Integrating Color, Texture, and Geometry for Image Retrieval

Nicholas R. Howe Daniel P. Huttenlocher
Cornell University
Department of Computer Science
Ithaca, NY 14853
{nihowe,dph}@cs.cornell.edu

Abstract

This paper examines the problem of image retrieval from large, heterogeneous image databases. We present a technique that fulfills several needs identified by surveying recent research in the field. This technique fairly integrates a diverse and expandable set of image properties (for example, color, texture, and location) in a retrieval framework, and allows end-users substantial control over their use. We propose a novel set of evaluation methods in addition to applying established tests for image retrieval; our technique proves competitive with state-of-the-art methods in these tests and does better on certain tasks. Furthermore, it improves on many standard image retrieval algorithms by supporting queries based on subsections of images. For certain queries this capability significantly increases the relevance of the images retrieved, and further expands the user's control over the retrieval process.

1 Introduction

The rapid growth of digital image and video libraries has created a corresponding need for efficient tools to manage those libraries. As the volume of image information grows, the management tools must become increasingly reliable and autonomous. One key task of such tools is *image retrieval* (ImR), consisting of the selection of appropriate images from a library in response to user queries. Working toward algorithms that yield ever more relevant retrievals, researchers have turned to approaches that integrate diverse sources of information: work based upon color histograms [20, 7] has given way to approaches based upon both spatial layout and color [16, 11, 18, 13]; similarly methods based purely upon texture [5] have given way to work combining color, texture, and distributional cues [15, 2]. By now, the merits of drawing on different types of image features for ImR are firmly established.

Our work capitalizes on this trend, providing a framework for fairly and consistently integrating diverse image properties into a description amenable to fast, reliable retrieval. We advance previous work in ImR by using a diverse and expandable set of image properties, and furthermore by allowing explicit user control over their interaction through a simple interface. An important advantage of the flexibility offered by our approach is the ability to make region-based queries, as described below. Additionally, our work brings image retrieval a step closer to existing text retrieval methods by adopting the cosine metric used successfully in that field [19]. Section 2 of this paper describes the details of our algorithm, called Stairs.

Section 3 addresses the lack of standardized evaluation techniques for ImR. We propose several new tests designed to test retrieval under certain specific adverse conditions, facilitating comparison between different algorithms. Using these tests in addition to one applied by other research groups, we directly compare Stairs to two other approaches, one state-of-the-art and one chosen as a baseline. Our approach competes well on these evaluations, regularly outperforming the baseline and beating both methods on many tests.

In addition to its abilities in standard retrieval rasks, Stairs allows more flexible queries than many ImR algorithms. A few reseachers have investigated retrieval based upon parts of images only, or upon objects within an image [17, 12, 4, 15]. This allows the user to form more powerful queries, but typically involves a tradeoff between the query expressiveness and the amount of extra computation required. The Stairs algorithm can operate in a regional-query mode with only a moderate increase in computational overhead. In this mode, a part of the query image is matched against any similar parts of the library images, and the best matches are returned. Section 4 describes this capability and shows that it can dramatically improve the rank of relevant images for some queries.

We conclude in Section 5 with a discussion of Stairs in relation to other ongoing work in ImR. While Stairs currently consists of a standard ImR engine that performs both full-image and region-based queries, it has the potential to provide a basis for additional capabilities at the cutting edge of research.

2 The Stairs algorithm

The acronym Stairs refers to the Semantic-Token-Based Automatic Image Retrieval System developed in this paper. It builds up a description of an image from a collection of primitive image elements, such as small patches of color or local color relationships. Like the words making up a document, each such element or token carries little meaning by itself, but in combination with other similar elements can form a meaningful whole. Stairs knowingly draws on successful approaches to text retrieval by treating images in a similar manner.

In text, words are a clear choice as the primitive semantic token. We will refer to the image equivalent 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. Most of the results presented in this paper use tokens consisting of local image patches described in terms of color, texture, and location. We have experimented with several other potential token definitions, such as pairs of locally neighboring patches, and single patches with additional descriptive features. For most tasks, the simple definition performs well and requires less computational overhead.

It is worth emphasizing that the approach detailed in this paper applies regardless of the choice of primitive unit or its description. For example, 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. This would give users the ability to easily build their own similarity measures from scratch, based upon the image features of their choosing.

2.1 Token extraction

Although the procedure that extracts the primitive tokens is not an integral part of the Stairs algorithm, the steps used are recorded here for completeness. All images are scaled to approximately 25,000 pixels (in practice, 128 $\times$ 192 pixels). Processing of each image begins with the fast local segmentation algorithm developed by Felzenszwalb and Huttenlocher [6] to find locally similar patches. This segmentation will produce quite large regions if the image is smoothly shaded, as in the case of a clear sky. Because we wish to capture local properties of the image, a post-segmentation stage breaks up all regions above a threshold size into roughly equal, contiguous pieces. This process consistently results in a segmentation of the image into about 500 pieces. (The specific segmentation process appears unimportant, as experiments using a simple 16×32 grid segmentation produce comparable results.)

Once the basic units making up the image have been identified, they can be described by a feature vector capturing their significant properties. The current Stairs implementation generates values based upon color, texture, and spatial properties of each token: the mean color in HSV space, a crude texture indicator consisting of the median difference between individual pixels and the smoothed patch color, and the location of the patch in the XY plane. The XY position is specified relative to the center of the image, in units related to the image dimensions. In addition to these properties, the token is given a weight proportional to its area. Once feature vectors are stored for each token, the general Stairs algorithm can create a single representation describing the entire image.

2.2 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.

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. Our implementation uses 28 bins for the color of each segment, 3 bins for the amount of texture present, and 21 bins for the spatial location. Thus a discrete vector $\vec{m}$ in $\mathcal{M}$ representing an image token has $N_{\mathcal{M}} = 1764$ possible values. 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 $\vec{m} \in \mathcal{M}$, then they are represented by a single ordered pair whose weight is the sum of the individual weights of all the tokens.)

$$R_{\mathcal{M}}(I) = \left\{ (\vec{m}_1, w_1), (\vec{m}_2, w_2), \ldots, \left(\vec{m}_{n_I}, w_{n_I}\right) \right\} \quad (1)$$

$R_{\mathcal{M}}(I)$ is called 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 our implementation $\mathcal{F}$ is a 1764-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 correspondance 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, $R_{\mathcal{F}}(I)$:

$$R_{\mathcal{F}}(I) = \sum_{i=1}^{n_I} w_i F(\vec{m}_i) \tag{2}$$

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 $\vec{m}$ to $F(\vec{m})$ 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).

2.3 Comparing images

The framework we have set up is analogous by design to simple text retrieval systems [19]. The vectors in $R_{\mathcal{M}}(I)$ correspond to individual words, and $R_{\mathcal{F}}(I)$ corresponds to the word vector representing a text. Continuing the analogy, we compare two $\mathcal{F}$-representation using a cosine metric. Abbreviating $R_{\mathcal{F}}(I)$ as $\vec{f}_I$, we have

$$D_{\mathcal{F}}(I_1, I_2) = \cos^{-1}\left(\frac{\vec{f}_{I_1} \cdot \vec{f}_{I_2}}{\sqrt{\left(\vec{f}_{I_1} \cdot \vec{f}_{I_1}\right)\left(\vec{f}_{I_2} \cdot \vec{f}_{I_2}\right)}}\right) \tag{3}$$

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, 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 $\vec{m}_i \neq \vec{m}_j$ implies $F(\vec{m}_i) \cdot F(\vec{m}_j) = 0$ even if $\vec{m}_i$ and $\vec{m}_j$ are near matches. To address this issue, we define a similarity function $S(\vec{m}_1, \vec{m}_2)$ on vectors in $\mathcal{M}$, such that $S(\vec{m}_1, \vec{m}_2)$ 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.

$$D_{\mathcal{F}}(I_1, I_2) = \cos^{-1}\left(\frac{\vec{f}_{I_1}^T \mathbf{S} \vec{f}_{I_2}}{\sqrt{\left(\vec{f}_{I_1}^T \mathbf{S} \vec{f}_{I_1}\right)\left(\vec{f}_{I_2}^T \mathbf{S} \vec{f}_{I_2}\right)}}\right) \tag{4}$$

where $\mathbf{S}$ is the matrix representation of S in $\mathcal{F}$. We use a symmetric $\mathbf{S}$ that has a Cholesky factorization, $\mathbf{S} = \mathbf{T}^T\mathbf{T}$. This gives an alternate interpretation of Equation 4, as computing the cosine difference between two transformed vectors $\mathbf{T}\vec{f}_{I_1}$ and $\mathbf{T}\vec{f}_{I_2}$.

2.4 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:

$$\mathbf{T}_{HSV}(i,j) = (p_H)^{\Delta_H(i,j)}\,(p_S)^{\Delta_S(i,j)}\,(p_V)^{\Delta_V(i,j)} \tag{5}$$

where p_H, p_S, and $p_V \in [0,1]$ are user-tunable parameters controlling how much to value near matches. At the extremes, $p_H = 0$ indicates that the hue must match exactly, whereas $p_H = 1$ indicates that any hue will match equally well. (This is a potentially useful setting, allowing the system to ignore a feature deemed not useful. 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*.

2.5 Stairs in practice

A naïve implementation of the algorithm described above would be unwieldy, especially if the base image token used requires $\mathcal{M}$ to be large. Even with simple tokens, $\mathbf{S}$ requires about 25 MB of memory to instantiate, and computing $D_{\mathcal{F}}$ involves a large matrix multiplication. Furthermore, the $\mathcal{F}$-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 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 $\vec{f}_I\mathbf{S}$ is linear in the size of $\mathbf{S}$ instead of quadratic [8]. Essentially, the result can be found by repeated multiplications of the smaller $\mathbf{S}_{F_i}$ matrices. With these modifications, the Stairs technique can scale to handle more complex image token descriptions than those described here. We have successfully run the system in interactive time with $N_{\mathcal{M}} \sim 10^6$, an increase of three orders of magnitude.

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 4 reveals that this can be made quite efficient, since the vector $\vec{f}_{I_1}^T \mathbf{S}$ need be computed for the query image I_1 only once, and $\vec{f}_{I_2}^T \mathbf{S} \vec{f}_{I_2}$ 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, most queries on the 19,000-image test library can be resolved in less than two seconds on a desktop PC.

2.6 Search Pruning

Pruning of the search space relies upon a fast analysis in each dimension of $\mathcal{M}$ independently, and eliminates the majority of images from consideration without incorrectly overlooking any candidates. We quickly calculate an upper bound on the similarity of each library image to the query, and use this bound to order and prune the retrieval process. Once we have found a sufficient number of close matches, we can remove from consideration all images whose similarity is bounded away from the top scores. Empirically, the number of images that must be fully searched appears independent of library size for most queries. The dramatic gains achieved with our pruning technique resemble those recently reported for a different image retrieval system with pruning based upon the triangle inequality [2].

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. Our pruning technique depends heavily this fact. 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. For fixed $\mathbf{S}$, these operations compare in time and space to retrieval using color histograms. (If we wish to use multiple $\mathbf{S}$ matrices, we must either cache projections for each or calculate them online.) Once the bounds have been computed, in order to find the k most relevant images it often suffices to look at only $3k$ or $4k$ images in all.

3 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.

We address these factors as best we can in this paper. We use a commercially available library of 19,000 images from Corel, a source also used by other research groups [4, 17]. We directly compare our technique against two other methods: a version of color histograms [20] as a baseline for comparison, and our implementation of the color autocorrelogram technique proposed by Huang, *et. al.* [13]. The latter is a state-of-the-art technique that has itself undergone extensive testing [12].

In order to allow comparison with other previous work in image retrieval, we carry out a classification test that has been previously used to evaluate retrieval methods [1, 9]. However, 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 judgements of similarity and can be easily duplicated 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 [10].

3.1 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. If the top retrieval is of the same category as the query image, then the image is considered a correct classification; otherwise it is incorrect. Results of this test are shown in Table 1 by category, for two separate test sets; the first has been used by other groups [1, 9] and the second includes additional categories in the same vein. As expected, Stairs and correlograms perform better than histograms. 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.

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

Category	Stairs	Hist.	Corr.	Stairs (+)
Airshows	**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 1. Performance of ImR methods on two classification tests. (Top is original.)

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 1 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.

3.2 Artificial-query tests

Artificial queries offer great potential in testing specific aspects of retrieval performance. Areas of inquiry include invariance to assorted image transformations and ability to cope with limited information. There is always the potential that alterations to a particular image will result in true similarity to a different image. Luckily, in practice such interactions are rare enough that assessment using large numbers of images produces stable results, even if the set of query images changes.

We introduce three artificial-query tests here: the *Crop* test, the *Jumble* test, and the low contrast (*Low-Con*) test. *Crop* produces a query image by cropping 50% of the image area around the borders and blowing up the remaining central portion. It tests performance in retrieving a full image given a closeup. *Jumble* takes the original image and randomly reshuffles its parts on a 4×4 grid. It tests retrieval of images with similar content in different arrangements. *Low-Con* remaps the RGB intensity values of the image onto the range [0.1,0.9]. It tests retrieval of images given a query from a damaged or miscalibrated source, or from video under varying lighting conditions. Figure 1 shows an example of each type of query.

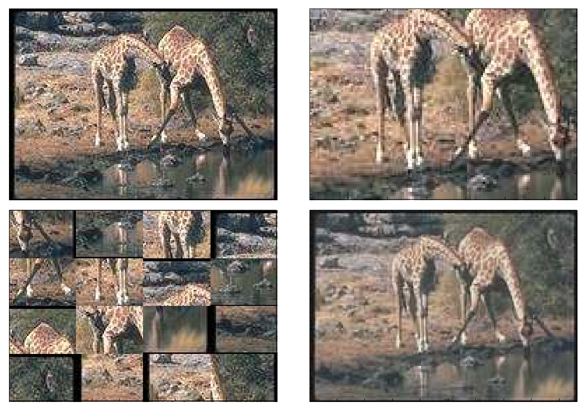

Figure 1. Examples of each type of artificial query. First row, l-r: Original image, Crop. Second row: Jumble, Low-Con.

Sorted plots of the ranks returned in artificial-query tests generally look flat over most of their range, with a sharp drop in the tail. In other words, most images are retrieved fairly reliably, but a few are not. Two numbers characterize this behavior: The median rank indicates how successfully most images are retrieved, while the mean rank expresses the severity of the worst cases. Table 2 reports these numbers for each of the retrieval techniques on each of the selected tests, using a randomly selected sample of 1000 images as queries. Two numbers are given for Stairs: one using a consistent **S** and one where **S** is chosen to be particularly appropriate for the task. The latter case simulates tasks where the main obstacles to retrieval are known. Not surprisingly, the tuned algorithm performs a bit better on the task (*Jumble*) where the untuned version has difficulty. The ability to tune Stairs to a particular task is powerful, but there are cases where the ideal setting is unknown. The untuned test provides a better indication of performance in such situations.

		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

Table 2. Target rank results in artificial-query tests. (Lower numbers are better.)

4 Queries on parts of images

Many researchers have noted the importance of sub-image queries [4, 15]. Often users are interested in only a portion of an image, perhaps a particular object in a scene containing many others. In such cases, a query based on the full image will return many false hits due to spurious matches with irrelevant areas of the scene. To solve this problem, an ImR system must retrieve images based upon a match of some region in the target image with a specified region of the query image. This process will be referred to as *region matching*, and the corresponding request a *region query*. Stairs supports a form of region matching as a special case of a more general capability: matching some image tokens more or less strictly than others. In this paradigm, a region query is formed by requiring a close match in the region of interest, while allowing the rest of the image to match anything.

Region matching can magnify the problem of image retrieval significantly, since for every image in the library there are a multitude of potential regions to match. Some early region matching schemes require searching over potential matches in each image, an approach that doesn't scale well as library size increases [14]. More recent work speeds up queries by pre-segmenting images into promising regions [4], while a few techniques actually perform a segmentation online in response to user queries [15].

4.1 Stairs region matching

Region matching in Stairs falls into the online category. The user's query explicitly defines a division of the image into subject and background regions. Stairs uses this division to divide $\mathcal{M}$-space into two subsets: $\mathcal{M}_S$, which contains tokens in the subject 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 camoflaged. In a sense, such cases are intrinsically more difficult.)

To avoid explicitly isolating the interesting portions of each target image, the distinctions between $\mathcal{M}_S$ and $\mathcal{M}_B$ can be rolled into Equation 4 by making the appropriate choice of $\mathbf{S}$. Consider two match matrices, $\mathbf{S}_S$ and $\mathbf{S}_B$, generated from different spread parameters p_{F_i}. $\mathbf{S}_S$ 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. These two matrices are combined as follows to form $\mathbf{S}$:

$$\mathbf{S}(i,j) = \begin{cases} \mathbf{S}_S(i,j) & \text{if}(m_i \in \mathcal{M}_S) \wedge (m_j \in \mathcal{M}_S) \\ \mathbf{S}_B(i,j) & \text{if}(m_i \in \mathcal{M}_B) \wedge (m_j \in \mathcal{M}_B) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The resulting $\mathbf{S}$ matrix compares potential subject features to the query subject features, and ignores the background features. With different choices of $\mathbf{S}_S$ 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.

Thus Stairs accomodates region 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 4 cannot be used. Even so, the entire region query computation for 19K images takes only 30 to 40 seconds on a 266 MHz Pentium II. This is competitive with times reported by other groups for region matching [3].

4.2 Evaluation of region matching

Tests of region matching techniques are even less standardized than those for full image matching. We present the results of two tests, one on a small domain (200 images of cars) and one on the full library.

The small domain, images of cars, is useful because the subject and background are easily defined, and the set of images may be divided into cars of different colors. Full image queries will sometimes retrieve cars of different color that happen to be on similar backgrounds. If the user is interested only in the cars themselves, then this behavior is undesirable. A simple test shows that region queries can significantly improve performance in this regard. Figure 2 shows mean recall/precision curves for a query on each of four categories: red, yellow, black, and white cars. In each case the region query is the top curve, particularly at the high-precision end.

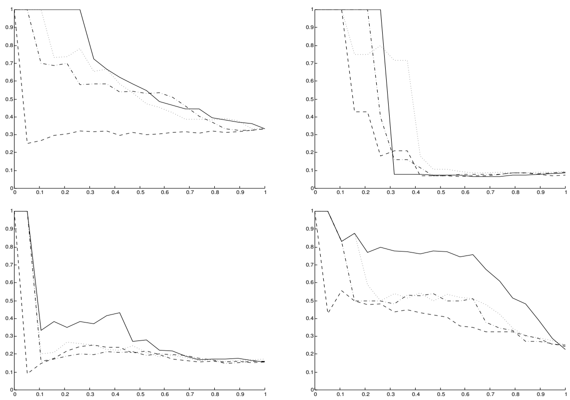

Figure 2. Precision vs. recall graphs for queries on color images of cars. (Solid line = region query; dashed = full image; dotted = correlogram; dashdot = histogram.)

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. We identified ten cases of this type and compared the images retrieved from a full image query to those from a targeted region query; Figure 3 shows the results from four. The median rank of the first relevant image retrieved was 5.5 for region queries, compared with 43, 90, and 59 respectively for regular Stairs, correlograms, and histograms. In a deployed system, this type of difference is probably enough to determine whether the first page of retrievals contains a relevant image. Thus region queries can significantly increase the relevance of the images retrieved, compared with any of the full-image approaches tested. (Note that while there is a region-query version of correlograms [12], we do not use it here because we are interested in comparing Stairs region queries to full-image methods.)

5 Conclusion

As evidenced by recent work in region matching, research in image retrieval is moving beyond simply searching a library for matches to a query image. Efforts are underway to provide users with more flexible and powerful query tools. Because of its encompassing architecture, Stairs may in the future prove useful as a basis for providing some of these higher-level tools. For example, Ratan *et. al.* report success in learning visual categories from examples [17]. Although their underlying architecture is quite different, it may be possible to perform a similar analysis on the representations used by Stairs. Carson *et. al.* take a different approach by pre-segmenting library images into regions of interest and facilitating sophisticated queries based on these regions [4]. Although the implicit region queries described in this paper already address some of the same issues, access to high-level segmentations would increase the functionality of the Stairs engine.

While rooted in simple primitives, Stairs combines multiple sources of information to create a remarkably complete description of an image. Objective evaluations show that this description can support effective image retrieval, performing at a level competitive with state-of-the-art ImR systems. Furthermore, the approach is flexible, allowing control both at the user level in the choice of spread parameters and at the design level in the choice of primitive image tokens and token descriptors. It is this flexibility which is the hallmark of the Stairs approach, and which suggests that it will be useful in developing new ImR tools in the future.

References

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

[2] 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.

[3] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. 1999. (in review).

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

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

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

[7] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

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

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

[10] N. Howe. Using artificial queries to evaluate image retrieval. Technical report, Cornell University, 2000. Submitted to CBAIVL '00.

[11] W. Hsu, T. Chua, and H. Pung. An integrated color-spatial approach to content-based image retrieval. In *The Third ACM International Multimedia Conference*, pages 305–313. Association for Computing Machinery, Inc., 1995.

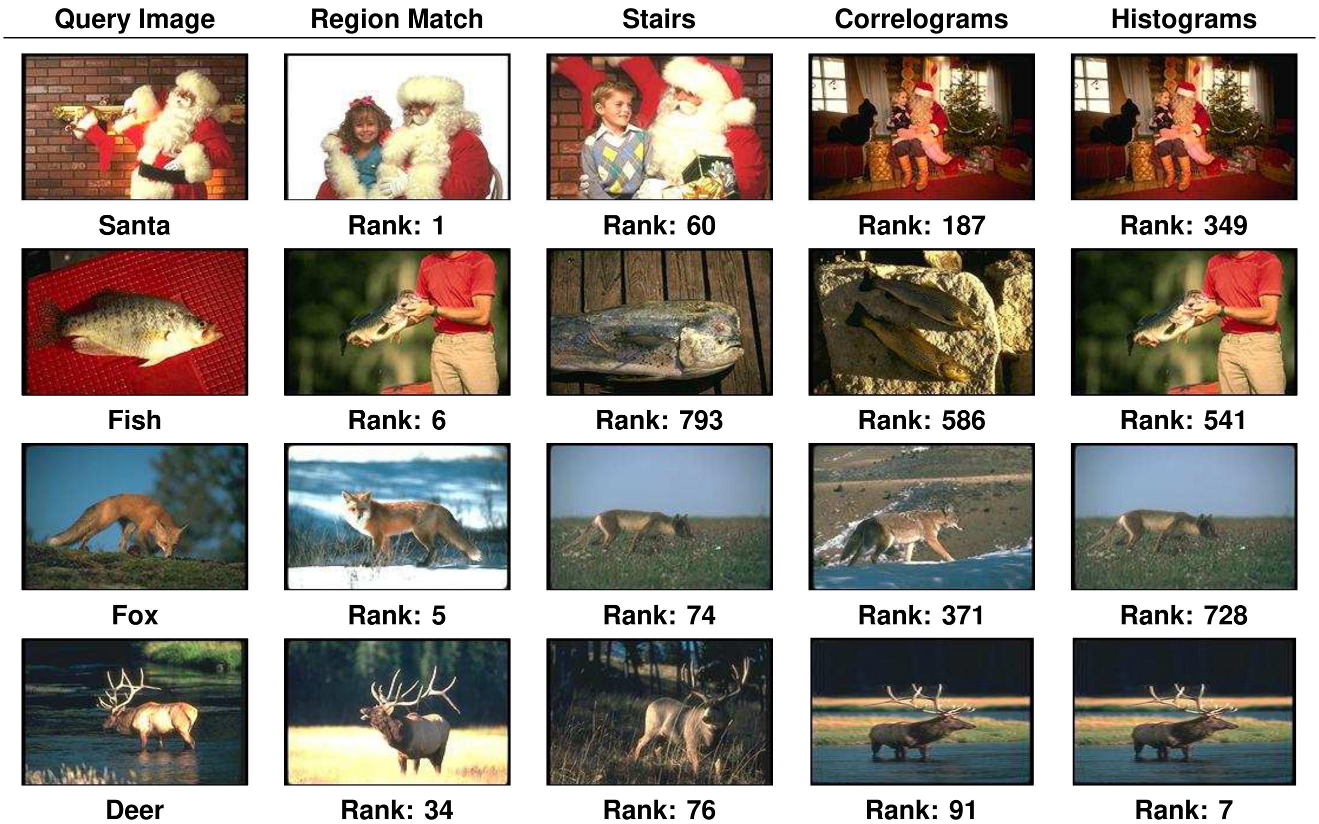

Figure 3. Region query results compared with full-image queries. Caption below pictures gives target category and rank of top relevant image.

[12] J. Huang. *Color-Spatial Image Indexing and Applications*. PhD thesis, Cornell University, August 1998.

[13] J. Huang, S. Ravi Kumar, M. Mitra, W. J. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1997.

[14] 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.

[15] 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.

[16] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *The Fourth ACM International Multimedia Conference*, pages 65–73, Boston, MA, November 1996. Association for Computing Machinery, Inc.

[17] A. L. Ratan, O. Maron, W. E. L. Grimson, and T. Lozano-Perez. A framework for learning query concepts in image classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 423–429, June 1999.

[18] R. Rickman and J. Stonham. Content-based image retrieval using color tuple histograms. In *SPIE Proceedings*, volume 2670, pages 2–7, February 1996.

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

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