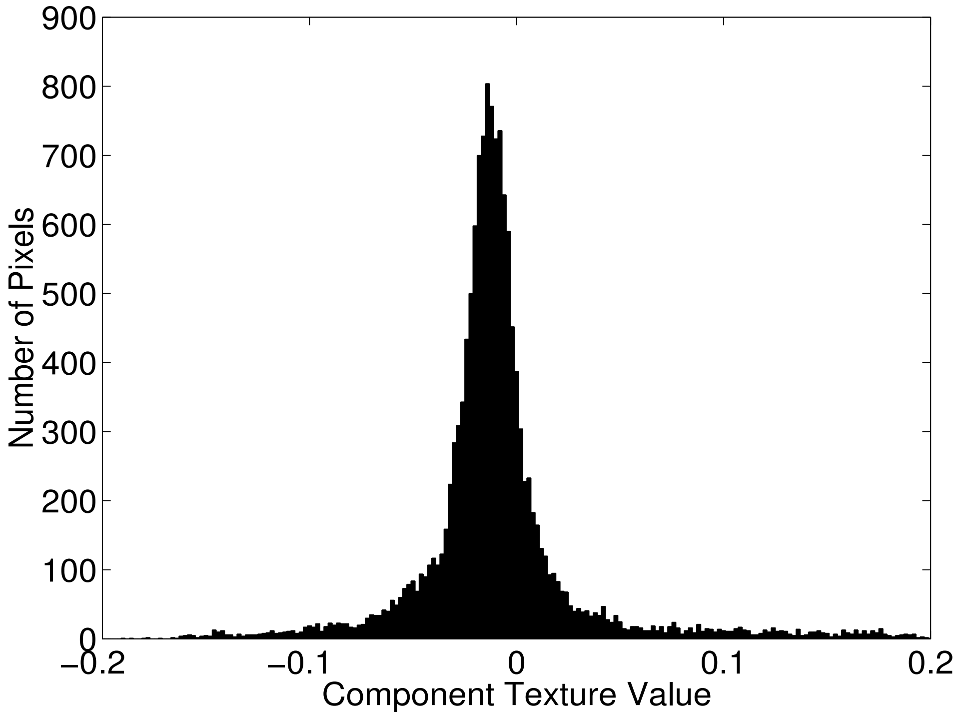
The location of each segment consists of the (x, y) coordinates of its center of mass. Since image dimensions can vary, the origin is chosen as the center of the image, and the unit length is the geometric mean of the lengths of the sides. This provides invariance against changes in aspect ratio, as well as simple uniform scaling of the image.

Early versions of Stairs discretize each spatial dimension into five bins, giving a total of 25 possible location bins (Figure 3.7(a)). Bin boundaries are set at -1/2, -1/6, 1/6, and 1/2. Changing the aspect ratio alters the visible areas of the bins on the edges, but leaves the grid unchanged. It might seem more natural to adjust the grid with the image size, but this leads to spatially distorted matches. A tall narrow object in a tall narrow picture would match a short wide object in a short wide picture, which generally is not desired.

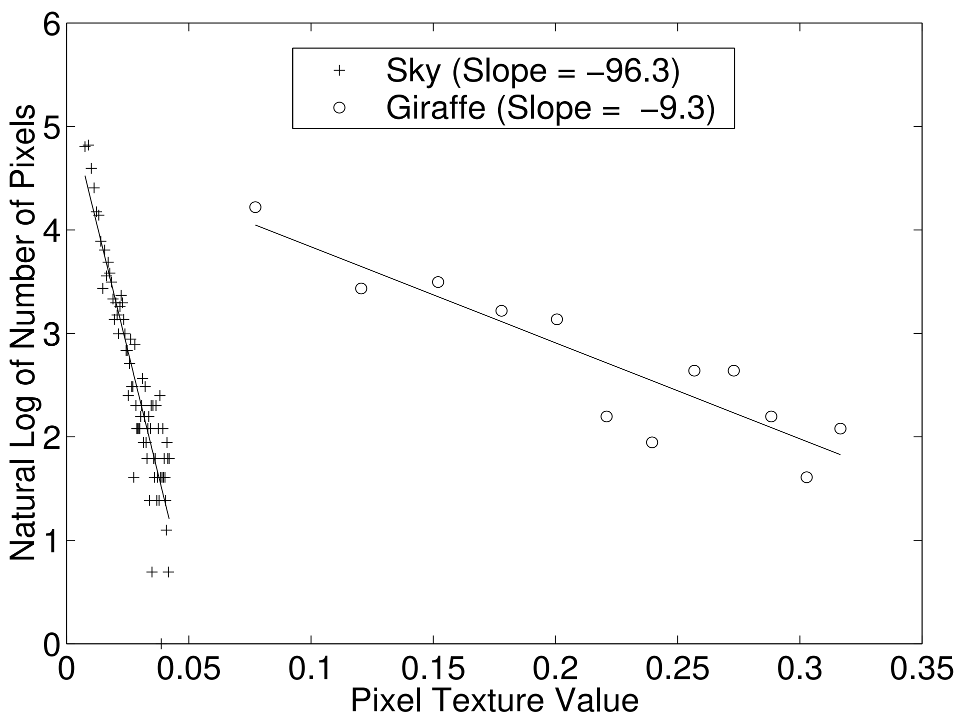
With the discretization described above, some of the outermost bins will be empty, depending upon the aspect ratio of the image. In fact, the four corner bins are never filled at all, no matter what the aspect ratio. This fact prompts a simplification of the spatial discretization in later versions of Stairs. Each dimension is discretized into three discrete bins, with boundaries at ± 0.2. In other words, the quantization of a square image looks like a tic-tac-toe grid (Figure 3.7(b)), with the center square slightly larger than the other regions (occupying 16% of the image area). The modification eliminates bins that are rarely filled, and leads to a more even distribution of the tokens over all the bins. Because the center part of the image is discretized at roughly the same resolution in both schemes, little if anything is lost in terms of discriminatory power due to the modification. The simpler discretization is more satisfactory, although most of the results reported in this chapter are based on the earlier work. In all cases, distance between location bins is calculated using the Manhattan distance between their centers.

3.1.4 Fine Quantization

As a supplement to the work described in the preceding three sections, we also conduct some additional experiments using finer quantizations of each feature. Accordingly, the term *finely-quantized Stairs* will refer to a system using 128 color bins, 5 texture bins with boundaries at $k_\tau \in \{10, 30, 40, 50, 60\}$, and 25 location bins with boundaries at $\{0, \pm 0.2, \pm 0.4\}$. This systems may be used to test whether dividing the feature spaces more finely is likely to affect performance.



(a)



(b)

Figure 3.5: Distribution of pixel textures.
(a) Histogram of the distribution. (b) Fitting a function to the distribution.

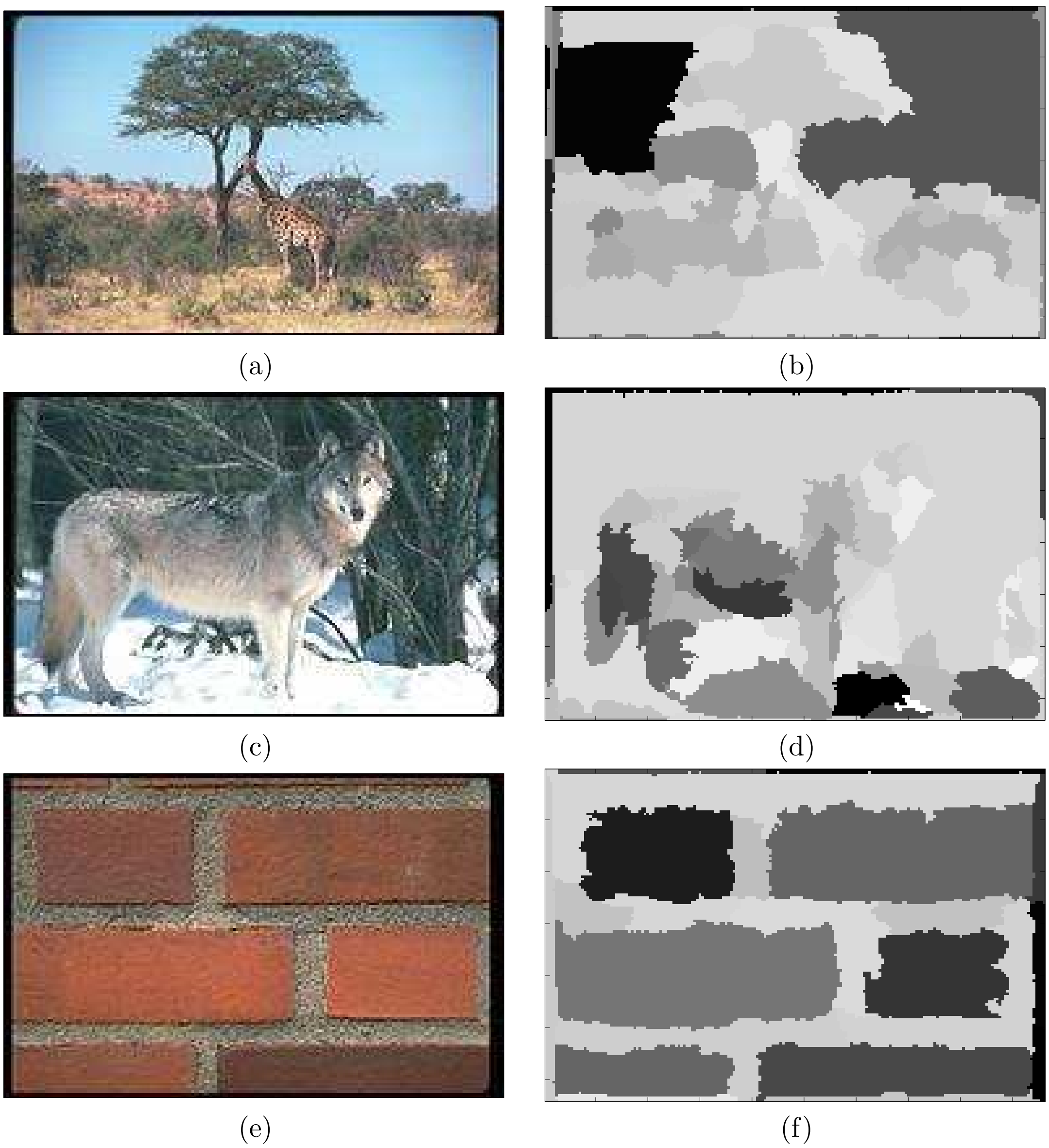
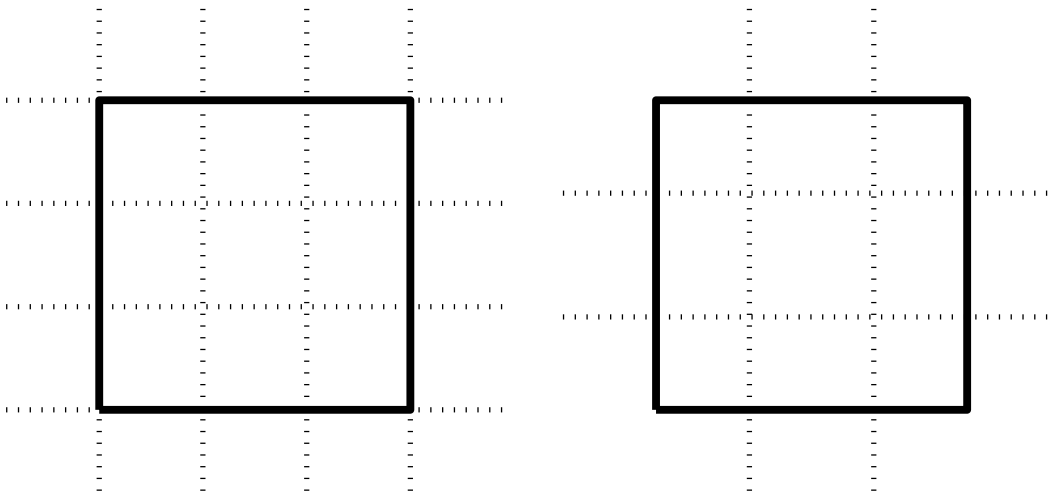


Figure 3.6: Texture measurements for three example images.
Scale is from $k_\tau = 0$ (White) to $k_\tau \geq 100$ (Black).



(a) (b)

Figure 3.7: Spatial discretization schemes.
(a) Original. (b) Modified.

3.2 Mathematical Background

The previous section developed a discretely quantized description of image segments in terms of their color, texture, and spatial location. This section will describe how to succinctly describe and compare images using such tokens. It is worth emphasizing that the mathematical foundations detailed here apply independently of the choice of primitive image token or its description. In Chapter 4, for example, the same foundation is used with tokens consisting of pairwise relationships between segments. Other arrangements are conceivable: local edge elements could be as easily used, without altering the basic algorithm. In fact, one could imagine formulating Stairs comparisons dynamically, using some sort of scripting language to select the type of tokens appropriate for a given task. This would give users the ability to easily build their own similarity measures from scratch, based upon the image features of their choosing.

Interestingly, at least one popularly used approach to image retrieval may be seen as a simple version of Stairs with rather different design choices from those made here. Cast in terms of Stairs, color histogram matching uses individual pixels as a primitive token, and characterizes them solely in terms of color. The histogram approach, while quite successful for its degree of simplicity, is inherently limited in that it neglects other interesting image properties such as texture and spatial layout. A desire to incorporate these additional properties motivates the choice of larger patches as the primitive token in Stairs.

3.2.1 Token Combination

The Stairs token combination algorithm takes as input the set of simple tokens that comprise an image and produces as output a single vector in a high-dimensional space $\mathcal{F}$, amounting to a joint histogram of the token features. In forming the final vector description, Stairs first represents the image tokens in an intermediate space $\mathcal{M}$, then converts them into space $\mathcal{F}$ and adds them together to form the final vector. Thus the final vector in space $\mathcal{F}$ amounts to a joint histogram of the token colors, textures, and locations.

In space $\mathcal{M}$, each image token is described by a discrete feature vector. Thus the first step is to divide any continuous features into discrete bins. the basic Stairs implementation uses 28 bins for the color of each segment, 3 bins for the amount of texture present, and 25 bins for the spatial location. Thus a discrete vector $\mathbf{m}$ in $\mathcal{M}$ representing an image token has $N_{\mathcal{M}} = 2100$ possible values (756 using the modified spatial bins). The collection of tokens found in an image is represented by a set of ordered pairs, typically one per image token, consisting of a vector in $\mathcal{M}$ and a weight w_i. (Technically, if two or more tokens are described by the same $\mathbf{m} \in \mathcal{M}$, then they are represented by a single ordered pair whose weight is the sum of the individual weights of all the tokens.)

$$R_{\mathcal{M}}(I) = \{(\mathbf{m}_1, w_1), (\mathbf{m}_2, w_2), \ldots, (\mathbf{m}_{n_I}, w_{n_I})\} \tag{3.4}$$

This text will refer to $R_{\mathcal{M}}(I)$ as the $\mathcal{M}$-representation of the image I. With $n_I \approx 500$ tokens per image on average, the number of unique $\mathcal{M}$-representations is around 10^{2000}, even ignoring differences in the weights.

Space $\mathcal{F}$ has exactly one dimension corresponding to each point of space $\mathcal{M}$. Thus $\mathcal{F}$ is equivalent to $\mathbb{R}^{N_{\mathcal{F}}}$, where the dimensionality $N_{\mathcal{F}}$ is the product of the number of bins used for each feature in X. (Perforce, in the basic Stairs implementation $\mathcal{F}$ is a 2100-dimensional space.) In other words, $\mathcal{F}$ is defined by an orthonormal basis set of $N_{\mathcal{M}}$ elements, and there is an implicit one-to-one correspondence between the elements of this basis set and the set of all possible vectors in $\mathcal{M}$. This allows the definition of a bijective mapping from the $\mathcal{M}$-representation of an image to the $\mathcal{F}$-representation, $R_{\mathcal{F}}(I)$:

$$R_{\mathcal{F}}(I) = \sum_{i=1}^{n_I} w_i F(\mathbf{m}_i) \tag{3.5}$$

where $F(m)$ is the one-to-one function from vectors in $\mathcal{M}$ to their corresponding basis vectors in $\mathcal{F}$. Note that the conversion from $\mathbf{m}$ to $F(\mathbf{m})$ allows the contributions of each token to be added to the others without blurring their individual identity. The transformation from $R_{\mathcal{M}}(I)$to $R_{\mathcal{F}}(I)$ is completely reversible, thus $\mathcal{F}$ is isomorphic to the space of $\mathcal{M}$-representations (although not to $\mathcal{M}$ itself).

3.2.2 Comparing images

The framework we have set up is analogous by design to simple text retrieval systems [66]. The vectors in $R_{\mathcal{M}}(I)$ correspond to individual words, and $R_{\mathcal{F}}(I)$ corresponds to the word vector representing a document. Continuing the analogy, we compare two $\mathcal{F}$-representation using a cosine metric. Abbreviating $R_{\mathcal{F}}(I)$ as $\mathbf{f}_I$, we have

$$D^*_{\mathcal{F}}(I_1, I_2) = \cos^{-1}\left(\frac{\mathbf{f}_{I_1} \cdot \mathbf{f}_{I_2}}{\sqrt{(\mathbf{f}_{I_1} \cdot \mathbf{f}_{I_1})(\mathbf{f}_{I_2} \cdot \mathbf{f}_{I_2})}}\right) \tag{3.6}$$

In practice, the inverse cosine need never be computed since we are interested merely in the relative similarity of images.

The astute reader will have noticed a problem with the method as described thus far. Consider three images, the first with a red area in one corner, and the other two with similar patches of orange and green respectively. According to Equation 3.6, the red image is equally dissimilar to both. This is undesirable, as red and orange show more similarity than red and green. In particular, if the hues in question happen to fall on either side of a bin boundary, then they may be very similar indeed. The system description needs to be modified to account for this type of similarity, which we term a *near match.*

In concrete terms, the problem is that $\mathbf{m}_i \neq \mathbf{m}_j$ implies $F(\mathbf{m}_i) \cdot F(\mathbf{m}_j) = 0$ even if $\mathbf{m}_i$ and $\mathbf{m}_j$ are near matches. To address this issue, we define a similarity function $S(\mathbf{m}_1, \mathbf{m}_2)$ on vectors in $\mathcal{M}$, such that $S(\mathbf{m}_1, \mathbf{m}_2)$ returns a high value for tokens that are near matches, and a lower value for tokens that do not match as well. Given such a function, we can redefine the difference between two images as a sequence of vector-matrix products.

$$D_{\mathcal{F}}(I_1, I_2) = \cos^{-1}\left(\frac{\mathbf{f}_{I_1}^T \mathbf{S} \mathbf{f}_{I_2}}{\sqrt{\left(\mathbf{f}_{I_1}^T \mathbf{S} \mathbf{f}_{I_1}\right)\left(\mathbf{f}_{I_2}^T \mathbf{S} \mathbf{f}_{I_2}\right)}}\right) \tag{3.7}$$

where $\mathbf{S}$ is the matrix representation of S in $\mathcal{F}$. Note that $\mathbf{S} = \mathbf{I}$ gives the standard cosine metric. Non-zero off-diagonal terms correspond to $F(\mathbf{m}_i) \cdot F(\mathbf{m}_j) > 0$, as desired.

Cholesky Factorization

An alternate way of looking at the near-match issue provides further motivation for using a matrix $\mathbf{S}$ of matching coefficients. Consider a photograph of a red object. Depending upon lighting conditions and exposure, the color of the object in the final image may vary somewhat. Given a collection of photographs of the same object, one will see a range of numbers for the hue, saturation and value. Assuming that the factors causing such variation are uncorrelated (which they will be in a broad collection comprising images taken under many different circumstances), the distribution of the numbers recorded will approximate a bell curve. Similar considerations lead to the conclusion that the measured texture and position of a given image patch may also vary between images.

We have argued that a patch of red in an image may represent an object that in most images would appear a slightly different color (orange-red, or purple-red, for example). Accordingly, for matching purposes, it makes sense to replace the single color red with a distribution of colors in proportion to the probability that they actually produced the observed red. Likewise, the saturation, value, texture, and location recorded should all be replaced by a distribution centered around their observed values. This may be accomplished by redefining $F(\mathbf{m})$ to return a vector representing the distribution of properties that might have produced token $\mathbf{m}$. Alternately, one can multiply the existing $\mathcal{F}$-representation of an image by a matrix that spreads each component appropriately.

$$\mathbf{f}_{spread} = T\mathbf{f} \tag{3.8}$$

Substituting this expression into Equation 3.7 yields the observation that $\mathbf{S} = \mathbf{T}^T\mathbf{T}$. Thus $\mathbf{T}^T\mathbf{T}$ is the Cholesky factorization of $\mathbf{S}$. By using a symmetric $\mathbf{S}$ that has a such a Cholesky factorization, we get an alternate interpretation of Equation 3.7 as computing the cosine difference between two transformed vectors $\mathbf{T}\mathbf{f}_{I_1}$ and $\mathbf{T}\mathbf{f}_{I_2}$.

$$D_{\mathcal{F}}(I_1, I_2) = \cos^{-1}\left(\frac{\left(\mathbf{T}\mathbf{f}_{I_1}\right)^T\left(\mathbf{T}\mathbf{f}_{I_2}\right)}{\sqrt{\left(\left(\mathbf{T}\mathbf{f}_{I_1}\right)^T\left(\mathbf{T}\mathbf{f}_{I_1}\right)\right)\left(\left(\mathbf{T}\mathbf{f}_{I_2}\right)^T\left(\mathbf{T}\mathbf{f}_{I_2}\right)\right)}}\right) \tag{3.9}$$

This alternate interpretation serves as a powerful guide to the intuition in setting the values of $\mathbf{S}$ and $\mathbf{T}$ in the next section.

3.2.3 Spread functions

The matrices $\mathbf{S}$ and $\mathbf{T}$ can be readily assembled from smaller matrices defined on the individual features of the image tokens. $\mathbf{S}$ and $\mathbf{T}$ are each simply the Kronecker product (or direct matrix product) of smaller matrices describing how to treat near matches in each feature. For example, let $\mathbf{S}_{HSV} = \mathbf{T}_{HSV}^T \mathbf{T}_{HSV}$ be the matrix specifying how to treat near matches in the color feature, and define similar matrices for the other features. We produce the 28×28 values in $\mathbf{T}_{HSV}$ using an exponential decay function on the distance of the corresponding bin centers in HSV space:

$$\mathbf{T}_{HSV}(i,j) = (p_H)^{\Delta_H(i,j)} (p_S)^{\Delta_S(i,j)} (p_V)^{\Delta_V(i,j)} \tag{3.10}$$

where p_H, p_S, and $p_V \in [0,1]$ are user-tunable parameters controlling how much to value near matches, and $\Delta_f(i,j)$ is the distances between bins i and j along the dimension f ($f \in \{H,S,V\}$). At one extreme, $p_H = 0$ indicates that the hue must match exactly. (Technically, $p_H = 0$ produces a singularity at 0^0, but for convenience the value at this point is defined to be 1. This corresponds to taking the limit of the family of functions ${p_H}^{\Delta_H}$ as $p_H \to 0$.) In contrast, $p_H = 1$ indicates that any two arbitrary hues will match equally well. This setting is potentially quite useful, allowing the system to ignore a feature deemed not relevant. For example, by setting the p_H and p_S to 1, the system can retrieve color images from a grayscale query. We refer to T_{HSV} and its counterparts T_T and T_{XY} as *spread functions*, and the parameters $\{p_H, p_S, \ldots\}$ as *spread parameters*.

3.3 Practical Concerns

A naïve implementation of the algorithm just described would be unwieldy, especially if the base image token used requires $\mathcal{M}$ to be large. Even with simple tokens, matters can quickly get out of hand. For the original Stairs implementation, with 28 color bins, 3 texture bins, and 21 viable location bins per token, $\mathbf{S}$ requires about 25 MB of memory to instantiate, and computing $D_{\mathcal{F}}$ involves a correspondingly large matrix multiplication. Furthermore, the full $\mathcal{F}$-representation of each image requires about 14 KB to store.

Fortunately, a clever implementation can do better than this. The $\mathcal{M}$-representation of each image can be stored sparsely in about 2 KB. When necessary, the stored $\mathcal{M}$-representation can be quickly converted to the full $\mathcal{F}$-representation. Another savings is due to the special format of $\mathbf{S}$. Because it is the Kronecker product of several much smaller matrices, computing the product $\mathbf{f}_I\mathbf{S}$ is linear in the size of $\mathbf{S}$ instead of quadratic [36]. Essentially, the result can be found by repeated multiplications of the smaller $\mathbf{S}_f$ matrices. With these modifications, the Stairs technique can scale to handle more complex image token descriptions than those described here. The system runs successfully in interactive time with $N_{\mathcal{M}} \sim 10^6$, about three orders of magnitude larger than the setup described here.

3.3.1 Image Retrieval

The fundamental operation in image retrieval is to compare an unknown image with a library of known images, searching for the closest matches. An examination of Equation 3.7 reveals that this can be made quite efficient, since the vector $\mathbf{f}_{I_1}^T\mathbf{S}$ need be computed for the query image I_1 only once, and $\mathbf{f}_{I_2}^T\mathbf{S}\mathbf{f}_{I_2}$ can be precomputed for each library image I_2. In fact, if the library images are stored in their $\mathcal{M}$-representation, then each individual comparison amounts to about 500 array lookups and twice that number of floating point operations. In conjunction with aggressive pruning of the search space, queries on the 20,100-image test library can be

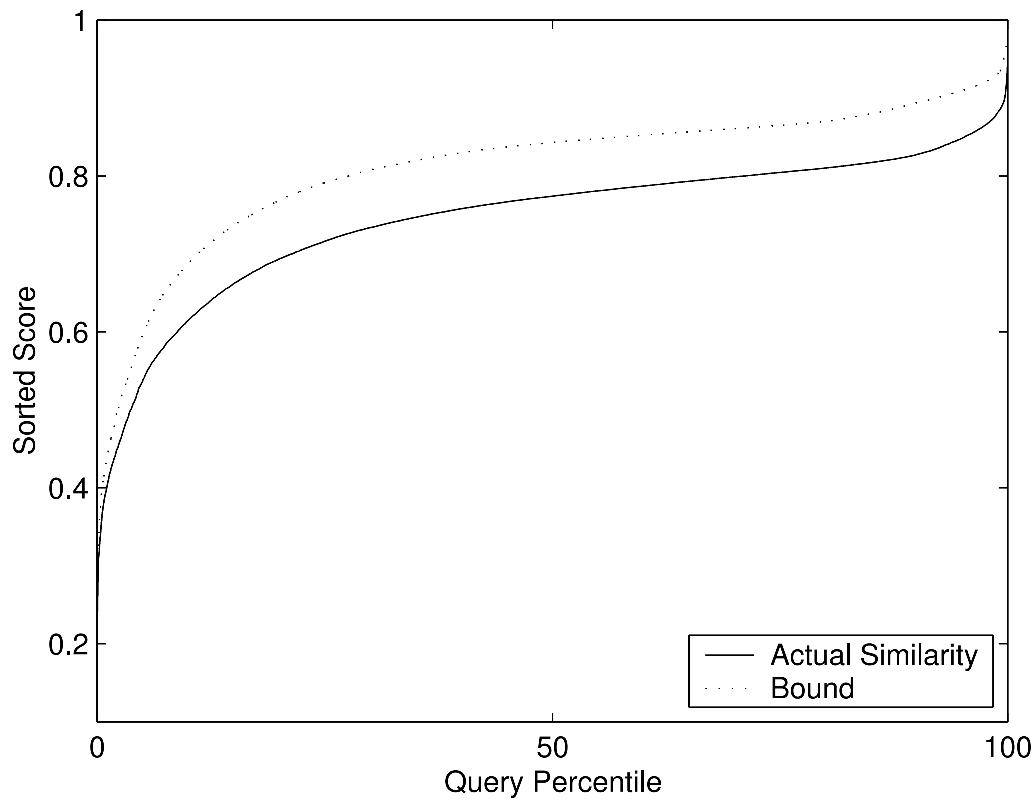


Figure 3.8: Typical separation between actual similarity curves and pruning bounds.

resolved in less than a second on a desktop PC. (The test set is described in more detail in Section 3.4.)

3.3.2 Search Pruning

A simple geometric observation leads to a pruning technique that can quickly eliminate the majority of images from consideration without incorrectly overlooking any candidates. Given two vectors in $\mathcal{F}$, the angle between them must be greater than or equal to the angle between an orthogonal projection of the vectors in a space of lower dimensionality. This fact forms the basis for pruning. For each of the library images, we cache the projection of the $\mathcal{F}$-vector onto subspaces corresponding to each of the dimensions of $\mathcal{M}$ (i.e., color, texture, and location). The angles between these projections and the corresponding projection of the query image can be quickly computed, and give lower bounds on the angle in the full space (corresponding to upper bounds on the possible similarity) operations compare in time and space to retrieval using color histograms. (If we wish to use multiple **S** matrices, we must either cache projections for each or calculate them online.) Figure 3.8 shows the sorted distances to other images for one typical query, along with the sorted bounds on these distances. The full calculation proceeds in the order indicated by the bounds. Once a sufficient number of close matches are located, all images whose similarity is bounded away from the top scores can be removed from consideration.

Empirically, the number of images that must be fully searched displays an interesting dependence on library size and the number of images returned. Figure 3.9a shows the average number y of images that must be checked in order to be assured of returning a specified number of top images x, for four different size image libraries. Notice that all the curves have a similar shape. If both x and y are normalized by the number of images n, then the curves fall nearly on top of each other. For comparison, the curve of $y = \sqrt{x}$ is also plotted. This suggests that the mean number of images to check varies approximately according a simple formula.

$$\frac{y}{n} \approx \left(\frac{x}{n}\right)^{0.5-\epsilon} \tag{3.11}$$

$$y \approx \left(x^{0.5-\epsilon}\right)\left(n^{0.5+\epsilon}\right) \tag{3.12}$$

The value of ϵ is small and varies somewhat with the specifics of the Stairs implementation. For these data, derived from the finely-quantized version of Stairs, $\epsilon \approx 0.04$. For the basic Stairs implementation it is more like 0.1. In any case, Equation 3.12's sub-linear behavior suggests that the pruning techniques scales well with library size. Interestingly, the dramatic gains achieved with these pruning technique resemble those recently reported for a different image retrieval system with pruning based upon the triangle inequality [11], although the methods themselves are quite different.

3.3.3 Parameter Selection

Section 3.2.3 describes how a full matrix of match coefficients may be created from a few spread parameters. The spread parameters used in the basic Stairs system are p_H, p_S, p_V, p_T, p_X, and p_Y, controlling the degree of match allowed between differing levels of hue, saturation, gray-level value, texture, horizontal coordinates, and vertical coordinates respectively.

The proper settings for these parameters will depend to a certain extent on the task chosen. For example, in order to compare a grayscale image with a library of color images, p_H and p_S should be set to 1. Likewise, if the exposure (or gain) on a query image is known to be badly adjusted, then the value of p_V should also approach 1. The various tests described in Section 3.4 could each have a unique set of optimal parameter settings. However, for a case in which the optimal settings are unknown, a reasonable set of default values can be established. Ideally these would prove to be in a region of parameter space where small changes to any of the parameter values would not result in any significant degradation of performance. Such a set of parameter values exists, as described below.

Parameter values may be tuned by monitoring a suitable performance measure as the individual values are modified in turn. Adopting a hill-climbing approach leads ultimately to a local maximum. Although there is no guarantee that the maximum found will be global, in practice the parameter values selected appear to be reasonable, and provide adequate performance. The classification task described below (Section 3.4) provides a decent test of general image similarity, and therefore performance on this task is chosen as the target performance measure. Since there are two separate data sets for this task, each set serves to select the parameters for the other. The hill-climbing process is controlled by hand, requiring perhaps a dozen steps to converge on reasonable values. As it turns out, the same parameter values are found with each data set and the overall performance shows little sensitivity to small changes in their values. Thus these settings, shown in Table 3.1, appear to provide the generality and stability required for use as default settings.

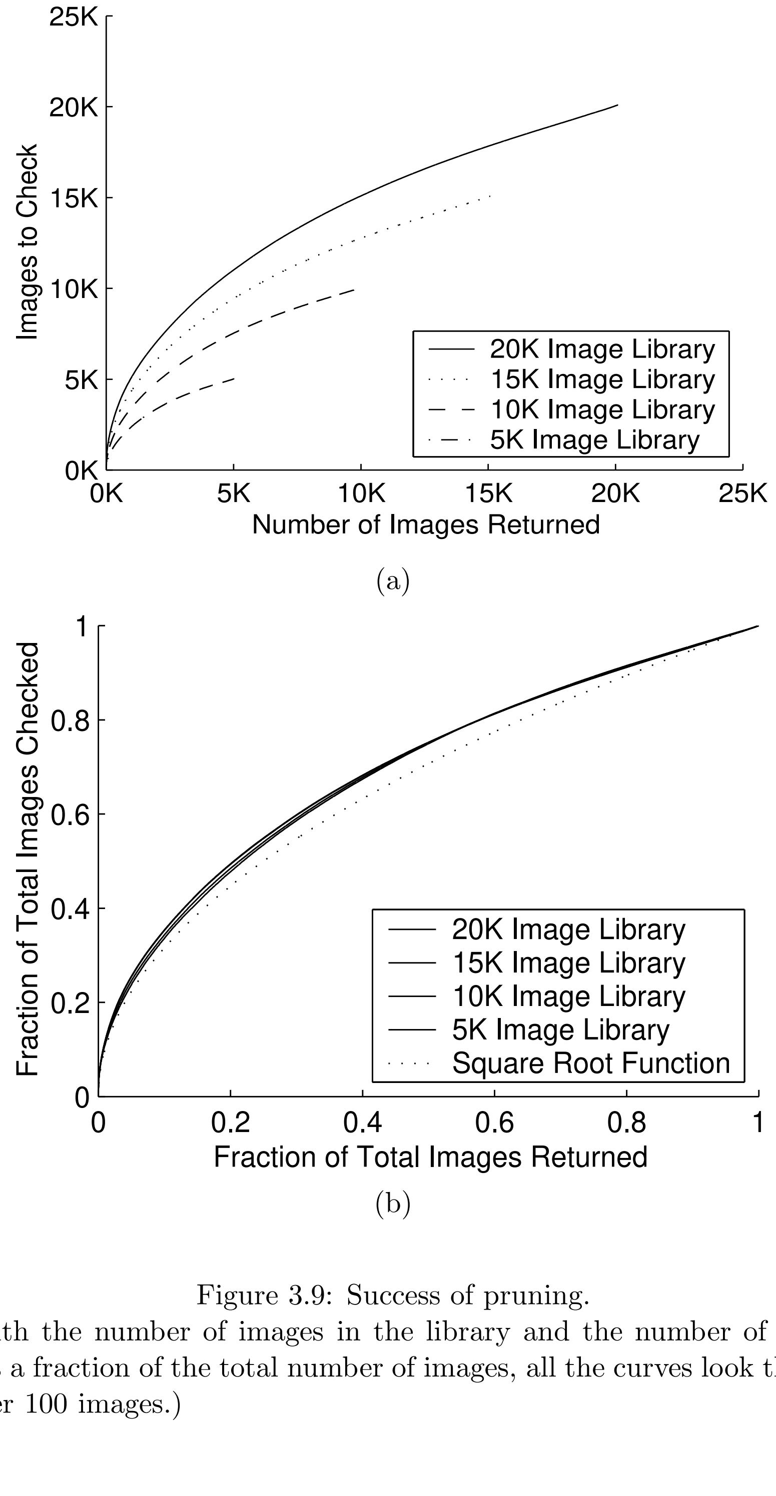


Figure 3.9: Success of pruning.
(a) Variation with the number of images in the library and the number of images returned. (b) Expressed as a fraction of the total number of images, all the curves look the same. (Curves are averaged over 100 images.)

Table 3.1: Default parameter values for the basic Stairs system.

Parameter	Feature	Value
p_H	Hue	0.001
p_S	Saturation	0.01
p_V	Gray Value	0.1
p_T	Texture	0.1
p_X	Horizontal Coordinate	0.01
p_Y	Vertical Coordinate	0.01

3.4 Evaluation

Historically, image retrieval algorithms have proven difficult to evaluate objectively, for a number of reasons. Image libraries are not standardized or centrally available, so that different research groups perform their evaluations using different sets of images. There has been limited sharing of code, so that even when one technique is compared directly against another, the implementation details differ. Finally, there is little agreement over what constitutes a "correct" retrieval, since image similarity involves subjective factors.

Test Images

We address these factors here in several ways. The tests use a library of around 20,100 images purchased from Corel, a commercial source also used by other research groups [20, 61]. The Corel images are stock photographs taken by professional photographers, covering a range of natural, cultural, and geographical subjects. They include landscapes, portraits, and close-up shots of chosen subjects. They were photographed under widely varying lighting conditions by many different photographers. Thus they may plausibly represent the range of images in a general-purpose, professionally photographed image library. The images are available on CD in high-resolution format, but the work described here is conducted with 128×192 thumbnail versions stored in JPEG format. This represents the plausible operating conditions for a working image library; working with thumbnails reduces the number of raw pixels that must be processed.

Baselines

Stairs is compared directly against two other methods: a version of color histograms [74] as a baseline for comparison, and our implementation of the color autocorrelogram technique proposed by Huang, et. al. [41]. The latter is a state-of-the-art technique that has itself undergone extensive testing [40].

In order to allow comparison with other previous work in image retrieval, we apply a classification test that has been previously used to evaluate retrieval methods [8, 38]. However, in order to develop a more complete picture of the performance of Stairs, we also propose several novel evaluations that test its response to specific retrieval conditions. These tests use artificially altered images as queries, with the goal of retrieving the original from the library. Because the tests are algorithmic in nature, they do not depend on subjective judgments of similarity and can be easily replicated by other research groups. Furthermore, because they alter the query images in well-defined ways, they can potentially probe with more precision the strengths and weaknesses of individual algorithms [39].

For the sake of readers unfamiliar with the histogram and correlogram statistics, a short synopsis appears here. The histogram is a vector describing the colors present in an image. To generate a color histogram, we discretize the colors in an image, as illustrated in Figure 3.3. The histogram is simply the vector formed by counting the fraction of the image pixels colored with each color in the palette. The vector thus has length equal to the number of colors in the palette and the sum over all components should equal one.

The correlogram (technically, the banded autocorrelogram [40]) of an image may be viewed as a set of probabilities calculated on the pixels in the image. As with the histogram, it begins with a discretization of the image using a discrete color palette. Additionally, the user must specify a set of distance bands; we use $d \in \{\{0,1\},\{2,3\},\{4,5\},\{6,7\}\}$ pixels as in the original correlogram research [40]. The correlogram is a vector with one component for each combination of color and distance band. For a given color and distance, the value of the component is the probability that a jump of the specified distance starting from a pixel of the specified color will land on another pixel of the specified color.

$$\psi_{c,d} = p\left(C(x',y') = c \,|\, C(x,y) = c \wedge \max\left(|x-x'|,|y-y'|\right) \in d \wedge Valid(x',y')\right) \tag{3.13}$$

Here $C(x,y)$ denotes the color of the pixel located at pixel coordinates x and y. Note that edge effects can affect the probabilities; we choose to define the correlogram to include only jumps that land within the image boundaries, represented by the predicate $Valid(x',y')$.

3.4.1 Classification test

Classification transmutes the question of image similarity into the selection of groups of visually related images. If the categories chosen tend to be visually self-similar and dissimilar to each other, then classification is a conservative but fair test of retrieval. We retrieve related images for a collection of images manually labeled with one of ten categories, and use the class of the most closely related image to predict the class of the unknown query. If the top retrieval turns out to be of the same category as the query image, then the image is considered to be correctly classified; otherwise an error has been made. Note that by focusing on the first image retrieved, the classification task does not necessarily identify all algorithms that may be good at retrieval. For example, it will give low ratings to an algorithm that returns 19 relevant images in the top 20 rankings, if by some chance the irrelevant image is the top-ranked one. In the terms of Section 5.3.1, classification looks for high precision at the lowest of recall values.

Cross Validation

Following standard usage in machine learning, the images given to the system with human-supplied labels are known as the *training set*, while the images to which the system must assign labels are known as the *test set*. Because the size and composition of the training set affects the final result, a classification test will typically be repeated multiple times, splitting the available set of images differently into training and test sets each time. Each repetition with different training and test sets is called a *fold*, and the entire process is called *cross validation*. When used with care it can lead to a more reliable measurement of an algorithm's performance than is possible with a single fold.

One common procedure is called *leave-one-out* cross validation because the training set in each fold is the entire set of images less one. Given n images with labels, the procedure calls for n folds. In each fold, one image forms the test set and the remainder form the training set. Leave-one-out cross validation is useful when the pool of available images is very small, because it uses nearly all the available images for training. However, there is no easy way to determine the variability of the performance numbers derived with this technique.

A more recently developed procedure [25] is called 5×2 cross validation, or *5x2cv* for short. Given n images with labels, the procedure splits them arbitrarily into two sets with $n/2$ images each. These form the training and testing sets for two folds, swapping roles after the first. The entire two-fold process is repeated five times for comparison, using different splits each time. The mean score over all five repetitions can be compared with the mean score for other algorithms, and the variance between repetitions gives an estimate of the reliability of these numbers. Scores from a 5x2cv analysis are generally lower than those found with leave-one-out cross validation, because the available training set is roughly one half the size.

Coarse Color Quantization Results

Results of for the classification test using leave-one-out cross validation are shown in Table 3.2 by category, for two separate test sets; the first has been used by other groups [8, 38] and the second includes additional categories in the same vein. As expected, Stairs and correlograms perform better than histograms in terms of the overall mean accuracy. Of the 28 individual categories, Stairs does best in twelve (counting one tie), histograms in two, and correlograms in 15. Interestingly, the relative performance varies noticeably by category. This suggests a weakness of the classification task as an evaluative technique: The particular image categories chosen can greatly affect the apparent performance of the algorithms being evaluated. On the other hand, it also indicates that each algorithm has independent strengths and weaknesses that might be exploited.

Stairs' flexible architecture allows additional features to be easily added if such an action seems necessary or desirable. The basic Stairs algorithm uses purely local information in its token descriptions. To see whether adding some regional information would improve performance on the classification test, we include a fourth feature in the token description, representing the fraction of neighboring tokens that have the same color and texture. Intuitively, this carries information about whether the token is within a large homogeneous region or is an isolated feature. The results, listed in Table 3.2 as Stairs (+), show a modest improvement over the basic algorithm. The ease with which additional features may be added for specific tasks is one of the strengths of Stairs' flexible architecture.

Fine Color Quantization Results

As described in Section 3.1.1, Stairs was originally developed using a fairly coarse color quantization scheme. For comparative purposes, histograms and correlograms have been tested using the same setup, which consists of only 28 color bins. In a sense, the sensitivity embodied in the color quantization is one of the parameters of the system. Too many color bins increases the amount of computation required and can potentially hurt the performance of algorithms like histograms and correlograms. On the other hand, using too few color bins can also hurt performance. Therefore we present here the results of experiments on various algorithms using a 128-bin color quantization, derived as the 28-bin quantization was from a Voronoi decomposition of the HSV color cone. (In the case of Stairs, this section uses the finely-quantized Stairs system described in Section 3.1.4)

Table 3.3 gives the classification results by category using the finer 128-bin color quantization and a 5×2 cross validation setup. The correlogram and histogram techniques are both helped by the change, suggesting that at 28 bins per color they are not operating at their peak performance. A version of Stairs using 128 color bins, five texture bins, and 25 modified location bins also shows some improvement, but not as much. The result is that the overall correlogram result appears clearly better than the other two, while the histogram score also improves relative to Stairs. On the other hand, variations persist in relative performance between categories. Stairs has the best score in only three individual categories, histograms in none, with correlograms taking the rest. These results show a decided advantage to the correlogram method, although it does not dominate in every category.

Alternate Forms of Evaluation

Unfortunately, evaluation techniques like the classification task are somewhat *ad hoc*, because they rely on the use of a specific image set and subjective determination of the correct response to a set of test queries [8, 13, 32, 38]. As a result, evaluations reported by different researchers are often incomparable, and the relative merits of many proposed retrieval algorithms are unknown. Although using the same image collection standardizes the situation a bit, for a variety of reasons this is not always possible.

In fairness, real difficulties hamper the development of consistent evaluation methodologies. A large, standardized, and universally available collection of natural photographic images has not yet developed. The image collections published by Corel form the current *de facto* standard, and are cited in many works [8, 13, 38]. Unfortunately, there are over 60,000 Corel images in all, and different researchers use different subsets of this collection. Many lack access to the entire set of images, which costs around $3000 to purchase in full. Thus the image sets used in evaluation can be expected to vary in both size and content for at least the near future.

Furthermore, even given a standard set of images, there is no agreement on what constitutes a proper set of queries, nor the corresponding correct responses. Researchers in text retrieval have solved this problem by establishing conferences where all work is tested using a common, shared evaluation package [77]. Unfortunately, such a system may be more difficult to set up for image retrieval, where the specification of correct answers actually constitutes a definition of the problem to a certain extent. For example, image retrieval potentially encompasses similarity based upon (among other things) thematic similarity, the appearance of specific objects or backgrounds, the subtle differences between a series of medical images, and vague notions of general visual similarity (see Figure 3.10). Each of these criteria implies a different answer to the question "What is most similar to image I?" and each has potentially useful applications. To choose a single set of universal queries and correct responses would necessarily favor some approaches and penalize others unfairly.

To address these issues, we propose a set of evaluations using query images that are algorithmically altered in known ways. Using artificially altered data may seem like an unusual step in evaluating a system that is designed to work with real images, but such methods have been successfully used to gain insight in other fields such as machine learning [1]. Note that the altered images are not synthetic, simplified constructions designed to make the retrieval algorithm's task easier; they are constructed from natural images and share their complexity, while differing from them in known ways.

In our proposal, images from the test set are modified in a specific manner and used as queries. The goal is to retrieve the originals from the test set. Thus the images in the retrieval library are all natural; only the queries are artificially altered. Although the specific images used will necessarily affect the result somewhat, relying on artificial queries means that the

(a) (b) (c)

Figure 3.10: Similarity depends on context.
Given (a) an image, different (b-c) images might be considered most similar depending on the situation.

differences between the query and the target are consistent across test sets. Furthermore, while a specific test may favor one algorithm over another, knowing the nature of the altered query makes the biases transparent. For example, color histograms might be expected to do poorly on queries where the color balance has been altered, but should do well on queries where image elements are moved around.

3.4.2 Altered-Image Query Tests

Suppose we have a test set $S = \{I_1, I_2, ..., I_n\}$ consisting of n images, and we wish to use these images to compare the performance of several retrieval algorithms. Assume further that the retrieval algorithms all operate in a query-image paradigm, i.e., given a query image Q which may or may not be an element of S, they produce an ordinal ranking on the test images. By convention, lower ranks are better; i.e., $rank(I_j) < rank(I_k)$ implies that I_j is more relevant than I_k. Typically algorithms produce their rankings by sorting on the distance from the query image according to some metric, but other mechanisms are possible.

Historically, algorithms have been evaluated on this task by comparing the ranks they produce to subjectively defined ground truth targets for each query. Human judges designate a target set $R \subset S$ for each query image, using standards that often go unreported. Algorithms are scored based upon the mean rank assigned to the target images, or perhaps the number of target images with rank below a certain threshold. Typically, published results consist of averages over many applications of this basic technique.

While results from such tests can provide valuable information about retrieval performance, they can implicitly incorporate biases that unfairly favor one algorithm over another. The criteria used to choose the target set R are usually difficult to specify or even express, muddying the interpretation of the results. They are also difficult to duplicate, preventing other research groups from conducting an equivalent test using a different set of images. Finally, they are arbitrary to a certain degree, since there may be contexts in which the images in R are not the most relevant images to the query Q. (As an example, given the picture of a flowerpot on a shelf in Figure 3.10, should a system retrieve pictures of windowsillsthe same shelf, or of flowerpots?) A good test should make the relationship between Q and R transparent, and hence also the context in which the algorithms are compared.

One solution is to use Q and R with a known, quantifiable relationship. We propose to do this by picking an image $I_j \in S$ at random, setting $R = \{I_j\}$ and $Q = f(I_j)$ where f is an easily computable transformation of the original image. To make the test effective, f is chosen to produce some change to the image that leaves a clear relation to the original. By using a variety of choices for f, we can see how different algorithms respond to diverse sorts of image variations, alone or in combination. If the choices are made carefully, the variations will correspond to those that might arise in a real application. The remainder of this section describes three choices of f that probe at different aspects of retrieval performance in action.

The Crop Test

Image databases often contain multiple views of the same scene or subject taken from different distances. A valid expectation for an image retrieval system would be to retrieve all views of the subject given a single view as the query. The *Crop* test attempts to simulate this task in a controlled manner. Query images are created by trimming a margin off the edges of the target image, resulting in a "close-up" of the center section. Because photographs are often centered on a subject, this will often be a closeup of the subject of the photo, but in some cases it may merely be a detailed view of some part of a scene. In either case it is reasonable to expect the original image to be retrieved as a related image.

The function f_{Crop} takes an additional parameter k, which is the percentage of the original image area to retain. Empirically, we have found $k = 50\%$ to be a good value, resulting in query images that are easily recognizable to humans and moderately challenging for machine algorithms. We crop the image such that the area remaining is centered and has the same aspect ratio as the original. (The area remaining may be slightly more than $k\%$ of the original, because any fractional pixels left after cropping are rounded out to whole ones.)

If the image I_j is represented as an $m_j \times n_j$ array of pixels, then borrowing array index notation from Matlab, the *Crop-k* transformation function f_{Crop} is

$$f_{Crop}(I_j) = I_j(1 + \delta_{m_j} : m_j - \delta_{m_j}, 1 + \delta_{n_j} : n_j - \delta_{n_j}) \tag{3.14}$$

where the crop margins are $\delta_{m_j} = \left\lfloor \frac{1}{2} m_j \cdot \sqrt{1 - k/100} \right\rfloor$ and $\delta_{n_j} = \left\lfloor \frac{1}{2} n_j \cdot \sqrt{1 - k/100} \right\rfloor$. Figure 3.11(b) shows an image that has been subjected to the *Crop-50* transformation.

The Jumble Test

In many applications, the identity of objects in a photograph is much more important than their arrangement. For example, pictures taken from different angles in the same room will show the same objects in different locations and orientations. (Although such photos would usually be judged as relevant images, this may not always be the case. For example, we may wish to search for all shots taken from a particular camera angle.) Nevertheless, in many cases a retrieval algorithm should ignore the details of object placement. The *Jumble* test simulates this condition, albeit imperfectly, by splitting the image into rectangular sections and exchanging them randomly. Although this procedure leads to artificial boundaries in the image, it can nevertheless prove an effective test. If the sections moved are large enough, recognizable object features will be preserved within them and moved *en masse* to a new location.

The function f_{Jumble} takes two additional parameters representing the number of divisions to make along each axis. Setting both to four (*Jumble-4x4*) divides the image into sixteen rectangular areas, which are shuffled randomly. This test appears to be more difficult for human observers than the others, but is still solvable with some effort. Anecdotally, it appears that people may resort to explicit feature matching between the jumbled image and the original; many machine algorithms do not explicitly look for such matches.

The transformation f_{Jumble} can be defined on a per-pixel basis:

$$f_{Jumble}(I_j)(r,s) = I_j\left(\left(r \textbf{ mod } \left\lfloor\frac{m_j}{k}\right\rfloor\right) + B_1(r,s), \left(s \textbf{ mod } \left\lfloor\frac{n_j}{l}\right\rfloor\right) + B_2(r,s)\right) \tag{3.15}$$

where

$$B_1(r,s) = \left\lfloor\frac{m_j}{k}\right\rfloor \cdot \left[J\left(r \textbf{ mod } \left\lfloor\frac{m_j}{k}\right\rfloor + k \cdot \left(s \textbf{ mod } \left\lfloor\frac{n_j}{l}\right\rfloor\right)\right) \textbf{ mod } k\right] \tag{3.16}$$

$$B_2(r,s) = \left\lfloor\frac{n_j}{l}\right\rfloor \cdot \left\lfloor\frac{1}{k}J\left(r \textbf{ mod } \left\lfloor\frac{m_j}{k}\right\rfloor + k \cdot \left(s \textbf{ mod } \left\lfloor\frac{n_j}{k}\right\rfloor\right)\right)\right\rfloor \tag{3.17}$$

and J is a permutation of $(1, 2, ..., kl)$. Figure 3.11(c) shows an image that has been subjected to the *Jumble-4x4* transformation.

The Low-Con and Gain Tests

Photography is subject to different lighting conditions, variations in development and scanning, and other processes that can make two otherwise identical pictures appear different. In almost all cases we wish to ignore differences in lighting, camera gain, contrast, and color balance for purposes of image retrieval. The *Low-Con* and *Gain* tests measure sensitivity to these factors. *Low-Con* decreases the image contrast, while *Gain* simulates a picture taken through a camera with a different gain. In general, human viewers are quite insensitive to these sorts of changes, but many color-based algorithms find them challenging.

Both $f_{Low-Con}$ and f_{Gain} take an additional parameter. For *Low-Con*, this is the percentage of the original color range that is used in the transformed image. For example, if the original RGB color values are scaled between zero and one, the *Low-Con-80* test will rescale them to run from 0.1 to 0.9. Similarly, *Gain-k* takes RGB values scaled between zero and one, and raises them to the kth power, for typical k values ranging from 0.8 to 1.2. The *Low-Con* test has the advantage of simplicity, but for algorithms that perform a crude color renormalization before retrieval, the *Gain* test may be more appropriate.

Assuming the RGB values stored in I_j are scaled between zero and one, the transformation functions may be defined as follows:

$$f_{Low-Con}(I_j)(r,s) = \frac{(1-k)}{2} + k \cdot I_j(r,s) \tag{3.18}$$

$$f_{Gain}(I_j)(r,s) = [I_j(r,s)]^k \tag{3.19}$$

Typical values of k are 0.2 for *Low-Con* and 0.8 to 1.2 for *Gain*. Figure 3.11(d) shows an image that has been subjected to the *Low-Con-80* transformation.

Other Tests

Other sorts of artificial queries may be devised as needed to test specific aspects of retrieval performance. For example, a *Blur* test that smooths the original image would test retrieval

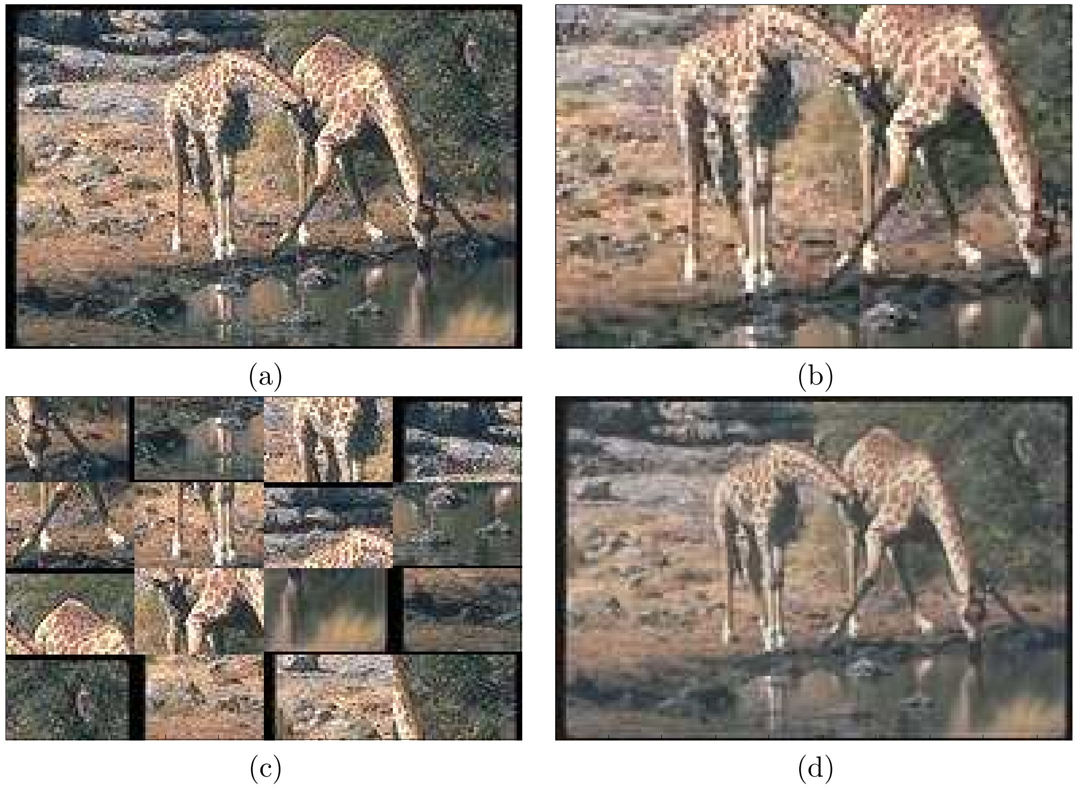


Figure 3.11: Examples of each type of artificial query.
(a) Original. (b) Crop-50. (c) Jumble-4x4. (d) Low-Con-80.

from a low-resolution or out-of-focus original. A *Gray* test could determine whether color images can be retrieved from grayscale queries. The *Low-Con* and *Gain* tests may be applied to individual color channels separately. One can imagine system designers using a large library of altered-image tests to select the algorithm that is best for a particular specialized retrieval task.

3.4.3 Validating Artificial Image Queries

If altered-image queries are to serve as a yardstick for comparing retrieval algorithms, then it is important to understand their behavior under different conditions. This section explores how results of altered-image query tests change as experimental parameters are varied. We begin with a look at a single retrieval algorithm, and proceed to a look at comparisons between three different ones.

Figure 3.12 shows a set of typical results for an altered-image task. The results were generated for a simple implementation of color histograms on the *Crop* test, at varying parameter settings. In order to clearly show the range of performance, the results are presented in sorted order, with the most successful queries on the left and the least successful on the right. Thus the x axis gives the percentile of the query, and the y axis gives the rank at that percentile. The curves are skewed, with a large region of slowly degrading performance terminating in a sharp tail. In other words, most target images are retrieved at a relatively low rank (which is good), but on a small number of queries the algorithm does much worse. The curves can be concisely summarized by two numbers. The median rank indicates the level of the majority of queries, and thus the performance on a typical case. The mean rank indicates the size of the tail, and thus the performance on the most difficult queries. As the difficulty of the task increases, both numbers generally rise.

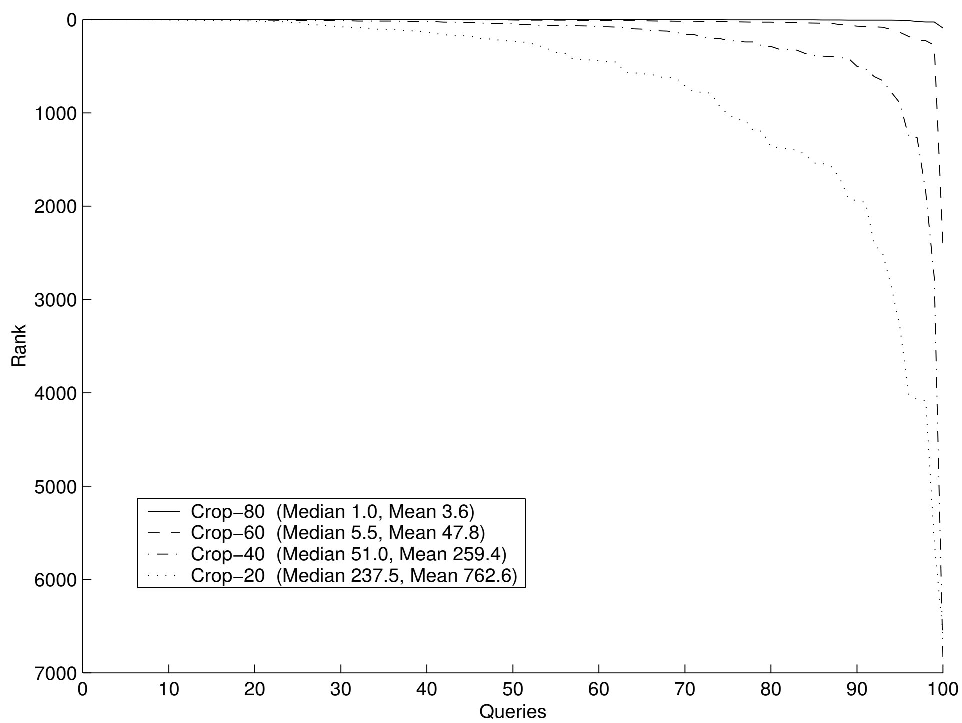


Figure 3.12: Results of histogram method on Crop task at varying difficulty levels. (Queries are sorted by increasing rank.)

Test Set Composition

One concern when comparing results reported in different places is how much difference the image test set makes. Clearly the test set has some effect: if for example, it contains many shots of the same subject from different distances, then the *Crop* test will be more difficult because many distractors will compete with the target. If on the other hand it contains many dissimilar images, the test will be easier. Most image sets fall short of these two extremes, and the average variation in score is small even when using different test sets.

How much variability should be expected from performing the same test on different image sets? To answer this question, we formed three entirely disjoint test sets of 6000 images apiece and ran the *Crop-50* test. The results, shown in Table 3.4, display a marked similarity given that the three sets have no images in common. The standard deviation of both the mean and median are around 20% of their values. A comparison with the results presented in Table 3.5 below shows that this variation is much smaller than the differences that can appear between different algorithms. This key observation suggests that results reported on different image sets may be compared with some hope of drawing accurate conclusions. Naturally it is best if the same images are used; failing that it probably helps if all of the images are drawn from a single source. (The results reported here all use images from the Corel collection.)

Test Set Size

The size of the test set may also make a difference. Figure 3.13 plots the mean and median ranks for the histogram algorithm as extra images are added to the test set, while keeping the query set constant. The numbers rise linearly with the test set size, suggesting that when comparing different results one should normalize by the total number of images.

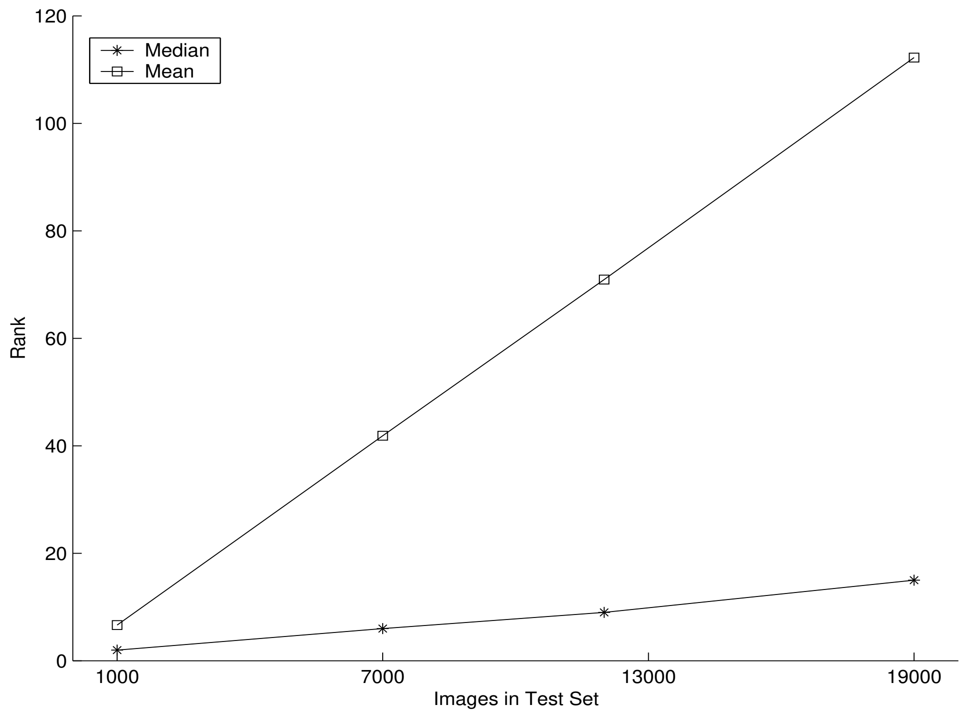


Figure 3.13: Linear dependence of mean and median ranks on test set size for color histograms on the Crop-50 task.

Different Queries

The choice of the query images may also make a difference, if the test is not repeated for every image in the test set. With a large set of images, it may be unnecessary to test all the images since the dominant trends become apparent after only a small fraction of the total has been tested. Figure 3.14 shows the variation in the results as increasing numbers of query images are tested. The numbers show some variability at small numbers of queries, but get steadier as results from more queries are averaged together. On the whole, while increasing the number of queries will give a more accurate result, a surprisingly accurate picture arises after testing fewer than 10% of the total number of images.

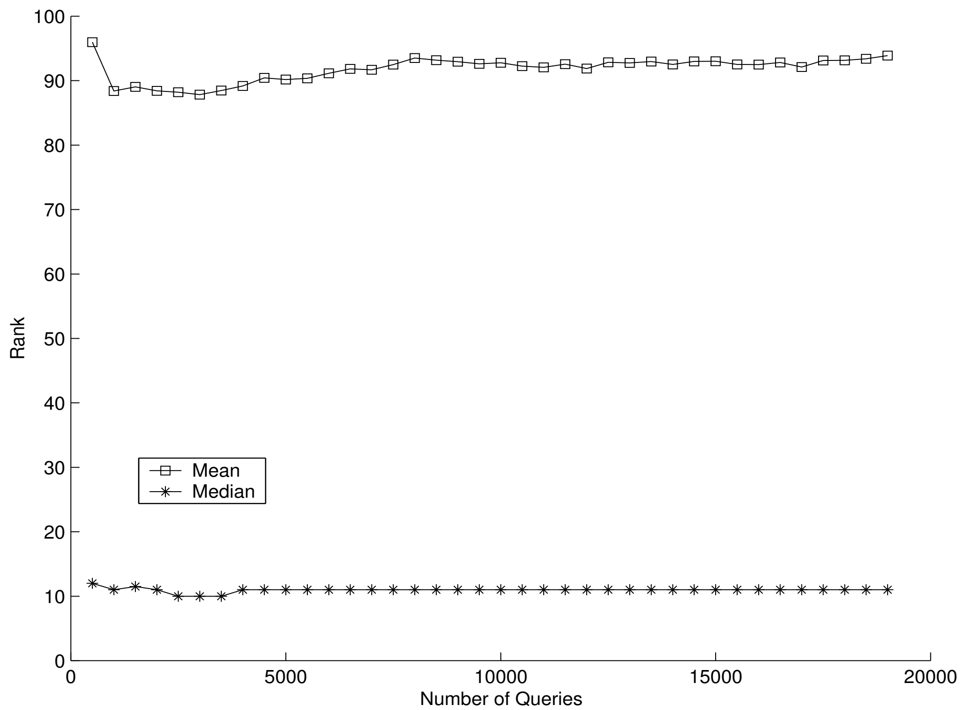


Figure 3.14: Change in score with size of query set for color histograms on the Crop-50 task. (19,000 images total).

3.4.4 A Comparative Test

This section presents a comparison between three image-retrieval algorithms: color histograms [74], color correlograms [41], and the Stairs algorithm. Each is evaluated using as queries a randomly selected 1000-image subset of our collection of 19,000 images. Each algorithm is tested on the *Crop-50*, *Jumble-4x4*, and *Low-Con-80* tests. (The same query set is used for each algorithm on any given test.)

The results are summarized in Table 3.5, which gives the median and mean ranks for each test. Analysis of the results provides more insight into individual strengths and weaknesses than a traditional test. We see that histograms do the worst on every test except *Jumble*, where they by definition have a perfect score. They perform particularly badly on *Low-Con*, because that test directly affects the distribution of color in the histogram. (We do not use any sort of color scaling scheme, but the *Gain* test would reveal a similar weakness in the histogram algorithm, even with scaling.) Thus the results spell out the strengths and weaknesses of color histograms.

Color correlograms also depend on the distribution of color, though to a lesser extent than histograms, and thus they also show the lowest score in the *Low-Con* test. On the other two tasks, they do quite well. Stairs, by contrast, does well on *Low-Con* but not as well on *Jumble*. This is because it has mechanisms that handle small changes in color, but relies by default on an explicit representation of the location of features in the image. On the other hand, Stairs allows the user to tune parameters relating to how much color, texture, and spatial features are valued. The bottom row of Table 3.5 indicates that these parameters can be adjusted to make Stairs successful in any of the environments tested.

Introspection might perhaps have led after a while to the insights into each algorithm described above. On the other hand, it would have been difficult to set up a traditional query-target test using only natural images that would have demonstrated the algorithms' respective strengths and weaknesses so clearly. This illustrates the value of using a diverse repertoire of evaluative tools.

3.4.5 Analysis of Altered-image Queries

This dissertation does not attempt to prescribe comprehensive tests for the evaluation of newly proposed retrieval algorithms or the comparison of existing ones. Rather, it proposes a new class of tests in the hope that they will be adopted and incorporated into the standard practice of the field. Research in image retrieval can only benefit from new evaluative tools, and perhaps their availability will spur new developments. In time, if extensive comparative testing becomes the norm, the field will be ripe for a conference along the lines of TREC [77]. However, a proposal of this sort is not the purpose of this thesis.

As the image retrieval field develops, the specific tests proposed here may become obsolete, proving too easy for advanced retrieval techniques. In particular, as work moves away from similarity of full images and towards retrieval based upon objects within images, new tests will be required. In such a future, the altered-image paradigm will remain viable but may be applied in new ways. For example, appropriately altered query objects may be placed within unrelated scenes to create an artificial target. Other tests may also be devised as the need arises.

Altered-image queries should not completely replace traditional testing methods for image retrieval, but they deserve a place alongside the traditional techniques. The different forms of evaluation serve complementary purposes. Traditional approaches can be seen as field testing under a simulation of real conditions. By contrast, altered-image query testing serves as a diagnostic tool for comparing different retrieval algorithms under more controlled conditions, and for identifying areas where a particular algorithm may be underachieving. Hopefully, the conscientious use of both forms of testing will lead over time to a greater understanding of the issues involved, and ultimately to improved algorithms. The development of more flexible and consistent evaluation tools can only advance that goal.

Table 3.2: Overall classification results by category for selected image retrieval algorithms.

Category	Stairs	Hist.	Corr.	Stairs (+)
Air shows	**68**	57	59	65
Bald Eagles	69	55	**70**	**70**
Brown Bears	**46**	35	35	43
Mountains	72	76	**82**	**78**
Cheetahs, etc.	65	62	**76**	**66**
Deserts	54	47	**57**	52
Elephants	**81**	**81**	76	**85**
Fields	**48**	46	43	**54**
Night Scenes	68	68	**70**	**71**
Polar Bears	60	49	**66**	54
Sunsets	64	68	**75**	64
Tigers	99	97	**100**	99
Overall	67.2	63.4	**68.6**	**68.3**
Candy	68	59	**80**	**69**
Cars	**89**	57	63	**90**
Caves	42	34	**48**	42
Churches	**44**	33	37	39
Divers	56	71	**75**	**61**
Doors	**57**	39	52	**64**
Gardens	60	**72**	62	**61**
Glaciers	**78**	51	74	74
Hawks	58	60	**69**	57
MVs	**51**	33	42	**57**
Models	**75**	41	57	66
People	18	19	**20**	**25**
Ruins	**50**	40	48	**53**
Skiing	59	52	**65**	56
Stained Glass	70	74	**84**	**76**
Sunrises	**63**	52	60	**68**
Overall	**58.6**	49.2	58.5	**59.9**

Table 3.3: Overall classification results by category for algorithms with fine color quantization

Category	Stairs	Hist.	Corr.
Air shows	72.0	66.6	**74.0**
Bald Eagles	75.6	74.6	**81.4**
Brown Bears	26.8	27.8	**33.0**
Mountains	72.0	86.0	**81.6**
Cheetahs, etc.	64.4	72.2	**86.0**
Deserts	58.2	56.0	**61.8**
Elephants	**77.2**	74.2	76.2
Fields	41.4	42.0	**49.8**
Night Scenes	68.8	77.6	**82.2**
Polar Bears	16.2	20.0	**24.8**
Sunsets	75.0	66.4	**77.0**
Tigers	79.8	84.2	**93.8**
Mean (by class)	60.6	62.3	68.5
Mean (by instance)	66.1 ± 0.5	68.0 ± 0.3	74.7 ± 0.6
Candy	65.0	65.0	**71.6**
Cars	**76.6**	71.6	71.0
Caves	55.2	57.0	**60.8**
Churches	40.2	33.8	**45.4**
Divers	86.0	86.4	**86.4**
Doors	50.4	46.8	**50.2**
Gardens	78.2	91.0	**92.0**
Glaciers	74.6	68.0	**80.8**
Hawks	52.4	63.6	**74.6**
MVs	56.0	59.0	**75.2**
Models	52.2	48.8	**52.6**
People	12.0	26.4	**22.2**
Ruins	46.6	59.4	**68.8**
Skiing	56.6	70.4	**79.4**
Stained Glass	76.6	85.8	**88.2**
Sunrises	**59.2**	48.4	57.4
Mean	58.6 ± 0.5	61.3 ± 0.3	67.3 ± 1.2

Table 3.4: Results for three disjoint sets, using color histograms on the Crop-50 task. Compare with the variation visible in Table 3.5.

	Set 1	Set 2	Set 3	Mean	Dev.
Median Rank	5	5	7	5.7	1.2
Mean Rank	29.6	33.9	45.5	36.3	8.2

Table 3.5: Target rank results in artificial-query tests for three image retrieval algorithms.

		Crop	*Jumble*	*Low-Con*
Histograms	median	18	1	86.5
	mean	126.6	1	350.3
Correlograms	median	1	1	5
	mean	12.4	2.0	83.6
Stairs Default	median	1	26	1
	mean	38.9	205.2	18.2
Stairs Tuned	median	1	1	1
	mean	17.0	1.2	22.6

Chapter 4

Relational Stairs

The area-matching approach holds that two images should be considered similar the more that they contain equal areas of similar material. It leaves open the question of what definition of similarity to apply. The Stairs framework captures many aspects of similarity involving local features of image tokens, including color, texture, and absolute position in the image. As demonstrated in Chapter 3, this results in a viable image-retrieval system that can successfully perform a wide range of retrieval and classification tasks.

Nevertheless, the interpretation of a particular piece of the image cannot be divorced from the environment in which it appears. For example, a yellow patch surrounded by green could be a flower in a field. If it is surrounded by more yellow, on the other hand, it is more likely to be part of a larger yellow object such as a toy rubber duck. Thus in terms of the area-matching approach, it might make sense to use a definition of similarity that takes into account the context in which each piece of the image appears. Up to this point, the Stairs engine has not included the regional context of a token in its definition of similarity. This chapter explores various ways of taking such factors into account. Although built on the Stairs framework, the algorithms described here differ significantly in their structure and intent. When necessary to distinguish them from the standard Stairs techniques described in Chapter 3, we will refer to these algorithms as *Relational Stairs*, or R-Stairs.

4.1 Context as Pairwise Relations

One way to incorporate contextual information is to look at all of the pairwise relations in an image. From this point of view, the context of a particular patch in the image is the set of all pairwise relations between it and the other patches in the image. The basic image token becomes the pair of patches, rather than the singular patch. Of this pair, the first patch is considered primary, meaning that it is the patch under description. The second patch is considered secondary, meaning that it is included as an indication of the context of the primary patch. Typically, given any two patches in the image, there will be two tokens corresponding to the pair, one with each member as primary.

Given that Stairs identifies some 500 patches in a typical image, incorporating all $500^2 = 250,000$ pairs as image tokens would place an untenable strain on the system architecture. Therefore, we have implemented two different varieties of pair-based tokens in Stairs. Each variety simplifies the problem in one way or another to accommodate the burden of the quadratic rise in the number of basic image tokens.

4.1.1 Method I: Local Pairs

One way to limit the number of pairs that must be considered is to look just at neighboring pairs. Intuition suggests that the area directly around an image patch is most likely to affect its interpretation. By contrast, regions of an image that are farther away are less likely to alter the meaning. Thus by examining only the local context (i.e., pairs formed with nearby neighbors) of a patch, perhaps one can still extract most of the information that would be available from an examination of all pairs. This hypothesis forms the basis of Method I, which looks only at neighboring pairs.

Algorithm

R-Stairs I defines the context of an image patch to be the patch itself plus all 4-connected patches. For each patch in the image, the algorithm creates a small set of tokens, one for each patch in the primary patch's context. (Note that this includes one token where the primary patch is paired with itself. Given that the segmentation into patches can be somewhat arbitrary, it is important to include self-pairs. In a similar image, the same area could potentially be divided into two patches.) On average, each patch touches about six neighbors, so this algorithm produces an image representation with about 3500 tokens from an image of 500 simple patches. This increase is quite manageable.

The following scheme assigns weights to each token: the group of tokens sharing a particular primary token have a total weight equal to the area of the patch. Within this group, weight is distributed according to the relative areas of the secondary patches. This corresponds with the intuition that the collection of tokens sharing a primary together describe the context of that primary.

In addition to having sevenfold more patches, the algorithm must use a more complex description for each patch, in order to take into account the properties of both members of the pair. R-Stairs I doubles up the color and texture features, so that each member of the pair is described as in the standard Stairs architecture. The single location feature records the position of the primary (central) patch in the pair. A directional feature indicates the relative position of the second member of the pair. This directional feature is discretized into four bins aligned with the four cardinal directions, plus a special fifth bin indicating no direction (for use in the case of "self-pairs"). Note that any two neighboring patches will appear as two separate image tokens, one with each patch as primary, and the direction feature diametrically opposed.

In total R-Stairs I uses an $\mathcal{F}$-space with $28^2 \times 3^2 \times 25 \times 5 = 882,000$ dimensions. In spite of this increase in size, the pruning techniques and other optimizations described in Chapter 3 ensure that most queries on the 20,100 image library can be processed in under two seconds on a 266 MHz Pentium-II desktop PC. Thus the increase in complexity does not lead to an overburdening increase in running time.

4.1.2 Method II: Macro-Patch Pairs

A second way to limit the total number of pairs is to decrease the number of patches that must be considered. This may be achieved through some sort of selection process, or by grouping patches into larger regions. We choose to explore the latter course by applying the work on segmentation developed in Chapter 2.

Algorithm

R-Stairs II defines the context of the patch to be the entire image. In order to reduce the total number of pairs to be considered, it applies a much coarser segmentation to generate the patches. Specifically, the mid-level segmentation described in Chapter 2 forms the basis of the representation. On average, this segmentation produces about 50 segments per image, although the variance is high and some images have as few as a dozen while others more than 100. Thus a typical image might comprise a few thousand pairs of patches.

At first, a simple scheme was used to assign weights to tokens: the product of the areas of the two contributing patches gives the weight. However, this method tends to disproportionately increase the relative weight associated with the largest patches in the final image representation. Thus images containing a large patch of something (say sky) tend to match best to other images containing large amounts of sky, regardless of the contents of the rest of the image. Furthermore, dimensional analysis suggests that the quantities used as weights, with units of $\langle area \rangle^2$, have no intuitive meaning. For these reasons we also look at using weights equal to the geometric mean of the areas of the two contributing patches. This decision is also somewhat difficult to motivate intuitively, but avoids the imbalanced weighting associated with using the product of the areas. In Section 4.3, results for the first weighting scheme appear labeled R-Stairs IIa, while those for the second appear as R-Stairs IIb.

The token description chosen for R-Stairs II focuses more on the primary member of the pair, with the secondary member described entirely in terms of the first. The intent of this change is to better capture their differing roles as the piece under description and the environment in which it resides. Each token includes the standard color, texture, and location features for the primary. Additionally, it includes features describing the secondary's hue, saturation, value, texture, and position relative to the primary.

The specifics of the relative quantizations follow. Method II quantizes relative hue into four bins: one bin within $\pi/6$ of the primary hue (assuming that hue is defined on a circle of circumference 2π), two bins within $\pi/2$ to the left and right, and one bin for everything else. Relative saturation divides into three bins, with the divisions at ± 0.5 from the primary value. Relative value also divides into three bins, with the divisions at ± 0.25 from the primary value. Relative texture places the divisions at ± 4 on the slope-related measure of texture described in Section 3.1.2. Relative position is a more complicated matter, incorporating both direction and distance. Figure 4.1 shows the divisions into relative bins.

In total R-Stairs II uses an $\mathcal{F}$-space with $28 \times 3 \times 25 \times 4 \times 3 \times 3 \times 3 \times 13 = 1,061,424$ dimensions. This is slightly larger than the representation used in Method I, but remains roughly the same order of magnitude. As before, running times are acceptable.

4.2 Context as Extra Features

A second way to incorporate context is to encode it within extra features added to the single token describing each patch. The advantage of this scheme lies in its simplicity: the number of tokens does not increase quadratically, and the weight of each token remains clearly tied to the area of the corresponding patch in the image. The drawback is more subtle: boosting the number of token features causes the size of $\mathcal{F}$-space to balloon dramatically. Although a vector in $\mathcal{F}$-space need never be instantiated in memory, and thus there is no hard limit on its size, there is still a cost incurred for using indirect representations. In practice, our Matlab implementation limits the number of dimensions allowed in $\mathcal{F}$-space to 2^{31}, as Matlab provides 31 bits of storage for row indices in its sparse matrix representation.

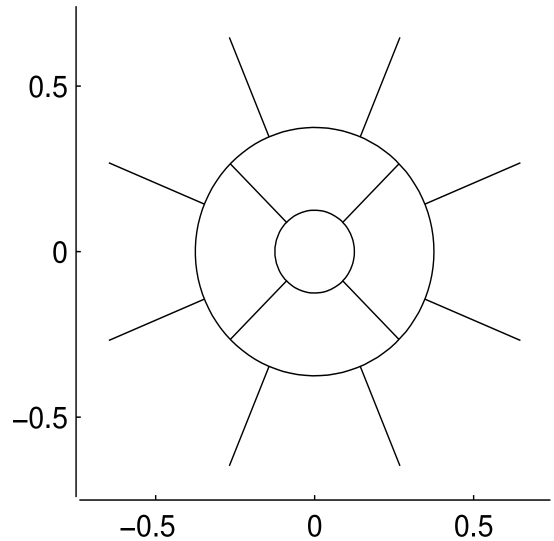


Figure 4.1: Quantization of relative position into bins.
Unit length is the geometric mean of the image dimensions.

4.2.1 Method III: Color Context Features

R-Stairs III defines the context of a patch to include the patch itself plus all of its 4-connected neighbors. In order to limit the size of $\mathcal{F}$-space somewhat, the description of the context comprises nine binary features indicating the colors present in the context. Specifically, the HSV color space is coarsely divided into six color bins distributed around the outside of the cone, and 3 grayscale bins along the central axis. Bins that are occupied by one or more patches within the local context have their corresponding feature set to 1, while unoccupied bins have the feature set to 0. The collection of all nine feature values may be alternately viewed as a single 9-bit binary number describing the local context. For example, in the middle of the sky every patch in the local context may be blue, corresponding to a 9-bit descriptor that has only one bit set. If a piece of a tree intrudes into the context then the green bit may also be set. This corresponds to a different 9-bit descriptor that differs from the first in exactly one bit (the green feature).

In total R-Stairs III uses an $\mathcal{F}$-space with $28 \times 3 \times 25 \times 2^9 = 387,072$ dimensions. This is actually smaller than the representations used in Methods I and II. However, significantly expanding it would result in an unwieldy size. For example, using the 28-bin color quantization would give $\mathcal{F}$-space $28 \times 3 \times 25 \times 2^2 8 \approx 5.6 \times 10^{11}$ dimensions, too many to conveniently store in Matlab's sparse matrix representation. We claim that the nine-bin quantization should be sufficient to show whether this sort of approach holds any promise.

4.2.2 Method IV: Context Similarity Features

R-Stairs IV defines the context of a patch as a set of concentric rings. Specifically, it uses four such rings, representing a range from near context to far. Succeeding rings double in radius, starting with a radius of 0.05 for the smallest (given a unit length equal to the geometric mean of the sides of the image). Thus the largest circle encompasses roughly one-half the area of the image.

A single feature associated with each ring records the aggregate similarity between the primary token and all of the tokens located in the annulus between the ring and its inner neighbor. We calculate this similarity using a spread matrix as described in Section 3.2.3; the spread parameters are taken from the tuned values used in one of the experiments in Chapter 3. (To be specific, $p_H 0.02$, $p_S = 0.002$, $p_V = 0.02$, $p_T = 0.1$. Location features are discounted, i.e., $p_X = p_y = 1$.) These spread values lead to a fairly even distribution of feature values on the rang between zero and one. We discretize the range into five bins.

In a sense, this method borrows its motivation from the correlogram statistic. Correlograms also define rings (albeit rectangular ones) on which they compute a strict notion of the similarity with the center pixel, namely the proportion of other pixels of the identical color. However, the scales differ considerably: correlograms count pixels instead of the small patches that image tokens represent, and the largest correlogram ring covers not quite 1% of the total image area. Furthermore, correlograms use a one-dimensional notion of similarity, measuring only the number of pixels that match. R-Stairs IV adjusts scores according to both the quality and quantity of matching tokens.

R-Stairs IV is based upon the fine quantization, meaning that it uses in total an $\mathcal{F}$-space with $128 \times 6 \times 25 \times 5^4 = 1.2 \times 10^7$ dimensions. This is the largest of the four R-Stairs methods, and therefore noticeably slower than the others. (Computing **Sf** for batch comparisons can take tens of seconds. On the other hand, calculating individual comparisons between images takes a few thousandths of a second because the $\mathcal{F}$-representations are highly sparse.)

4.3 Evaluation

The R-Stairs algorithms may be evaluated on the same tasks used to evaluate Stairs. We focus here on the classification task as the more intuitively understandable evaluation method. The following section presents the results for this tests.

4.3.1 Class prediction

Table 4.1 summarizes the performance of the four methods described above on the classification test sets. All tests are 5×2 cross validation tests, using the same methodology as described in Chapter 3 applies here. As before, the mean variation between repetitions on different train/test splits is small, on the order of one percentage point or less. For comparative purposes, the table includes the results for basic Stairs and the finely-quantized Stairs.

The first three R-Stairs algorithms perform poorly in comparison with basic Stairs: None of the three beats any of the baselines. Of the three, Method II is weakest no matter which weighting scheme is used, while Methods I and III do slightly better and are roughly comparable. This may simply reflect the fact that they are more like basic Stairs than Method II is, and therefore the added features may not hurt as much. Method II is the only one to attempt to include all pairs of tokens, and it is also the only method using the larger tokens from the mid-level segmentation. Either of these factors, particularly the latter, could explain its lower performance.

R-Stairs IV does better than the other three methods, but is still not quite competitive with the top approaches. Why does this particular method of incorporating context work better? Interestingly, rather than trying to describe the context, it merely notes the similarity of the context to the primary token. The success of correlograms indicates that this approach may

Table 4.1: Results for Relative Stairs algorithms on classification tasks. None outperform the basic Stairs algorithm on any task.

Algorithm	Test Set 1	Test Set 2
Stairs	± 0.8	54.0 ± 0.6
Stairs Fine	66.1 ± 0.5	58.6 ± 0.5
R-Stairs I	57.8 ± 1.8	53.7 ± 1.0
R-Stairs IIa	50.1 ± 0.9	41.6 ± 0.5
R-Stairs IIb	57.8 ± 0.9	49.2 ± 0.8
R-Stairs III	61.2 ± 1.2	52.3 ± 0.5
R-Stairs IV	64.2 ± 0.9	57.8 ± 0.9

offer more benefits than context modeling does. However, it still falls short when compared with less complex approaches that do not account for context.

4.4 Conclusion

In spite of the intuition motivating the development of Relational Stairs, none of the implementations we tried demonstrate encouraging results. While it is impossible to rule out the possibility that some other implementation incorporating context effects could increase retrieval accuracy, these results do suggest that there is no easy solution. Taken as a whole, the three R-Stairs methods cover a range of possible implementation details and design choices. Because they share little in common besides the basic Stairs architecture, one cannot easily explain away their collective performance in terms of these factors. Instead, it seems most likely that visual context is not as important a cue for image retrieval as was thought, or that including it somehow detracts from the algorithms' performance as much as it helps.

Although the disappointing results for the R-Stairs techniques have led us to the conclusion that incorporating context is not helpful in full-image retrieval, it may yet prove important in another task. When looking at retrieval of a cohesive object extracted from within a larger picture, context seems potentially more important than in the case of full images. This is because the object, so far as it is anything, is the context – the particular arrangement of image tokens close together in the image plane. This thought is pursued farther in the following chapter.

Chapter 5

Partial-Image Retrieval

Traditionally, research on image retrieval has focused on comparing one entire image with another. One reason for this history is that full-image comparisons are simpler, requiring no segmentation or combinatorial search. For images that are scenes or landscapes, full-image comparisons make good sense, because there may be no single object or portion of the image that is most important. However, people often connect images based upon some common subject. For example, a user might desire to find "pictures of elephants". In such a case, the elements in the picture that surround the elephant may not be useful in guiding retrieval.

There are exceptions. In situations (perhaps when dealing with portraits, close-up shots, etc.) where images consist of a single subject and an empty background, comparing full images may still work because the background does not provide much distraction. If the desired subject takes up most of the image area, then again a full-image comparison may successfully identify related images because the distractions are minimized. Unfortunately, images fitting the target description that also happen to contain significant irrelevant background will be difficult to retrieve using full-image comparisons.

Another exception worth noting is when the background, though irrelevant, happens to be the same in both query and the candidate under consideration. This can happen for several reasons. Some subjects tend only to appear in specific contexts, so that independently created images will have a high probability of appearing on similar backgrounds. Another factor present in the types of commercial image collections often used to evaluate image retrieval is the inclusion of multiple images taken at the same time and place, by the same photographer. These will often tend to have similar backgrounds, simply because the photographer and subject may not move much between shots. Others have already noted that photographs taken at the same time or on the same roll of film have a high probability of being related [60], and it makes sense to use information on progeny when it is available. However, the effect of using these artificially related images to test retrieval algorithms for supposedly independent images has not yet received the attention that perhaps it should.

In any case, in order to identify the set of images containing a query subject that may not encompass the entire image, one should use method of comparison that does not automatically compare one entire image with another. Such techniques will be referred to as *partial-image* comparisons, and retrieval based upon such comparisons as *partial-image retrieval.* Progress in this area has proven difficult, but the Stairs framework provides new ways to look at the problem.

5.1 Existing Techniques

Color histograms, now used typically for full-image comparisons, were originally developed with the goal of object recognition in mind[74]. The original experiments using color histograms were performed on a collection of only a few hundred images. Unfortunately, histogram techniques quickly begin to confuse objects when the size of the database gets large. One may frame the problem in the following way: Given the possibility of different views and lighting conditions, a range of color distributions could indicate the presence of a particular target object. With a sufficiently large number of objects, the space of potential color distributions is not large enough to contain separate non-overlapping ranges for each object even if the objects have unique color distributions under ideal conditions. Add in the possibility that distinct objects might in fact have quite similar color distributions (brown-furred animals, for example) and the problem becomes even more difficult.

In order to address these issues, researchers have proposed computing statistics of the object coloration that may be more distinctive than simple counts of the amount of each color present. For example, some researchers have proposed using the moments of the color distribution instead [73]. Others introduce the color adjacency graph [53], a construct that records which colors are found next to each other. All of these are reasonable approaches, and in general they make some tradeoff between the level of detail allowed in an object description and the speed of the search. However, a data acquisition bottleneck holds back most object-based retrieval schemes, including those mentioned above. Specifically, in order to compute the description of an object (as opposed to an entire image), the object must first be identified to the system. This presents practical problems for anyone wishing to search for objects in a large collection, as objects must be either labeled by hand (a labor-intensive process fraught with error) or via automatic techniques (which are typically even less reliable, and not necessarily faster). Nevertheless, several systems have been built based upon automatic high-level segmentation [20, 50, 23]. These can be used to find objects that happen to correspond to the predefined segmentation, although their overall performance has been somewhat disappointing.

Given the data acquisition bottleneck, we need to develop methods which do not require careful or accurate segmentations to be provided in advance. Typically, these work with a description of the entire image, and can dynamically extract a portion of the description that best corresponds to the query object. For example, a technique based upon the earth mover's distance (EMD) focuses on quickly finding images containing a given color distribution [22]. Because color distributions can vary greatly, this technique is best suited for finding distinctive patterns of color that tend not to vary much, such as trademarks and consumer packaging. Another technique attempts to adapt color correlograms for object-based retrieval [40]. However, this approach is limited in that the object based correlograms still retain information from their embedded surroundings, and the correlogram computed within this area is compared for compatibility with the uncropped correlograms of all the images in the collection. (Correlogram values are highly sensitive to boundary effects. In order to minimize boundary-related distortions, the value of the correlogram for a given distance must include all jumps of that distance, whether they land inside or outside the object. The alternative, disallowing jumps that land outside the box, skews the probability distributions and results in correlogram values that are incomparable with each other.)

Using the Stairs framework, partial-image retrieval can be achieved that simultaneously satisfies two important goals: It bypasses the data acquisition bottleneck, operating on the fully general Stairs image representation, and it offers a richer query language than simple color distributions. The method allows arbitrary portions of an image to be specified as a query, and only considers matches between relevant parts of the image descriptions. There exists a small amount of other research along these lines [55], but it has not been extensively evaluated and does not benefit from the systematic framework provided by Stairs.

5.2 Partial-Image Retrieval in the Stairs Framework

Even within the framework established by Stairs, there are a number of ways in which partial-image retrieval might be implemented. This chapter will examine three methods in particular and analyze the strengths and weaknesses of each. The primary evaluation tools consist of a sets of images carefully selected so as to eliminate any information provided by the surrounding context. There are two such image sets, described further in Section 5.3.1 and Section 5.3.2. However, before looking at the implementations or the evaluation, some underlying foundations must be laid as background.

5.2.1 Preliminary Considerations

Before examining the specifics of partial-image retrieval in Stairs, a bit of reflection on the task will help clarify issues that arise later. The usual issues of semantic similarity vs. visual similarity apply in this case as to the case of full images. When the user indicates a region of interest as the basis for retrieval, he or she expects to receive images containing the same entity (semantic similarity). In certain cases, this may not correspond to images with a region that looks the same (visual similarity). Nevertheless, like most image retrieval systems, Stairs retrieves images based upon visual similarity because this is the only information readily available to it. This notion is formalized via the area-matching assumption, which holds that the most relevant images are those containing regions that closely match the region specified in the query.

In short, the goal of partial-image retrieval in Stairs will be to retrieve images containing a region that is visually as similar as possible to the selected query, and surrounded by regions that may exhibit any quality at all. (In cases where the context is deemed important, this may be accounted for by using some weighted combination of the partial- and full-image queries. Nevertheless, for research purposes it makes sense to look at the problem of pure partial-image retrieval.) One question that arises is whether to try to match the scale of the query region. For example, if a relevant image exhibits a given fraction of its area that is highly similar to the query region, should another image with twice as much highly-similar area be deemed even more relevant? Careful reflection suggests that this is not desirable, as it would lead to the highest relevance being assigned to images that look entirely like the query region. In effect, this setup amounts to full-image retrieval using a small query image. While closeup photographs do exist and in some circumstances may be desirable retrievals, in most cases the goal is to find the same object, at the same scale but on a different background. Thus as a reasonable default behavior, the system should try to retrieve images that match the same area as specified in the query image, with differences in total area resulting in correspondingly decreased relevance. The background may be anything that contrasts with the specified query region; including non-contrasting backgrounds would again favor the retrieval of full-image closeup views of the query subject. (Cases where the target object often appears on a similar background, such as animals with natural camouflage, present inherent difficulties for partial-image retrieval. If the discriminatory power of the system's image representation cannot sufficiently distinguish the object from the background, then full-image retrieval may be the best available option.) Each of these considerations must guide the development of the partial-image retrieval algorithms described in the following sections.

5.2.2 Method I: Feature-Space Division

One way of performing partial-image retrieval is to attempt dynamic segmentation of images. This must be done quickly in order to provide a reasonable response time, and must take the information provided by the user as a basis. The definition of image feature space provided by Stairs makes such a dynamic segmentation possible.

The user's query explicitly defines a division of the image into subject and background regions. Stairs uses this division to implicitly segment each of the images stored in the collection. It accomplishes this by dividing $\mathcal{M}$-space into two subsets: $\mathcal{M}_Q$, which contains tokens found in the query region and potentially similar tokens, and $\mathcal{M}_B$, which contains all other (background) tokens. These subsets implicitly segment every image in the library, isolating and identifying areas similar to the query region. (Unfortunately, because the success of the implicit segmentation depends upon the subject and background falling into different bins, it will not work well in cases where the object is well camouflaged. In a sense, such cases are intrinsically more difficult.)

To avoid explicitly isolating the interesting portions of each target image, the distinctions between $\mathcal{M}_Q$ and $\mathcal{M}_B$ can be rolled into Equation 3.7 by making an appropriate choice of $\mathbf{S}$. Consider two match matrices, $\mathbf{S}_Q$ and $\mathbf{S}_B$, generated from different spread parameters p_{F_i}. $\mathbf{S}_Q$ requires a close match on all token features (although it could be relaxed in some ways, for example to allow the location to vary). $\mathbf{S}_B$ allows any tokens to match, i.e., $\forall_{F_i} p_{F_i} = 1$. These two matrices are combined as follows to form $\mathbf{S}$:

$$\mathbf{S}(i,j) = \begin{cases} \mathbf{S}_Q(i,j) & \text{if}(m_i \in \mathcal{M}_Q) \wedge (m_j \in \mathcal{M}_Q) \\ \mathbf{S}_B(i,j) & \text{if}(m_i \in \mathcal{M}_B) \wedge (m_j \in \mathcal{M}_B) \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The resulting $\mathbf{S}$ matrix compares potential subject features to the query subject features, and ignores the background features. With different choices of $\mathbf{S}_Q$ and $\mathbf{S}_B$ other queries could be formed. For example, Stairs could search for an arbitrary object in a fixed background, or a specific object on a highly-textured background of any color, etc.

A key detail in implementing this retrieval method is deciding how to split $\mathcal{M}$-space. This is particularly difficult when certain features appear in both the object and the background. The current implementation applies the spread matrix $\mathbf{T}$ to both the selected query region and the background, and splits $\mathcal{M}$-space according to which area is strongest in each component. It seems that a good split in feature space is crucial to achieving good retrieval results with this technique, so continued research on this problem will be important.

Thus Stairs accommodates dynamic segmentation for partial-image queries without any major changes to its architecture. Unfortunately, because a new $\mathbf{S}$ is used in response to each region query, the tactic of pre-computing the denominator in Equation 3.7 cannot be used. Even so, the entire region query computation for 20K images takes only a few seconds on a PC with a reasonably fast processor. This is competitive with times reported by other groups for region matching [19].

5.2.3 Method II: Vector Components

A second method of performing partial-image retrieval stems from a component analysis of the image vectors in $\mathcal{F}$-space. Component analysis is fast and easy to perform. On the other hand, the interpretation of its behavior in terms of the original image is less clear than in the case of dynamic segmentation.

Consider an image with $\mathcal{F}$-vector $\mathbf{f}$, and suppose that a user specifies some portion of the image as the query. In accordance with Section 3.2.1, the image tokens within the specified region have an $\mathcal{M}$-representation, and a corresponding $\mathcal{F}$-representation $\mathbf{f}_Q$. Because it describes a subset of the image tokens that $\mathbf{f}$ describes, each component of the vector $\mathbf{f}_Q$ is strictly less than or equal to the corresponding component of $\mathbf{f}$. One can construct a vector $\mathbf{f}_B$ out of the remaining background tokens, representing the difference between the two.

$$\mathbf{f}_B = \mathbf{f} - \mathbf{f}_Q \tag{5.2}$$

Splitting $\mathbf{f}$ into components is important because it isolates the features that a user is interested in. Images that contain an area like the query region will have vectors that include a component similar to $\mathbf{f}_Q$, plus an arbitrary background component $\mathbf{f}'_B$. These will show up, respectively, as components parallel and perpendicular to $\mathbf{f}_Q$, since a parallel component indicates the presence of image tokens that appear in the query region, while a perpendicular component indicates the presence of dissimilar image tokens. Images with less area that is like the query region will have a smaller component in the $\mathbf{f}_Q$ direction, while images with too much area like the query region will have a larger component in this direction. Thus measuring the balance between the two components provides a way to check for the presence of the right amount of query-like area. If we define a synthetic vector ϕ for each $\mathbf{f}$, consisting of the components of the normalized $\hat{\mathbf{f}}$ parallel and perpendicular to $\mathbf{f}_Q$, then we can retrieve images based upon the angle between their synthetic vectors and that of the query image.

$$\phi = \left\langle \left\| \frac{(\mathbf{f}_Q \cdot \mathbf{f})}{\|\mathbf{f}_Q\| \, \|\mathbf{f}\|} \right\|, \left\| \frac{\mathbf{f}}{\|\mathbf{f}\|} - \frac{\mathbf{f}_Q}{\|\mathbf{f}_Q\|} \frac{(\mathbf{f}_Q \cdot \mathbf{f})}{\|\mathbf{f}_Q\| \, \|\mathbf{f}\|} \right\| \right\rangle \tag{5.3}$$

$$D_{VC}(I, I') = cos^{-1}\left(\phi_{I_1} \cdot \phi_{I_2}\right) \tag{5.4}$$

For the reasons discussed in Section 3.2.2, it is better to deal with spread vectors. Thus, in matrix form and using $\hat{\mathbf{Tf}}$ to denote $\mathbf{Tf}$ normalized to unit length, Equation 5.3 may be rewritten.

$$\phi = \left\langle \left\| \mathbf{T}\hat{\mathbf{f}}_Q (\mathbf{T}\hat{\mathbf{f}}_Q)^T \mathbf{T}\hat{\mathbf{f}} \right\|, \left\| \mathbf{T}\hat{\mathbf{f}} - \mathbf{T}\hat{\mathbf{f}}_Q (\mathbf{T}\hat{\mathbf{f}}_Q)^T \mathbf{T}\hat{\mathbf{f}} \right\| \right\rangle \tag{5.5}$$

Equation 5.4 applies as before. The quantities in this equation are not difficult to compute from the standard Stairs representation. The first (parallel) component of ϕ is equivalent to a standard retrieval on $\mathbf{f}_Q$. Computing the second (perpendicular) component of ϕ requires instantiating a vector in $\mathcal{F}$. This may take a while in implementations where $\mathcal{F}$ has many dimensions, but for the standard implementation with a few thousand dimensions it is also quite fast.

5.2.4 Method III: Explicit Area Matching

From the point of view of the area-matching approach, the ideal algorithm for partial-image retrieval would explicitly match the selected area in the query image with the best-matching area in each image of the library and retrieve those with the best overall match. This can actually be done, although in general it is slower than the two methods described previously. However, it gives an approximate upper bound on the performance that may be expected from any algorithm motivated by the area-matching approach.

Explicit matching may be set up as a minimum-cost network flow problem. Polynomial-time algorithms for network flow have been known for a long time [3], and code to solve network flow problems is available on the web [35, 34]. The best general algorithms run in the worst case in approximately cubic time in the number of network nodes. In the current application, the number of nodes is proportional to the number of tokens per image (about 500).

Minimum-Cost Network Flow

For convenience, a synopsis of the network flow problem will be given here. Network flow is defined on a directed graph, with the number of nodes designated by n. Each node has an associated supply (or demand, if negative) s_i, representing the amount of flow it produces (or consumes). These numbers must be assigned so that the total supply equals the total demand, i.e., $\sum_{i=1}^{n} s_i = 0$. Each arc has upper and lower bounds u_i and l_i on the flow that it may carry, with negative numbers indicating flow opposed to the direction of the arc. Furthermore, each arc has a cost per unit flow c_i.

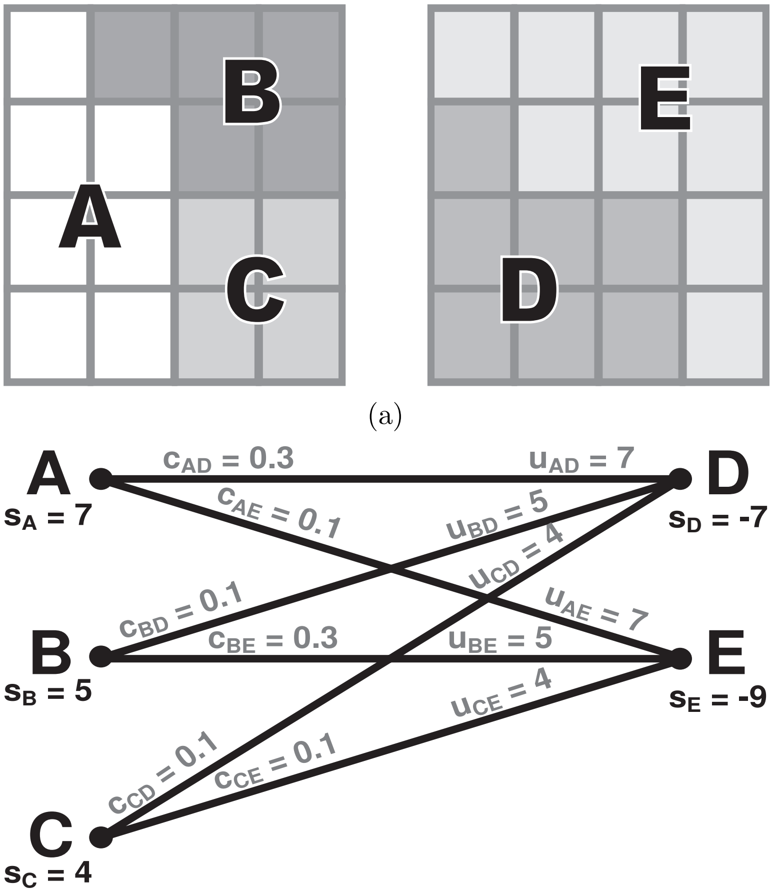
A valid flow on such a graph satisfies two conditions. First, the sum of the flows into every node plus the supply at the node must equal the sum of the flows leaving the node. Second, the flow along every arc must be between the upper and lower bounds specified for that arc. Given a valid flow, its total cost is simply the sum of the flow assigned to each arc times the cost of the arc. A minimum-cost flow for a given network is a valid flow with minimal cost. In general, the minimum-cost is unique, although there may be multiple minimum flows with equal costs.

Explicit Area Matching as Minimum-Cost Network Flow

We set up the explicit area matching problem as a minimum-cost network flow problem. Suppose that I_1 and I_2 are to be compared. Define a graph with one node for each image token in the two images. Flow in the network will correspond to matching between image tokens. Nodes corresponding to tokens from I_1 are assigned a supply equal to the token area (in pixels), while nodes corresponding to tokens from I_2 are assigned a demand equal to the token area. (This assumes that the two images have equal area to begin with; if not then the values must be scaled so that the supply and demand sum to zero.) For convenience, call the two groups of nodes respectively the left hand side and the right hand side. Connect each left hand node with each right hand node. The maximum capacity of each link will be the lesser of the areas of the two tokens corresponding to the nodes on either end; the minimum capacity is zero. Assign a cost to the arc that is the inverse of the similarity between the two tokens. We use the matrix $\mathbf{S}$ of match coefficients to generate the costs: given tokens corresponding to the ith and jth row and column, the cost assigned to the arc is $1 - \mathbf{S}_{ij}$. Figure 5.1 shows a simple example of two 4×4 grayscale images and the resulting network.

A valid flow on the resulting network corresponds to a matching between the areas of the two images. Since there is no capacity for backward flow in the graph, each unit of supply (representing a pixel in the left hand image) must flow across a link and be consumed by a unit of demand (representing a pixel in the right hand image). Finding the minimum flow ensures that the matching has the lowest global cost. The cost of the minimum flow indicates the overall quality of the match, and thus the degree of relevance between the two images under the area-matching assumption.

A simple modification allows the same setup to be used for partial-image matching. Tokens in the selected query region are assigned to left hand nodes as before. The areas of the image not selected form a single additional left hand node, with supply equal to the entire unselected



(a)

(b)

Figure 5.1: Minimum-cost area matching.
(a) Two 4×4 grayscale images. (b) The network for computing the minimum-cost area matching between them. The cost on each arc reflects the difference in shade between the regions. The minimum-cost matching pairs 7 pixels of A with E, 5 pixels of B with D, and two pixels of C with each of D and E. Note that location does not affect the matching in this example.

area. The difference is that all arcs leaving this special node have zero cost, reflecting the desire to allow arbitrary matches in the background. As a result, the minimum cost flow on the new network reflects the best match for the selected query region.

Although the network flow formulation implements the area-matching approach explicitly, it is not as fast as the standard Stairs techniques. At about 500 tokens per image, it can compare perhaps twenty full images per second. On the other hand, region matching can be orders of magnitude faster, if the query region has many fewer segments than the full image.

5.3 Evaluation of Region Matching

There exist no widely accepted standards for evaluating region matching. One typical approach is to select by hand a few images that contain related motifs, and use one as a query to retrieve the others [40]. Depending upon the images chosen, it is possible to skew the apparent performance of any given algorithm using this approach. We will attempt to address these concerns by using relatively large, well-defined categories in the tests presented here. One consequence of this policy is to make the retrieval task more difficult, because broad categories tend to exhibit less visual similarity over their range than do a small set of hand-picked photographs. Nevertheless, it is important to reduce the role of subjectivity in evaluation to the greatest extent possible. In this section, we use two test sets to evaluate the effectiveness of partial-image retrieval algorithms. The first is a small set of 200 images of cars, and the second is the entire set of 20,100 images described in Chapter 3.

5.3.1 The Car Test Set

The collection of cars, although small, exhibits several important properties crucial to the effective assessment of partial-image retrieval. The first, and perhaps most important property, is that the background context is unhelpful in guiding retrieval. In the presence of many unrelated images, the detection of pavement and/or roadway would provide a strong indication that a car might be present, thus biasing the results of full-image retrieval. By limiting the collection solely to images of cars, almost all of which have a roadway as background, the information provided by the background is eliminated.

The car image collection comprises both race cars and street cars, of various colors. There are 66 red cars, 45 white, 31 black, 14 yellow, 12 silver, 9 blue, 2 green, and 21 that could not be classified, for example because they had no single dominant color. The set task is to retrieve cars of a similar color, given one example. (Note that this simple task gives an advantage to methods based purely upon color, such as histograms.) We conduct testing by leave-one-out cross validation, using one image as a query and the remaining set as potential targets. The results are averaged across all the examples of a particular color.

Recall and Precision

We present the results as precision-recall curves. These are a standard tool in information retrieval [66], but a short summary appears here for those unfamiliar with their use. Four underlying quantities make up the definition of precision, recall, and other measures used to evaluate retrieval and classification (such as the receiver operating characteristic, or ROC curve). These are T^+, equal to the number of true positives (relevant images retrieved), T^-, equal to the number of true negatives (irrelevant images not retrieved), F^+, equal to the number of false positives (irrelevant images retrieved), and F^-, equal to the number of false negatives (relevant images not retrieved). Note that $T^+ + T^-$ gives the number of images handled correctly, and $F^+ + F^-$ gives the number of images handled incorrectly. Furthermore, $T^+ + F^+$ gives the number of images retrieved, and $T^- + F^-$ gives the number of images not retrieved.

Typically, the total number of images in a collection is fixed, and the number of images retrieved is a parameter that may be set to different values. Two more quantities must be known in order to fix the values of the four underlying numbers. The quantities most often quoted are recall, R, and precision, P.

$$R = \frac{\text{\# relevant retrievals}}{\text{total \# relevant images}} = \frac{T^+}{T^+ + F^-} \tag{5.6}$$

$$P = \frac{\text{\# relevant retrievals}}{\text{total \# retrievals}} = \frac{T^+}{T^+ + F^+} \tag{5.7}$$

For many applications, the high-precision end of the curve is most important. This region represents the images judged most relevant to the query. In terms of a search engine where only a limited number of images may be displayed, this would correspond to the first page of hits. Thus achieving high precision at the lowest recall values is an important goal.

As an aside, some fields show results as ROC curves. These contain the same underlying information, but plot it as sensitivity (another word for recall), vs. 1 minus the specificity, S.

$$S = \frac{\#\text{ irrelevant non-retrievals}}{\text{total }\#\text{ irrelevant images}} = \frac{T^-}{T^- + F^+} \tag{5.8}$$

With some effort, an ROC curve may be converted into a precision-recall curve, and vice versa.

Full-Image vs. Partial Image Retrieval

Figure 5.2 shows precision-recall curves for several algorithms on selected color classes. Not all colors are shown, but the figure includes representative examples of both high- and low-frequency classes. The three algorithms described in Section 5.2 are shown, together with the full-image Stairs as a control. All three algorithms based upon partial-image retrieval do better than the control. However, their relative performance varies also.

In general, Method III outperforms the other two algorithms. However, it is also the slowest. Methods I and II have different strengths and weaknesses. As mentioned in Section 5.2.2, Method I does well when it is easiest to split the feature space, i.e., on classes that have good contrast with the background. Method II performs more evenly, but never stands out.

As mentioned earlier, both color histograms and correlograms can be modified to perform partial-image retrieval. Because these methods are both color-based, they might be expected to do quite well on the car data set. However, as shown in Figure 5.3, the area matching approach (Method III) performs as well as or better than these other methods.

5.3.2 The Full Test Set

Useful as the car test set is for comparing partial-image retrieval algorithms on a small scale, it does not necessarily provide the best indication of their performance in a large image library. Problems might appear in scaling up that are not evident in the small test set. For this reason, we conduct test runs using the entire 20,100 image set. These tests do not use the car images because cars tend to appear on a consistent background, making full-image queries much more powerful. In fact, the full image queries are sometimes more successful at retrieving car images from the full test set than are partial-image queries.

In the context of a large image library, one would expect region queries to be most effective when the target object appears on a variety of different backgrounds. An appropriate test set should also include enough images to give a meaningful average behavior. In some sense these goals contradict each other: One wants variety in the backgrounds, but consistency in the foregrounds. We solve the problem by identifying a set of images that typically appear on one of several different backgrounds. We split the image set into subsets, one for each background type, and form folds. Each fold uses the images from one background type as queries and the remaining background types as the targets that must be found.

The test set we use with these properties is a collection of 110 images of wolves, on three background types. Of the wolf images, 53 appear on snowy backgrounds, 19 on green forest backgrounds, and 38 on other backgrounds such as rock or dirt. Figure 5.4 shows an example from each of these categories.

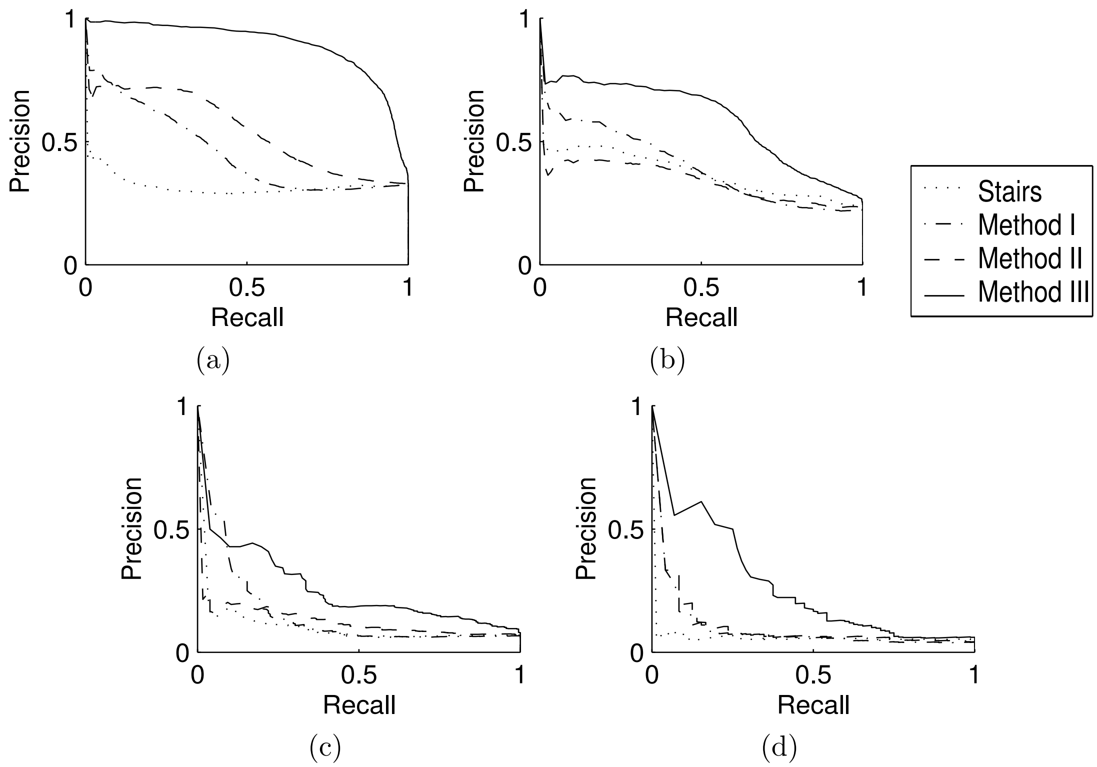


Figure 5.2: Precision vs. recall graphs for queries on color images of cars. Red (a), white (b), yellow (c), and blue (d). (Dotted line = Full-image Stairs; dashdot = Method I; dashed = Method II; solid = Method III)

In general, partial image retrieval becomes much more difficult in the context of large image libraries. Wolves are difficult to identify because there are many objects with colors and textures similar to wolves. The overall scores are not high for any of the algorithms tested on this task. Nevertheless, the relative performance is interesting. Table 5.1 shows the area under the recall-precision curve for various algorithms. For comparison, a score of 100% would indicate perfect retrieval, i.e. all the target wolves were returned as the top-ranked images.

The first line of Table 5.1 shows the expected performance if the images were retrieved in random order, and represents the underlying prevalence of the target images in the library. None of the algorithms does very well in retrieving wolf images; at best they do only a few times better than chance. The low scores reflect the difficulty of the task. Interestingly, the partial-image versions of correlograms underperforms the corresponding full-image version. This high performance for full-image correlograms is puzzling, but probably results from the strength of residual context effects. For example, some of the images contain both snow and greenery in their background. Although they are classified according to the dominant background component, the presence of mixed backgrounds may have helped the full-image correlogram to retrieve similar images in the other folds, boosting its performance.

As expected, the Stairs Matching (Method III) algorithm does consistently better than full-image Stairs, and is at or near the top performance on each of the three folds. It beats each of the other partial-image techniques on every fold. The partial image version of histograms

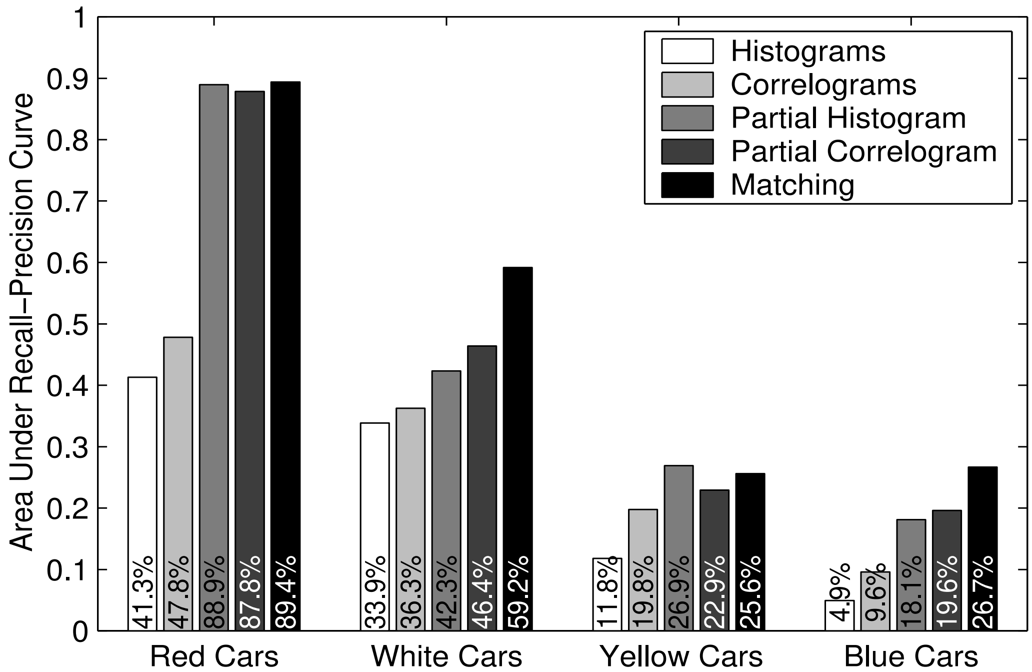


Figure 5.3: Area under the recall-precision curve on the car data set for selected algorithms. The explicit area matching technique performs best overall.

also does slightly better than the full image version, but compared to the matching technique it gives inferior results.

5.4 Conclusion

Partial-image retrieval is a difficult but potentially useful ability. Full-image retrieval works quite well in many cases, because context effects often aid retrieval even when the user is only interested in a foreground object. Nevertheless, context can also detract from retrieval if the query image contains a background that happens to be unusual or highly variable. In such cases, partial-image retrieval is a vital tool in the user's repertoire.

(a) (b) (c)

Figure 5.4: Pictures of wolves appear on different backgrounds.

Table 5.1: Area under the recall-precision curve, for three different query sets. (The images in the remaining two sets form the target set.)

Algorithm	Snow Wolves	Green Wolves	Other Wolves
Random	0.3%	0.5%	0.4%
Full Histograms	0.4%	0.5%	0.4%
Full Correlograms	0.8%	1.0%	0.7%
Full Stairs	0.6%	0.7%	0.4%
Partial Histograms	0.5%	0.8%	0.4%
Partial Correlograms	0.5%	0.9%	0.4%
Partial Stairs Matching	0.7%	1.4%	0.6%

In order to be most useful, partial-image retrieval should not depend upon human annotation or other involvement when the image is introduced into a library. Ideally it should also not depend upon automatic segmentation of the image, since such segmentation is error-prone and unable to adapt to the context of a particular user query. Instead, one would like to make any segmentation or region selection dependent on the user at the time the query is generated. Dynamic region selection affords the most flexibility for the user. However, it places severe demands upon the system, because interaction constraints limit the time available for computation. Thus a successful partial-image retrieval system must be able to rely upon precomputed information while retaining the flexibility to respond to the user's demands.

This chapter develops three algorithms for partial-image retrieval based upon the Stairs architecture. Each relies on the user to specify a region of interest in a query image, and then formulates a search based upon the description of that particular region. The first two of these work directly with the $\mathcal{F}$-space vector descriptions. The third carries out an explicit optimal matching between the vector components, representing the different areas in the image. Of the three, the matching technique shows the best results as measured by the area under a recall-precision curve. This method outperforms the other two on a small test set comprising images of cars. It similarly does better than partial-image retrieval methods based upon color histograms and color correlograms. In a test using the entire 20,100 image collection, it also beats the other methods in retrieving images of wolves on varying backgrounds. Its one drawback is speed; it is probably too slow to work interactively in very large collections, at least without large-scale use of parallelism or some sort of pre-sorting to identify good potential candidate.

Apart from the practical issues, the success of the matching technique represents a vindication of the area-matching approach. The algorithm explicitly matches the region of interest in the query image (as specified by the user) with the closest match in each of the library images, and returns the target images that exhibit the highest quality match. Interestingly, this approach seems to work better in the context of partial-image matching than for full image matching. This probably may be explained by an appeal to different kinds of similarity involved in the two tasks. Partial-image techniques are well suited for detecting similarity based upon the presence of a particular object, because that object will tend to create areas that look similar in all the images where it appears. Full-image similarity, on the other hand, may stem from other factors, such as a "thematic similarity", that do not necessarily mean the same objects are included in the image. The area-matching approach is most likely to work where the similarity of images is based upon similar composition in the entities making up the scenes in question.

Chapter 6

Stairs and Machine Learning

6.1 Introduction

Historically, there has been little useful connection between research working with images and research investigating machine learning in other contexts. This situation is a bit odd, because machine learning research regularly deals with issues of comparison, classification, and similarity, all of which are important for image retrieval and comparison. This chapter will look at the links between the research on image comparison and retrieval presented in the rest of this dissertation on the one hand and the historically separate body of research in machine learning on the other.

The first part of the chapter shows that the techniques developed herein, designed originally with images in mind, have direct applications to other forms of data, and are thus of interest to the machine learning community as a whole. The second part of the chapter shows that techniques developed by researchers in machine learning have immediate application to problems relevant to those interested in image retrieval and comparison. Thus the chapter as a whole may be taken as evidence that stronger ties between the two communities could benefit both sides.

6.2 Stairs for Machine Learning

Images contain multiple layers of complex substructures. Dealing with this complexity forms one of the motivations behind the Stairs architecture. Yet the application of Stairs need not be confined to images and visual data. Many other types of complex data come in forms that are amenable to the same approaches that work on image data. In such cases, it is possible to apply the Stairs architecture to create vector representations of the data. Once the data are represented in vector form, many popular algorithms, including most machine learning approaches, can be applied. This half of the chapter looks at the use of Stairs in machine learning applications, as well as at the use of results and techniques from machine learning to expand the range of tools for manipulating images.

A quick perusal of the UCI repository of machine learning data sets reveals that the most frequently cited entries consist of data that are condensed into a convenient format easily digested by most machine learning algorithms [12]. Typically such data consist of a set of instances I, perhaps already divided into subsets for training and testing. Each individual instance is described by a set of features X, including a class feature that the learning algorithm must predict accurately.

Although the data sets in the UCI repository provide a convenient testbed for new ideas in machine learning, they do not fully represent the difficulty of solving problems encountered in the real world. Often the hardest part of applying learning methods to a previously unexamined task is the codification of the problem in a form that machine learning algorithms can handle. This step has already been performed on most of the repository data, and usually there is no access to the original form of the data set or documentation on how it was transformed. Thus there is need for research and discussion on the analysis of data in a more natural format.

Furthermore, some types of raw data may not be amenable to expression in the standard feature-value format, and therefore require special treatment. For example, some tasks involve learning properties of entities that are themselves made up of an arbitrary number of components. As a concrete example, consider learning properties of credit accounts, where each account is represented as the set of transactions posted to the account. This chapter will refer to such collective entities, or *ensembles*, in particular when the constituent components, or *records*, happen to have a concise featural description. A direct analogy may be drawn between ensembles and images, and between records and image tokens. This analogy drives the work described below, and the astute reader will notice the parallels between the processing of images in Chapter 3 and the processing of ensembles described below.

Instead of devising features that summarize the ensemble as a whole, which can obscure useful information, we adopt a description that preserves important details of each individual record. The general algorithm presented in this dissertation automatically generates a feature vector of uniform size from any ensemble, provided that the component records have a standard feature-vector description. This provides a novel way to look at data from many domains, including computer event logs, credit card histories, and others where ensemble data are involved. We present insight from implementations of the technique to a domain that in superficial aspects differs greatly from collections of images: climate measurements from floating buoys in the Pacific Ocean.

The remainder of this half of the chapter describes the treatment of ensemble data. Section 6.2.1 gives a description of the algorithms for processing and comparing data ensembles. Section 6.2.2 presents the two test domains and examines the performance of the system on them. Finally, Section 6.2.3 concludes with a discussion of how the ensemble approach fits in with other work in machine learning, and possible future directions.

6.2.1 Handling Ensemble Data

In the type of domain addressed here, the task requires learning properties of ensembles of records. By definition, each ensemble may contain an arbitrary number of records. Furthermore, we assume that each record can be described by a simple feature vector. To be precise, we give a formal description of such an ensemble before describing how it is processed.

A record is an arbitrary set of m feature-value pairs, $r = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, where $X = \{x_1, x_2, ..., x_m\}$ is a consistent set of features shared by all records in the data, and the y_j are values of those features. (In some domains, features may be missing from some records, and thus the features of r form a subset $X_r \subseteq X$.) An ensemble is simply a collection (possibly a weighted collection) of records. As a concrete example, in a credit card domain each ensemble might represent one account. Its component records would be the charges posted to the account, each described by a feature set, such as $\{amount, charge\_date, payment\_date\}$. Some accounts would have fewer charges posted than others.

Data Preparation

Processing of ensemble data into a more manageable form takes place in two steps. First, we express the individual records in a discrete space $\mathcal{M}$, which is a discretization of the original feature space. Once this is done, a one-to-one function transforms the entire ensemble into a vector in a high-dimensional space $\mathcal{F}$. Vectors in this space may be thought of as joint histograms of the original record feature values. All subsequent processing takes place in $\mathcal{F}$, which is better suited to the application of standard machine learning techniques.

Records are mapped into space $\mathcal{M}$ by discretizing each feature x_j. Points in $\mathcal{M}$ are tuples of the discretized feature values. Thus, to map a record to a point $\mathbf{m}$ in $\mathcal{M}$ we simply determine the appropriate bin for each of its feature values. For the credit card example just described, a hypothetical record might fall into a bin like $(\$50 - 100, Jun99, Oct99)$.

Ensembles are represented as a set of ordered pairs, each consisting of a point in $\mathcal{M}$ and an associated weight. (Weights arise naturally in some domains, or can be set uniformly to one if not needed. If two or more records are described by the same $\mathbf{m}$, they are represented by a single ordered pair with weight equal to the sum of the individual weights.) We refer to this as the $\mathcal{M}$-representation $R_{\mathcal{M}}(e)$ of the ensemble e.

$$R_{\mathcal{M}}(e) = \{(\mathbf{m}_1, w_1), (\mathbf{m}_2, w_2), ..., (\mathbf{m}_{n_e}, w_{n_e})\} \tag{6.1}$$

where n_e is the number of records in the ensemble. Space $\mathcal{F}$ has exactly one dimension corresponding to each point of space $\mathcal{M}$. Thus the dimensionality of $\mathcal{F}$ is the product of the number of bins used for each feature in X. For example, in the simple credit account domain described above, there might be ten bins for the *amount* feature, and 20 each for the two date features, giving $\mathcal{F}$ a total of $10 \cdot 20 \cdot 20 = 4000$ dimensions. Points in $\mathcal{M}$ correspond one-to-one with the orthonormal basis vectors of $\mathcal{F}$. Thus there exists an isomorphic mapping f between $\mathcal{M}$-representations and vectors in $\mathcal{F}$.

$$f\left(R_{\mathcal{M}}(e)\right) = \sum_{i=1}^{n_e} w_i F(\mathbf{m}_i) \tag{6.2}$$

where $F(\mathbf{m})$ is the mapping from points in $\mathcal{M}$ to basis vectors of $\mathcal{F}$. This means that two ensembles with distinct $\mathcal{M}$-representations also have distinct representations in $\mathcal{F}$.

Vector Comparisons

In many applications, the significant information resides in the distribution of the records in $\mathcal{M}$ space rather than the actual number of records. If this is so, then the natural distance metric to use is the cosine metric, which measures the angular deviation between two vectors and ignores their length. Thus in $\mathcal{F}$ space, the distance between two vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ is

$$D_{\mathcal{F}}(\mathbf{f}_1, \mathbf{f}_2) = \cos^{-1}\left(\frac{\mathbf{f}_1 \cdot \mathbf{f}_2}{\|\mathbf{f}_1\|\,\|\mathbf{f}_2\|}\right). \tag{6.3}$$

While the plain cosine difference metric may work well in some situations, as in the case of images a number of considerations suggest the use of a slightly more complicated form than Equation 6.3. If continuous variables are discretized to form $\mathcal{M}$ space, then their relative ordering is lost. Even for discrete variables, some pairs of values indicate greater similarity than others. We would like to capture this information in the ensemble distance metric, and can do so by modifying Equation 6.3 to include a similarity matrix with cross terms:

$$D_{\mathcal{F}}(\mathbf{f}_1, \mathbf{f}_2) = \cos^{-1}\left[\frac{\mathbf{f}_1^T \mathbf{S} \mathbf{f}_2}{(\mathbf{f}_1^T \mathbf{S} \mathbf{f}_1)\,(\mathbf{f}_2^T \mathbf{S} \mathbf{f}_2)}\right]. \tag{6.4}$$

Here $\mathbf{S}$ is a matrix whose off-diagonal entries account for the varying similarity of different feature values. It should be a symmetric matrix so that distances are symmetric, and can be devised to have a Cholesky factorization $\mathbf{S} = \mathbf{T}^T\mathbf{T}$. Under these conditions, Equation 6.4 can also be interpreted as the simple cosine difference between the two transformed vectors $\mathbf{T}\mathbf{f}_1$ and $\mathbf{T}\mathbf{f}_2$.

The choice of $\mathbf{S}$ will greatly affect distance measurements and therefore any results based upon them. We use the same system that we developed for generating suitable matrices for use with images. This approach has the advantage of allowing the user significant control over how much individual features contribute to the final distance, without an overly complex interface.

We form $\mathbf{T}$ (and hence $\mathbf{S}$) as the Kronecker product (or direct matrix product) of smaller matrices $\{\mathbf{T}_1, \mathbf{T}_2, ..., \mathbf{T}_m\}$, each corresponding to one of the features x_j. This allows us to focus on one feature at a time, and also allows for significant computational efficiencies. For features that were originally continuous, we use cross terms that decay exponentially with the distance between the bin centers:

$$\mathbf{T}_j(k, l) = (p_j)^{\Delta_j(y'_k, y'_l)}. \tag{6.5}$$

Here $\Delta_j(y'_k, y'_l)$ is the distance between the centers of bins k and l, and $p_j \in [0, 1]$ is a parameter that controls the size of the cross terms. If $p_j = 0$, then only exact matches are allowed, while $p_j = 1$ means that any value of feature x_j matches all others equally well. Intermediate values of p_j result in better matches for closer values. Thus the setting of p_j is a knob by which the user can exert control over the system. Although neither of the example data sets we examine in Section 6.2.2 contain discrete features, $\mathbf{T}_j$ matrices for discrete features can be created by an analogous process, by using something like the value-difference metric [72].

Scalability is a valid concern in the system so far outlined, but we have already addressed analogous issues of similarity when dealing with images. Clearly, restraint must be used in discretizing features, since the dimensionality of $\mathcal{F}$ is the product of the number of bins in each feature. Nevertheless, the technique scales sufficiently for the analysis of interesting problems, as the experiment of Section 6.2.2 demonstrates.

6.2.2 Implementation and Evaluation

Although it is easy to imagine domains where data are structured as ensembles of records, actually acquiring such data is more difficult. In many cases, domains that fit the ensemble paradigm are described using summary information, with the raw data in ensemble form not available. For example, the UCI repository contains a credit screening domain, but information on individual accounts is condensed into sixteen features and there is no listing of individual transactions. Nevertheless, some candidate domains can be found. Earlier chapters describe the use of ensemble techniques on image data (Stairs). Here, the technique is applied to the analysis of ocean climate measurements from the Pacific. The point of this exercise is to show that high-quality machine learning can be done under the ensemble of records paradigm. Because the the climate domain differs significantly from the image domain, it reveals different aspects of the approach.

Ocean Buoy Measurements

The data for these experiments come from climate measurements taken from ocean buoys moored in the equatorial Pacific Ocean [7]. This is a natural domain for application of the ensemble approach, since each buoy measurement forms a unit that must be aggregated with others to form a picture of the climate conditions at any one time. The buoy data set offers an interesting contrast with the natural image data: it derives directly from physical measurements rather than calculations, it contains examples of missing data, and it covers the study area in an irregular manner. Furthermore the goal is not retrieval, but detection of patterns in the data. Thus although ensemble methods still form the core of our approach, the specifics will differ somewhat from the previous case.

The ocean buoy data span the period from March 1980 to June 1998, during which time there were four major El Nino events recorded (1982-83, 1986-87, 1991-92, and 1997-98). Buoys take measurements of wind speed and direction, humidity, and temperatures of the air and sea, along with the date and location where the measurement was taken. However, not all buoys are equipped for all types of measurements, and there are gaps in the data, particularly in 1980 and 1983. New buoys were deployed throughout the study period, particularly around 1985-89 and 1991-93.

We discretize the buoy measurements as follows: latitude, one bin; longitude, five bins; zonal and meridional winds, seven bins each; humidity, eleven bins; air and sea temperatures, fifteen bins each. In addition, we include an extra bin for missing values of each feature except longitude. This gives a total of 983,040 dimensions in our final space $\mathcal{F}$. To form ensembles we aggregate all buoy measurements made over a calendar month; in theory this should give a picture of the ocean climate during that month. (If the climate changes over a shorter period of time, we might want to aggregate over a shorter time span, but a month seems suitable since the duration of an El Nino episode is about one year.) We also need to choose $\mathbf{S}$; we experimented with different settings, starting with $p_j = 0.1$ for each feature. Although the results are qualitatively similar for a wide range of $\mathbf{S}$ matrices, we get the clearest results with settings that focus on longitude and temperature. This is unsurprising since El Nino events are identified primarily as a rise in sea surface temperatures in the eastern Pacific [54].

Calculating the difference between each month, the first trend observed is a striking dependence on measurement date (Figure 6.1a). This turns out to result from the addition of buoys over time. Rather than adding buoys in a random fashion, the positions of the new buoys tends steadily westward (Figure 6.1b). This affects the measurements because the Western Pacific is warmer than the Eastern Pacific. Thus the character of the ocean described by the buoy measurements is significantly different in 1980 than in 1990, and the cosine metric reveals this fact.

To account for the nonuniform deployment of buoys, we perform comparisons between months using only measurements from buoys that exist in both months. This procedure eliminates the gross time dependence exhibited in Figure 6.1a, although there are hints of weaker trends remaining over time. Nonetheless, this step clears the way to look for other patterns in the data.

Since there are only four El Nino events during the span of the measurements, setting up a supervised learning task seems overzealous. However, if El Nino events are present in the data, they should present a distinctive pattern that is self-similar and unlike the data of other years. To test this hypothesis, we can plot the cosine scores of an El Nino period compared to all other observations. The result is somewhat noisy, so we smooth locally to get the curve shown in Figure 6.2. This plot shows clear dips in 1982-83, 1986-87, 1991-92, and 1997-98, all El Nino years. The 1991-92 El Nino was followed by two years of lingering El Nino effect, and this also appears in the curve. On the other hand, the 1982-83 El Nino was noted as the strongest of the century, and the size of the dip in the curve does not really reflect this. Missing data in the latter half of 1983 probably causes this effect by narrowing the minimum, which is then filled in by the smoothing step.

As a comparison, we also plot a curve for the La Nina year 1995-96 in Figure 6.3. La Nina is the opposite of El Nino, so this curve should show minima in different spots. Indeed, the curve is high during El Nino years, but dips down during the La Nina year 1988-89. It also shows a fairly significant minimum around 1984-85, which was not classified as a La Nina episode. However, since this immediately follows the 1982-83 El Nino, there may well have been weak La Nina conditions that year that were not severe enough to warrant official classification.

Experience with the ocean data provides numerous insights regarding the ensemble method. The technique can clearly help in discerning patterns in physical raw data, but the application process may not be completely straightforward. Researchers need to remain alert to biases that may affect the records making up the ensembles being compared, as with the placement of new buoys. Also, since the El Nino pattern was originally identified through other means, it is not clear that the ensemble of records machinery provides any new insight in this case. Nevertheless, as a test of the technique in a physically motivated domain the results are encouraging.

6.2.3 Discussion of the Ensemble of Records Approach

The experiments described in this chapter merely begin to examine what can be done with data organized as ensembles of records. Once ensembles are expressed in $\mathcal{F}$-space, virtually any machine learning technique can be adopted. For example, any two vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ in $\mathcal{F}$ define a planar classifier based upon $\text{sign}\,(\mathbf{f}_{new} \cdot (\mathbf{f}_1 - \mathbf{f}_2))$. Collections of such simple classifiers can be used to do boosting and create other large-margin classifiers. We expect that other machine learning technology can be similarly applied.

This work is far from the first to look at descriptions of data other than the canonical feature-value representation. Researchers in case-based reasoning often adopt complex or unusual case descriptions [46]. For example, Branting looks at legal cases represented as graphs [17]. Additionally, the field of reinforcement learning may be thought of as employing data in a nonstandard format [44]. While dealing with nonstandard data representations, in general work in these fields does not focus on the issue, and naturally the specifics differ from those presented here.

The work described herein owes a debt to the field of text and information retrieval, which makes heavy use of the cosine metric [66]. In one sense, our approach can be seen as an extension of that work, with an ensemble of records analogous to a bag of words. This means that advances in text retrieval may lead to insights applicable to ensembles of records. There is also some similarity between this work and multiple instance learning [51]. The latter uses collections that can contain many elements irrelevant to the target concept, whereas we assume that each record in an ensemble contributes to the overall identity of the whole.

This section has presented a fresh approach to looking at a class of raw data. We describe algorithms for handling data that can be organized in an *ensemble of records* paradigm. Apart from its performance on specific data sets, this approach provides the machine learning community with a novel viewpoint for looking at raw data. Evidence of a need for more such viewpoints exists: although it is not hard to come up with domains structured as ensembles of records, it is difficult to find existing data sets organized in this manner. By and large, existing data sets are already processed into formats that fit existing paradigms. A principal contribution of this work, therefore, is the expansion of the research viewpoint to include a new paradigm.

6.3 Machine Learning for Image Retrieval

Connections between machine learning and image processing go both ways. Yet only a few researchers have looked at the use of machine learning for image retrieval. For example, Tieu and Viola applied boosting as a method of feature selection [75]. Such techniques could stand more widespread application. This section looks at some experiments designed to examine the success of boosting in conjunction with Stairs. We will take a closer look at classification as the prototypical machine learning task, before examining the role boosting can play.

6.3.1 Image Classification

In the context of classification within large image libraries, there exists a potentially enormous number of categories that might be detected. By considering them one at a time one can reduce the problem to a two-class problem, detecting the presence or absence of a particular category. In nearly all cases, the population of the two classes of interest will be highly skewed, with the negative examples making up the vast majority of the population. The challenge lies in identifying the relatively few positive cases. Often the positive cases will be diffuse, with negative examples interspersed between them. This makes the classification problem particularly difficult because the image space cannot be neatly partitioned. Furthermore, one must guard against overfitted solutions, where each positive example is surrounded by a small bubble of positive space but no broad generalization occurs.

Classification as developed in this chapter differs somewhat from the use of a classification test to evaluate retrieval algorithms in Chapter 3. There, we dealt with a limited number of images from a relatively small number of mutually exclusive categories. The different retrieval algorithms were rated essentially on their ability to separate the internal representations of the categories. Here, the goal is to achieve the highest accuracy possible in discriminating between members and non-members of the category. ROC curves (described in Section 5.3.1) are often used to express the range of possible performance, but for consistency with previous chapters we will continue to show precision vs. recall. (As mentioned previously, the two curves are closely related, and good results on one imply good results on the other.) Classification in this context is thus more closely related to retrieval, with the difference that the classifier typically receives a set of example images rather than a single query. Additionally, the classifier must receive a set of negative examples (non-target images). These will assist it in choosing the best boundary between areas in image space likely to contain positive examples and those unlikely to contain them.

Using more than one example image seems like it should naturally result in improved performance, but achieving this is not as straightforward as it might appear. Naïvely relying on the distance to the nearest known positive example doesn't do much better than querying on a single image of median quality. Consider the class of images depicting sunrises and sunsets, long considered a simple category to identify. Figure 6.4 shows the recall as a function of number of images retrieved for a collection of sun images, and for the distance to the nearest positive example. Some single images do noticeably better than the multiple-image curve. Presumably, the sunset images are scattered in image space with the best examples near the center of the distribution. The multiple-image curve suffers in comparison with the best single-image curves because it values a match with a sunset image on the periphery of the distribution just as highly as a match with a sunset image that is close to the center.

One way to take advantage of multiple examples is to run a cross-validation study on the training set, trying to identify the best example image or images. This step can potentially increase the recall rate to equal or exceed that of the best images, by identifying the most predictive examplars. For example, classes that form a multimodal distribution in image space will not be modeled well by a single example image, but a few well-positioned images might model the distribution quite well. We employ a simple greedy methodology to find suitable images. Our algorithm choosing images one at a time, adding the single image that most improves the area under the precision vs. recall curve on the training data. The algorithm terminates when adding an additional image no longer increases the area under the curve. Although we employ this selection mechanism only for the positive examples of the target class, our approach recalls other work reporting high-quality classification using a few exemplars of a set [2, 1].

It turns out that a simple normalization step can also reach performance levels similar to those achieved by the best example images while retaining the ability of an image population to closely model the distribution of the positive images. To see why this is so, first note that particular images vary greatly in their mean similarity to other images in the collection. Figure 6.5a shows this variation graphically for ten sunset images. The shapes of the curves are similar, but the level varies greatly. This means that for a given similarity score, some curves show only a few outliers above that level, while for other curves most of the images lie above it. In a nearest-neighbor setting, the images with the highest curves will tend to dominate, in the sense that the majority of the nearest neighbors found will be for these images. At the same time, the higher similarity of these nearest neighbors means less in a statistical sense: their high similarity may less significant, relative to the mean similarity for the higher curve, than an outlier on a lower curve that has a slightly lower absolute score. The obvious way to fix this problem is to adjust all the curves so that their similarity scores are comparable. Figure 6.5b shows the ten curves from Figure 6.5a with their mean value subtracted off. All the curves now appear highly similar, and the outliers in similarity stand out. As demonstrated in Section 6.3.3, using the normalized similarity yields great improvements in classification accuracy.

6.3.2 Boosting

Having defined the classification problem from a machine learning perspective, we can now look at ways in which tools from that field may be applied. Boosting has been extensively explored by the machine learning community with great success [67, 30, 68, 31]. Because the results in this field are less well known than they should be by researchers in other fields, a short summary of the technique appears here. The basic idea is that when faced with a machine learning task, such as a classification problem, an algorithm that displays marginal success (accuracy slightly better than chance) can be "boosted" into an algorithm that performs much better than this. The secret lies in training the marginal algorithm to solve a collection of related problems, each focusing on a different aspect of the original. When faced with a new problem to solve, the booster relies on the collective input of all the collected solvers, which as a group show more discrimination than any single member of the group. Figure 6.6 summarizes one of the most popular boosting algorithms, AdaBoost [30], for the case of a two-class system (adapted from Friedman, et. al. [31]). Two-class classifiers can be easily generalized to multi-class classifiers if necessary [26].

AdaBoost and boosting algorithms in general require a base learning algorithm, often referred to as a *weak learner*, that can classify any set of weighted instances with better than 50% accuracy. With a two-class system, such weak learners are not hard to develop: nearly any division of the space of possible instances will do. Although the theoretical results place only weak requirements on the base algorithm, empirical experience suggests that more powerful classifiers tend to work better.

A Planar Base Classifier

To develop a base classifier for images, we dispense entirely with distance relations and work directly via planar partitions of $\mathcal{F}$-space. Because each image in $\mathcal{F}$-space is represented by a normalized vector, the space of images may be divided by a plane passing through the origin. Although a novel approach in the present context, this setup shares a similarity of spirit with traditional uses of boosting algorithms, which are often illustrated with planar classifiers. Linear combinations of planar classifiers can divide $\mathcal{F}$-space in complex and meaningful ways, as we shall demonstrate.

A simple algorithm yields a planar classifier with the majority of the positive examples on one side and the majority of the negative examples on the other. Each example image has a corresponding vector in $\mathcal{F}$-space. The algorithm takes the weighted vector sum of the positive and negative examples separately. The hyperplane located equidistant from these two "center of mass" vectors will form the classifier's boundary surface. If the two center of mass vectors are normalized and added together, they yield a vector in the plane. Subtracting all components in this direction from one of the original vectors yields a vector normal to the plane. Images can be quickly classified by checking the sign of their dot product with this normal vector. (Note that if spread matrices $\mathbf{T}$ are used (see Section 3.2.3), all processing should take place after the spreading operation.)

$$\mathbf{v}_+ = \frac{1}{n_+}\sum_{i=1}^{n_+}(w_{\mathbf{f}_i^+})(\mathbf{T}\mathbf{f}_i^+) \tag{6.6}$$

$$\mathbf{v}_- = \frac{1}{n_-}\sum_{i=1}^{n_-}(w_{\mathbf{f}_i^-})(\mathbf{T}\mathbf{f}_i^-) \tag{6.7}$$

$$\mathbf{v}_\perp = \mathbf{v}_+ - \frac{\mathbf{v}_- \cdot (\mathbf{v}_+ + \mathbf{v}_-)}{\|\mathbf{v}_+ + \mathbf{v}_-\|} \tag{6.8}$$

$$C(\mathbf{f}) = \begin{cases} 1 & \text{if } (\mathbf{T}\mathbf{f}) \cdot \mathbf{v}_\perp > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

If the positive and negative examples are well separated, then the hyperplane described above will place most of the positives on one side and most of the negatives on the other. However, sometimes after a number of rounds of boosting the weighted examples become so interspersed that the distribution of positive and negative is nearly equal on both sides, and may even be the reverse of the expected direction. In this case we simply flip the sign of the classifier in order to generate the greater than 50% weighted accuracy required for boosting. Typically after several rounds of close divisions, the instance weightings shift around enough that better divisions can again be found.

A Conic Base Classifier

A trivial change to the algorithm uses conic surfaces of division instead of planes to yield a better separation in the closely interspersed cases. Implementing the variation entails comparing the dot product with the normal vector $\mathbf{v}_\perp$ to a threshold, rather than just checking its sign. The best threshold to use can be found by computing the dot product with all training vectors and selecting the threshold that yields the highest weighted accuracy. Usually the best conic division is substantially better than the planar division; it is guaranteed to be at least as good because the plane is one possible conic solution, corresponding to a threshold of zero. Although theoretical results on boosting do not predict that the quality of the base classifier should make a difference, empirical results have shown that it does tend to make a difference [31].

6.3.3 Classification Results

We present results for a classic two-category (positive/negative) classification task. The target category comprises 200 images of sunrises and sunsets, traditionally considered an easy class to identify. The remaining 19,900 images in the collection form the negative examples. All tests use a 5×2 cross validation methodology, with both the positive and negative examples equally split between the two folds.

Table 6.1 summarizes the results, giving statistics on the area under the precision vs. recall graph for various algorithms. The scores for nearest-neighbor algorithms appear first. Using the raw Stairs similarity scores does rather poorly, for the reasons described in Section 6.3.1. Color histograms also do poorly, which is somewhat surprising in that sunsets are primarily identified through their distinctive colors. However, histograms have previously been shown to have poor discriminatory power as library size increases [40], so this result may be understood on those terms. Correlograms and normalized Stairs both do well, although correlograms appear to have a slight edge.

The subset selection algorithms, which use only the best exemplars of a class as chosen by the greedy algorithm described in Section 6.3.1, do surprisingly well compared to the nearest-neighbor methods. The area under the precision vs. recall curve increases significantly for all similarity metrics (histogram, correlogram, and Stairs). However, the results using normalized Stairs distances do not increase as much, so that the effects of normalization are statistically insignificant when combined with subset selection.

The boosted algorithms also show strong performance. As expected, the boosted conic classifier does better than the boosted planar classifier. Both outperform the best unboosted Stairs algorithm (Stairs Select), although the difference does not achieve statistical significance for the planar algorithm. Furthermore, the boosted conic Stairs shows the highest mean area of any method (although again the statistical comparison with Correlogram Select is inconclusive).

A look at the averaged graphs of precision vs. recall reveals a bit more about the relative performance of the algorithms. The nearest neighbor algorithms tend to drop fairly quickly to low precision values, from which they slowly decline further. The subset selection algorithms, on the other hand, tend to start at a higher precision at the low recall end, declining steadily thereafter. The two best algorithms, Boosted Conic Stairs and Correlogram Select, show fairly similar curves, varying by no more than about 10% precision either way over the entire graph. The only significant difference appears at the lowest recall values: the boosted algorithm maintains 100% precision for a short span. This indicates that the top-ranked two or three images for this algorithm are particularly reliable.

6.3.4 Lessons from Classification Experiments

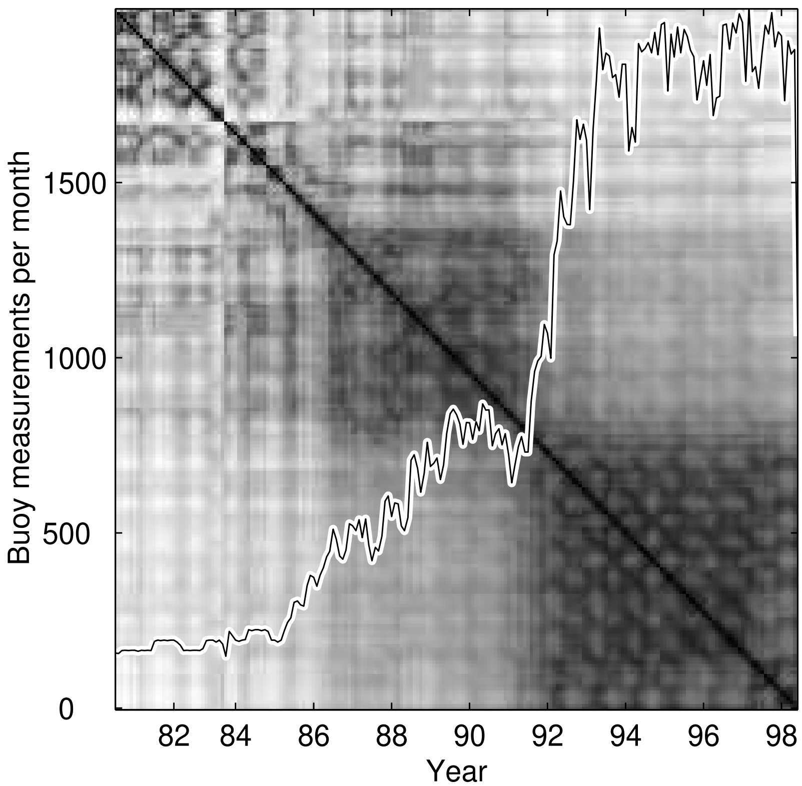
This section investigates whether imported techniques from machine learning research can improve the analysis and comparison of images. The answer is clearly yes, so long as the problems one is trying to solve are those for which machine learning methods are designed. Boosting

Table 6.1: Area under the recall-precision curve for various classification techniques on sunrise/sunset images (average over 10 folds).

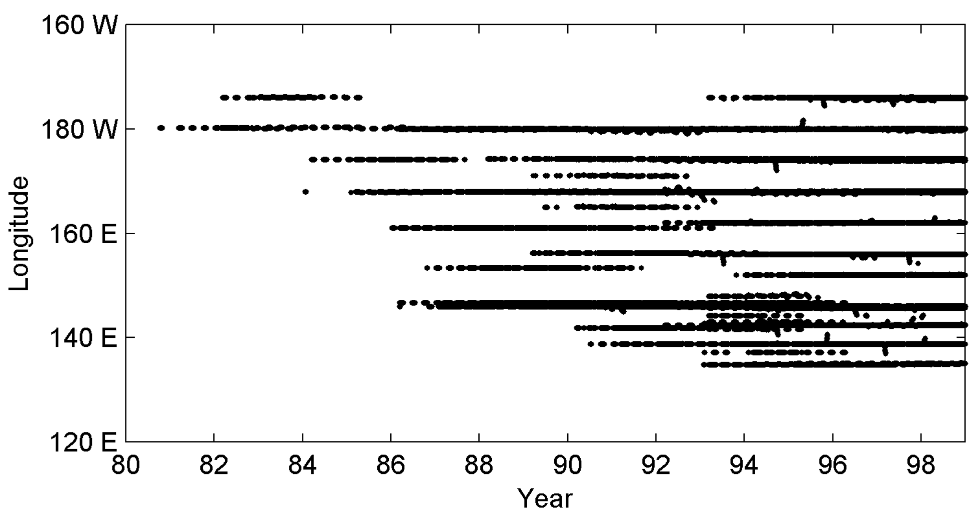
Algorithm	Area Under Curve (%)
Histogram Nearest Neighbor	5.35 ± 0.56
Correlogram Nearest Neighbor	19.70 ± 0.74
Stairs Nearest Neighbor	4.02 ± 0.14
Normalized Stairs Nearest Neighbor	16.98 ± 1.13
Histogram Select	15.44 ± 2.36
Correlogram Select	35.27 ± 3.08
Stairs Select	26.24 ± 2.22
Normalized Stairs Select	25.71 ± 1.34
Boosted Planar Stairs	26.85 ± 2.66
Boosted Conic Stairs	37.42 ± 2.45

improves on existing methods of classification for positive/negative style categories. (It presumably could also improve in multi-category classification, although this capability appears less useful in the context of image databases.) Unfortunately, it cannot be applied in many retrieval situations, where the user may only be able to provide a single example of the desired image category. Only as additional category examples are found could boosting play a role, as a sort of feedback mechanism used to refine the user's search. The most likely application for the sort of classification demonstrated in this section is in some sort of automated system for annotating newly acquired images. This would allow new additions to a library to be placed automatically into previously established classes, which could perhaps be accessed by users via keywords. Such a system may be a long way off, but binary classification algorithms like the ones demonstrated here form viable building blocks.

This section has looked at boosting only in conjunction with the Stairs framework. It is interesting to speculate whether comparable or even greater increases in performance could be achieved via boosting within a different image comparison paradigm, for example with correlograms. The answer may be yes, but the different structure of the correlogram statistic as compared with the Stairs $\mathcal{F}$-representation requires that the base mechanism differ in the two cases. For example, one could build decision trees based upon the correlogram values, and use these as the base algorithm in a boosting setup. However, even if such experiments reveal differences in performance between the paradigms, it would still remain unclear whether this resulted from the innate properties of the image space, or from the cleverness (or lack thereof) of the conversion to a boostable system. The important issue is not the underlying image description, but that boosting has been shown to significantly improve classification using at least one such description.



(a)



(b)

Figure 6.1: Addition of buoys over time.
In (a), the matrix of distance scores between months (represented by background shading) shows close correlation with the number of buoy measurements taken each month (foreground curve). In (b), the longitude of measurements is plotted versus time, revealing a westward trend over time.

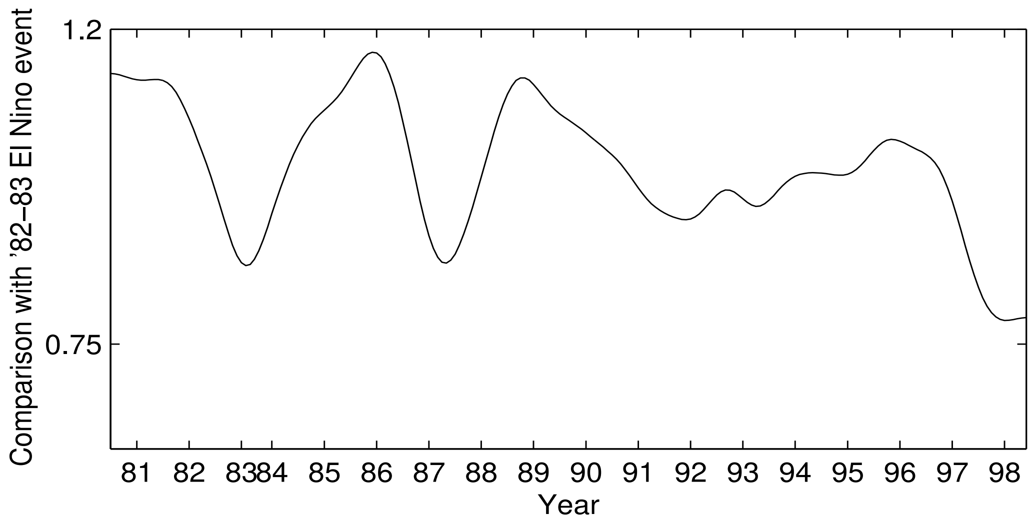


Figure 6.2: Comparison of month-by-month data with 1982-83 El Nino period (smoothed). Local minima reveal other El Nino events in 1986-87, 1991-92, and 1997-98.

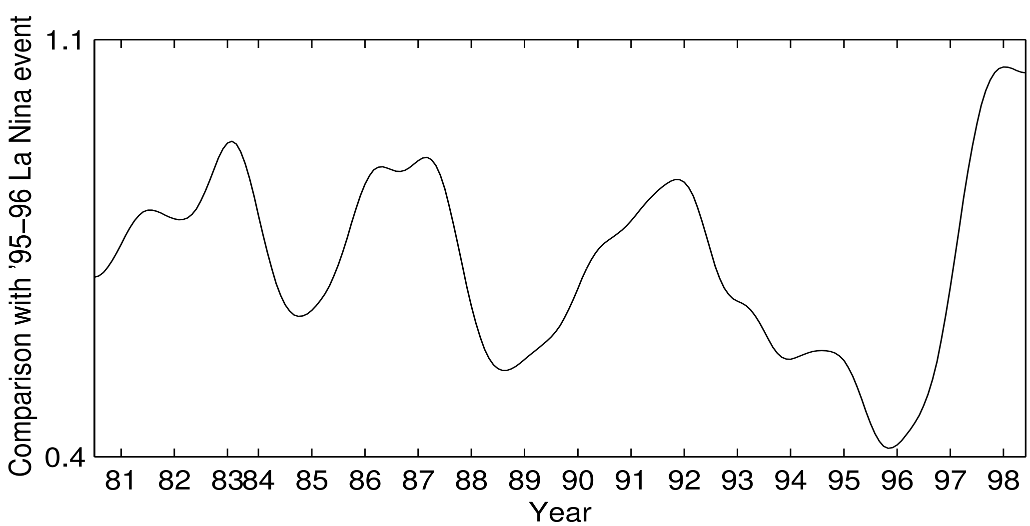


Figure 6.3: Comparison of month-by-month data with 1995-96 La Nina period (smoothed). Local minima reveal another La Nina in 1988-89 and a spurious but weaker signal in 1984-85.

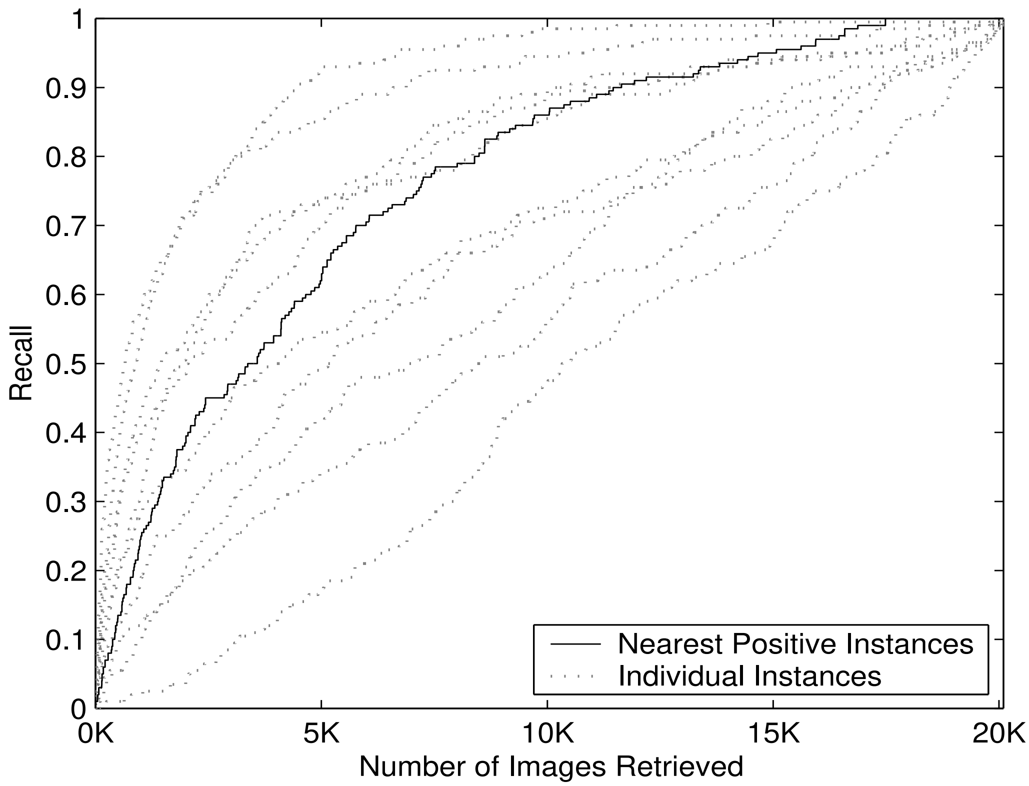


Figure 6.4: Recall vs. number of images retrieved for selected sunset images.

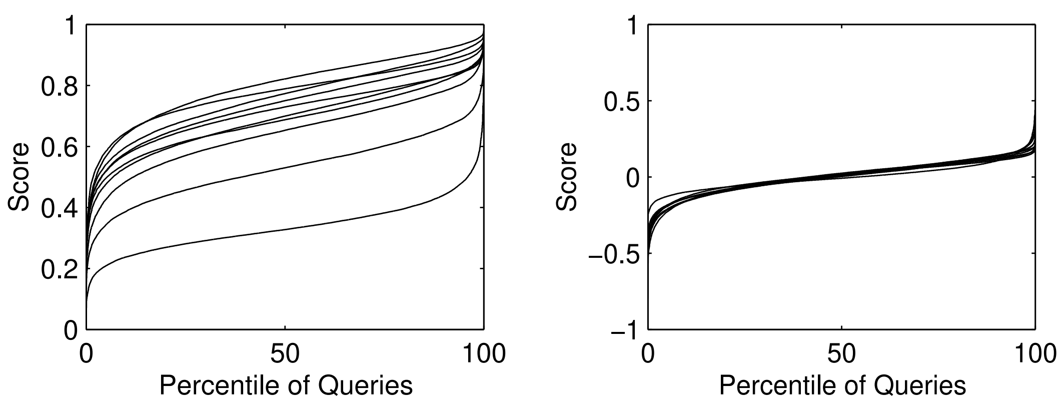


Figure 6.5: Sorted similarity curves for ten sunset images.
(a) Raw. (b) Normalized.

Input: Training set $\{x_1, x_2, ..., x_n\} \subset X$, corresponding classes $y_1, y_2, ..., y_n \in \{-1, 1\}$, and a base learning algorithm. Given x_i, y_i, and a set of weights w_i, $i = 1, ..., N$, the base learner must be able to output a classifier function $f(x)$ from the space of possible instances X onto the set of output classes $\{-1, 1\}$.

1. Assign each element of the training set an initial weight $w_i = \frac{1}{n}$, for $i = 1, 2, ..., n$.

2. Repeat for $j = 1, 2, ..., m$:

   (a) Use the base algorithm to learn a classifier $f_j(x)$ using weights w_i on the training data.

   (b) Count misclassifications on the training set:
   $$z_i = \begin{cases} 0 & y_i = f_j(x_i) \\ 1 & y_i \neq f_j(x_i) \end{cases}.$$

   (c) Compute the weighted error: $\varepsilon_j = \sum_{i=1}^{n} w_i z_i$.

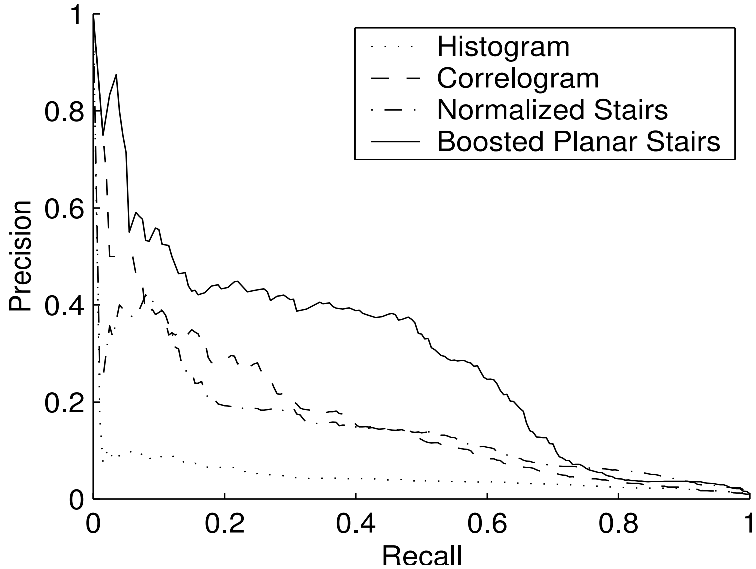
   (d) Set $c_j = \log\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$

   (e) Update the weights: $w_i \leftarrow w_i e^{z_i}$, $i = 1, 2, ..., n$.

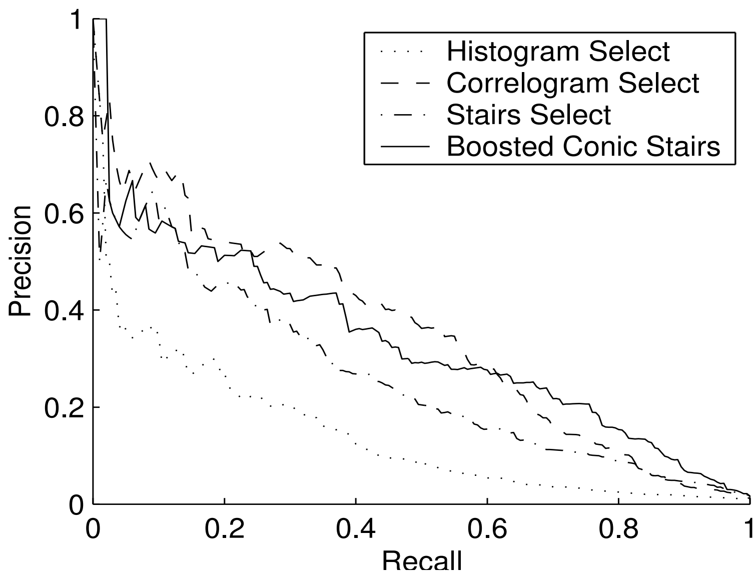
   (f) Renormalize the weights so that $\sum_{i=1}^{n} w_i = 1$.

3. Classify new instances according to **sign** $\left(\sum_{j=1}^{m} c_j f_j(x)\right)$.

Figure 6.6: The AdaBoost algorithm.



(a)



(b)

Figure 6.7: Precision vs. recall for selected classification algorithms.

Chapter 7

Conclusion

This dissertation has considered the challenge of using computers to perform automated tasks of image understanding. Specifically, it has looked at the comparison of images for retrieval and classification, with an eye toward building working systems. In the process, it has developed a number of new algorithms and related evaluation frameworks.

The development and analysis of the Stairs framework forms the primary contribution of this work. Careful analysis and comparative evaluation of existing algorithms such as histograms and correlograms make up a secondary contribution. Additional original work presented here includes a novel mid- to high-level segmentation scheme, the development of new evaluation techniques using altered-image queries, and the further investigation of potential ties between research in machine learning and computer vision.

The Stairs framework represents an implementation motivated by the area-matching assumption, as formulated and advanced in this work. This assumption holds that two images are visually similar to the extent that they are composed of equal areas of visually similar materials. By design Stairs allows the measurement of both the quantity and quality of matches between image regions, forming a final similarity score based upon both factors. The results of the tests presented here tend to vindicate the area-matching assumption, but only to a limited degree. While Stairs is competitive in most of the tests with other image similarity metrics on retrieval and classification tasks, it by no means dominates them. Correlograms, in particular, provide a strong challenge and in several cases outperform Stairs. Furthermore, even when the area-matching techniques prove strongest, such as with the work on partial-image retrieval, the overall results are poor enough that one cannot help expecting more.

Because correlograms do not perform strict area matching, their conspicuous success raises vital questions. Do the non-area-matching aspects of correlograms help or hinder their performance? One possibility is that they are helpful, allowing the algorithm to match images containing similar objects at different scales. Alternately, the explanation may lie in correlograms' unique description of texture, which none of the Stairs implementations match adequately. A choice between these potential explanations may not be reached without substantial further research, and perhaps would not reveal much of value in any case.

The contrast between a statistic like the correlogram and something like Stairs is that the correlogram, while effective, leaves no obvious room to grow. Stairs provides a framework in which to add richer and more abstract features that nudge the overall form of similarity implemented towards more human-like judgments. All current automated comparison metrics are based upon primitive image features that lie below the level of human comparisons. As a theory, the area-matching assumption has merit because it codifies a type of visual similarity that correlates with a certain concrete level of human judgments. Yet it leaves open the choice of features used to judge matches between image areas. Thus a clear area for research is the improvement of the languages and techniques used to describe and compare image areas, with the aim of increasing their richness and power. Improving the descriptions of image regions may not come easily. We would have preferred to include more complex descriptions of texture, as well as shapes, in our image token descriptors, but chose not to because the technology required to do so in a reliable and consistent way was not available. As is typical with image processing applications, one aspect of a problem can affect many others, and progress in many can be blocked by snags in a few. We expect that with time and continued research on a number of fronts, the descriptions used within the Stairs framework can grow richer, and that the overall performance possible will improve as a result.

Many people view image retrieval as a subfield of information retrieval, despite the fact that research on information retrieval has been dominated by the retrieval of text. Indisputably, image retrieval is a younger field than text retrieval, with its own issues and challenges. The computing power necessary to perform sophisticated processing of images was not available until fairly recently, whereas many basic text retrieval algorithms were developed decades ago [66]. Researchers in text retrieval regularly attend conferences where the top algorithms can be compared on identical problems; such events have not yet developed in the field of image retrieval. Yet one important lesson may be drawn from experience with text: research progress comes in part from the interplay and competition between different systems employing different methodologies. Much research on sophisticated semantic processing has failed to yield fruit in terms of implemented systems, because in their complexity they were not as reliable as simple statistical methods. Nevertheless, their development has been important as a foil to the statistical approaches. In this sense, the development of Stairs represents a major contribution simply because it establishes and investigates a new and different approach to image retrieval. While the performance of Stairs may not match that of correlograms in every case, its value lies in the fact that it can achieve comparable performance through a very different route. Future researchers, taking heed of the lessons embodied in a variety of previous approaches, will have better guidance than if they followed only a single star.

The best way to retrieve images from a large database is a thorny issue. As we have seen, the best definition of the problem is not always clear. Nevertheless throughout this thesis we have striven to include thoughtful, carefully designed evaluations that can shed light on the strengths and weaknesses of algorithms that we and others have designed. The results are not always pretty, but only through unflinching examination of the results can true understanding of the problem be achieved.

Figure 7.1: A sunset.

Bibliography

[1] D. W. Aha. *A Framework for Instance-Based Learning Algorithms: Mathematical, Empirical, and Psychological Evaluations*. Ph.D. thesis, University of California, Irvine, 1990. Technical Report 90-42.

[2] D. W. Aha. Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[4] N. M. Allinson and A. W. Ellis. Face recognition: Combining cognitive psychology and image engineering. *Electronics and Communication Engineering Journal*, 4(5):291–300, October 1992.

[5] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and semi-automatic methods for image annotation and retrieval in QBIC. In W. Niblack and R. C. Jain, editors, *Storage and Retrieval for Image and Video Databases III*. SPIE – The International Society for Optical Engineering, 1995.

[6] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In A. R. Hanson and E. M. Riseman, editors, *Computer Vision Systems*, pages 3–26, New York, 1978. Academic Press.

[7] S. D. Bay. UCI KDD archive, 1999. ⟨`http://kdd.ics.uci.edu`⟩.

[8] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Recognition of images in large databases using a learning framework. Technical Report 97-939, University of California, Berkeley, CA, 1997.

[9] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using the expectation-maximization algorithm and its application to content-based image retrieval. In *ICCV98*, pages 675–682, 1998.

[10] S. Belongie and J. Malik. Finding boundaries in natural images: A new method using point descriptors and area completion. In *ECCV98*, 1998.

[11] A. P. Berman and L. G. Shapiro. Efficient content-based retrieval: Experimental results. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1999.

[12] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1999. ⟨`http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`⟩.

[13] J. S. De Bonet and P. Viola. Structure driven image database retrieval. *Advances in Neural Information Processing*, 10, 1997.

[14] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 648–655, 1998.

[15] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision*, pages 377–384, 1999.

[16] Y. Boykov, O. Veksler, and R. Zabih. Energy minimization with discontinuities. *International Journal of Computer Vision*, in submission.

[17] L. K. Branting. *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*. Ph.D. thesis, University of Texas, Austin, TX, 1991.

[18] T. Caelli and D. Reye. On the classification of image regions by colour, texture, and shape. *Pattern Recognition*, 26(4):461–470, 1993.

[19] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001. (in review).

[20] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of the Third International Conference on Visual Information Systems*, 1999.

[21] S. Chang and A. Hsu. Image information systems: Where do we go from here? *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431–441, October 1992.

[22] S. Cohen. *Finding Color and Shape Patterns in Images*. Ph.D. thesis, Stanford University, May 1999.

[23] Y. Deng and B. S. Manjunath. An efficient low-dimensional color indexing scheme for region-based image retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3017–20, 1999.

[24] Y. Deng, B. S. Manjunath, and H. Shin. Color image segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[25] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998. Revised December 30, 1997.

[26] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[27] P. Felzenschwab and D. Huttenlocher. Efficiently computing a good segmentation. *International Journal of Computer Vision*, in submission.

[28] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 98–104, 1998.

[29] H. Freeman. Boundary encoding and processing. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictures*, pages 241–266, New York, 1970. Academic Press.

[30] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[31] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998.

[32] U. Gargi and R. Kasturi. Image database querying using a multi-scale localized color representation. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1999.

[33] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[34] A. Goldberg. Andrew Goldberg's network optimization library. ⟨`www.star-lab.com/goldberg/soft.html`⟩.

[35] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

[36] A. Graham. *Kronecker Products and Matrix Calculus with Applications.* Ellis Horwood Ltd., Chichester, England, 1981.

[37] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, 1990.

[38] N. R. Howe. Percentile blobs for image similarity. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 78–83, Los Alamitos, CA, 1998. IEEE Computer Society.

[39] Nicholas R. Howe. Using artificial queries to evaluate image retrieval. In *Proceedings of the IEEE Computer Society Workshop on Content-Based Access of Image and Video Databases*, pages 5–9, Los Alamito, CA, 2000. IEEE Computer Society.

[40] J. Huang. *Color-Spatial Image Indexing and Applications.* Ph.D. thesis, Cornell University, August 1998.

[41] J. Huang, S. Ravi Kumar, M. Mitra, W. J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 762–768, Los Alamitos, CA, 1997. IEEE Computer Society.

[42] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:850–863, 1993.

[43] H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1998.

[44] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:257–285, 1996.

[45] P. M. Kelly and M. Cannon. Query by image example: the CANDID approach. In W. Niblack and R. C. Jain, editors, *Storage and Retrieval for Image and Video Databases III*. SPIE – The International Society for Optical Engineering, 1995.

[46] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[47] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102, 1989.

[48] P. Lipson, E. Grimson, and P. Sinha. Configuration based scene classification and image indexing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1997.

[49] L. Liu and S. Sclaroff. Index trees for efficient deformable shape-based retrieval. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, Hilton Head, SC, 2000.

[50] W. Y. Ma. *NETRA: A Toolbox for Navigating Large Image Databases*. Ph.D. thesis, University of California at Santa Barbara, 1997.

[51] O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 341–349, San Mateo, CA, July 1998. Morgan Kaufmann.

[52] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman, San Francisco, 1982.

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

[54] M. J. McPhaden, W. S. Kessler, and N. N. Soreide. El Nino theme page, 1999. ⟨`http://www.pmel.noaa.gov/toga-tao/el-nino/nino-home.html`⟩.

[55] B. Moghaddam, H. Biermann, and D. Margaritis. Defining image content with multiple regions-of-interest. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1999.

[56] D. Nickerson. History of the munsell color system and its scientific application. *Journal of the Optical Society of America*, 30(12):575–86, December 1940.

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

[58] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proceedings of Intelligent Vehicles*, October 1998.

[59] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. Technical report, M.I.T. Media Laboratory, 1993.

[60] J. Platt. Autoalbum: Clustering digital photographs using probabilistic model merging. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 96–100, 2000.

[61] A. L. Ratan, O. Maron, W. E. L. Grimson, and T. Lozano-Perez. A framework for learning query concepts in image classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 423–431. IEEE Computer Society, 1999.

[62] Research Laboratories of the International Printing Ink Corporation. *Color In Use*. The International Printing Ink Corporation, New York, 1935.

[63] H. Rowley, S. Baluja, and T Kanade. Rotation invariant neural network-based face detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 38–44, Santa Barbara, 1998.

[64] Y. Rui, T. S. Huang, and S. F. Chang. Image retrieval: Past, present and future. In *Proceedings of the International Symposium on Multimedia Information Processing*, December 1997.

[65] E. Saber and A. M. Tekalp. Integration of color, edge, shape, and texture features for automatic region-based image annotation and retrieval. *Journal of Electronic Imaging*, 7(3):684–700, 1998.

[66] G. Salton. *Automatic Text Processing – the Transformation, Analysis, and Retrieval of Information by Computer*. Addison Wesley, Reading, MA, 1989.

[67] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[68] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[69] L. Shapiro. Personal communication, 2000.

[70] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of the International Conference on Computer Vision*, San Juan, Puerto Rico, June 1997.

[71] J. R. Smith and C. S. Li. Image classification and querying using composite region templates. *Journal of Computer Vision and Image Understanding*, 75(1/2):86–98, July-August 1999.

[72] C. Stanfill and D. Waltz. Toward Memory-Based Reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

[73] M. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 381–392, San Jose, CA, 1995.

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

[75] K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume I, pages 228–235, 2000.

[76] University of California, Berkeley, CA. The danger of keywords. ⟨`http://dlp.CS.Berkeley.EDU/photos/blobworld/warning.html`⟩.

[77] E. M. Voorhees and D. K. Harman, editors. *The Eighth Text REtrieval Conference (TREC-8)*. Department of Commerce, National Institute of Standards and Technology, 1999.

[78] H. Wechsler. Texture analysis – a survey. *Signal Processing*, 2:271–282, 1980.

[79] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, November 1993.

[80] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods*. John Wiley & Sons, New York, 2 edition, 1982.