Image Indexing Using Color Correlograms

Jing Huang* S Ravi Kumar† Mandar Mitra‡ Wei-Jing Zhu§ Ramin Zabih¶

Cornell University
Ithaca, NY 14853

Abstract

We define a new image feature called the color correlogram *and use it for image indexing and comparison. This feature distills the spatial correlation of colors, and is both effective and inexpensive for content-based image retrieval. The correlogram robustly tolerates large changes in appearance and shape caused by changes in viewing positions, camera zooms, etc. Experimental evidence suggests that this new feature outperforms not only the traditional color histogram method but also the recently proposed histogram refinement methods for image indexing/retrieval.*

1. Introduction

With the rapid proliferation of the internet and the world-wide-web, the amount of digital image data accessible to users has grown enormously. Image databases are becoming larger and more widespread, and there is a growing need for effective and efficient image retrieval (IR) systems.

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

For a retrieval system to be successful, the feature vector $f(\mathcal{I})$ for an image $\mathcal{I}$ should have certain desirable qualities: (i) $|f(\mathcal{I}) - f(\mathcal{I}')|$ should be large if and only if $\mathcal{I}$ and $\mathcal{I}'$ are not "similar", (ii) $f(\cdot)$ should be fast to compute, and (iii) $f(\mathcal{I})$ should be small in size.

Color histograms are commonly used as feature vectors for images [14, 3, 7, 9]. A color histogram describes the global color distribution in an image. It is easy to compute and is insensitive to small changes in viewing positions. However, it does not include any spatial information , and is therefore liable to false positives. This problem is especially acute for large databases. Moreover, the histogram is not robust to large appearance changes. For instance, the pairs of images shown in Figure 3 (photographs of the same scene taken from different viewpoints) are branded dissimilar by histogram methods. Recently several approaches have attempted to incorporate spatial information with color [13, 12, 10, 8]. Most of them divide the image into regions while the recent color coherent vector (CCV) method uses a histogram-refinement approach. CCVs are easy to compute and appear to perform much better than color histograms [8].

Our Approach. In this paper, we propose a new color feature for image indexing/retrieval called the *color correlogram.* The highlights of this feature are: (i) it includes the spatial correlation of colors, (ii) it can be used to describe the global distribution of local spatial correlation of colors; (iii) it is easy to compute, and (iv) the size of the feature is fairly small. Our experiments show that this new feature can outperform both the traditional histogram method and the recently proposed histogram refinement methods for image indexing/retrieval.

Informally, a color correlogram of an image is a table indexed by color pairs, where the k-th entry for $\langle i, j \rangle$ specifies the probability of finding a pixel of color j at a distance k from a pixel of color i in the image. Such an image feature turns out to be robust in tolerating large changes in appearance of the same scene caused by changes in viewing positions, changes in the background scene, partial occlusions, camera zoom that causes radical changes in shape, etc. (see Figure 3 for examples). We provide efficient algorithms to compute the correlogram.

*Supported by NSF grant ASC-8902827, and by DARPA under a contract monitored by ARL. huang@cs.cornell.edu

†Supported by ONR Young Investigator Award N00014-93-1-0590, NSF grant DMI-91157199, and CAREER grant CCR-9624552. ravi@cs.cornell.edu

‡Supported by NSF grant IRI-9300124. mitra@cs.cornell.edu

§Supported by DOE Grant DEFG02-89ER45405. wjzhu@msc.cornell.edu

¶Supported by DARPA under a contract monitored by ARL. rdz@cs.cornell.edu

We also investigate a different distance measure to compare feature vectors. The L_1 distance measure, used commonly to compare vectors, considers the absolute component-wise differences between vectors. The relative distance measure we use calculates relative differences instead and in most cases performs better than the absolute measure. The improvement is significant especially for histogram-based methods.

We conduct experiments using a large database of 14,554 images and evaluate our techniques using quantitative criteria. The objective nature of these measures enables us to fairly compare different methods.

Related Work. Several schemes for using spatial information about colors to improve upon the histogram method have been proposed recently. One common approach is to divide images into subregions and impose positional constraints on the image comparison (*image partitioning*). Another approach is to augment histograms with local spatial properties (*histogram refinement*).

Smith and Chang [12] partition an image into binary color sets. They first select all colors that are "sufficiently" present in a region. The colors for a region are represented by a binary color set that is computed using histogram back-projection [14]. The binary color sets and their location information constitute the feature. Stricker and Dimai [13] divide an image into five fixed overlapping regions and extract the first three color moments of each region to form a feature vector for the image. The storage requirements for this method are low. The use of overlapping regions makes the feature vectors relatively insensitive to small rotations or translations.

Pass and Zabih [8] use another approach. They partition histogram bins by the spatial coherence of pixels. A pixel is coherent if it is a part of some "sizable" similar-colored region, and incoherent otherwise. A color coherence vector (CCV) represents this classification for each color in the image. CCVs are fast to compute and appear to perform better than histograms. The notion of CCV is also extended in [8], by using additional feature(s) to further refine the CCV-refined histogram. One such extension uses the center of the image (the centermost 75% of the pixels are defined as the "center") as the additional feature. The enhanced CCV is called CCV with *successive refinement* (CCV/C) and performs better than CCV.

The color correlogram is neither an image partitioning method nor a histogram refinement method. Unlike purely local properties, such as pixel position, gradient direction, or purely global properties, such as color distribution, correlograms take into account the local color spatial correlation as well as the global distribution of this spatial correlation. While any scheme that is based on purely local properties is likely to be sensitive to large appearance changes, correlograms are more stable to these changes; while any scheme that is based on purely global properties is susceptible to false positive matches, correlograms prove to be effective for content-based image retrieval from a large image database.

2. The Correlogram

A color correlogram (henceforth correlogram) expresses how the spatial correlation of pairs of colors changes with distance (the term "correlogram" is adapted from spatial data analysis [15]). A color histogram (henceforth histogram) captures only the color distribution in an image and does not include any spatial correlation information.

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

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

Definitions. The *histogram* h of $\mathcal{I}$ is defined for $i \in [m]$ by

$$h_{c_i}(\mathcal{I}) \triangleq n^2 \cdot \Pr_{p \in \mathcal{I}}[p \in \mathcal{I}_{c_i}]. \qquad (1)$$

For any pixel in the image, $h_{c_i}(\mathcal{I})/n^2$ gives the probability that the color of the pixel is c_i.

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

$$\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) \triangleq \Pr_{p_1 \in \mathcal{I}_{c_i}, p_2 \in \mathcal{I}}[p_2 \in \mathcal{I}_{c_j} \mid |p_1 - p_2| = k]. \qquad (2)$$

Given any pixel of color c_i in the image, $\gamma^{(k)}_{c_i,c_j}$ gives the probability that a pixel at distance k away from the given pixel is of color c_j. Note that the size of the correlogram is $O(m^2 d)$. The *autocorrelogram* of $\mathcal{I}$ captures spatial correlation between identical colors only and is defined by

$$\alpha^{(k)}_c(\mathcal{I}) \triangleq \gamma^{(k)}_{c,c}(\mathcal{I}). \qquad (3)$$

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

While choosing d to define the correlogram, we need to address the following issue. A large d would result in expensive computation and large storage requirements. A small d might compromise the quality of the feature. We consider this tradeoff in section 5.

Example. Consider the simple case when $m = 2$ and $n = 8$. Two sample images are shown in Figure 1. The autocorrelograms corresponding to these two images are shown in Figure 2. The change of autocorrelation of the foreground color with distance is perceptibly different for these images. Note that it is difficult to distinguish between these two images using histograms or CCVs.

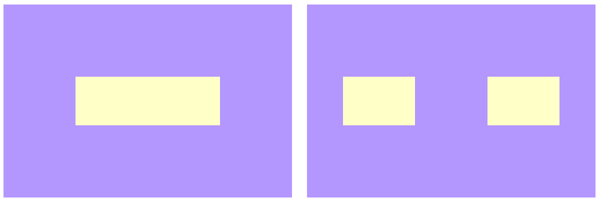

Figure 1. Sample images: image 1, image 2.

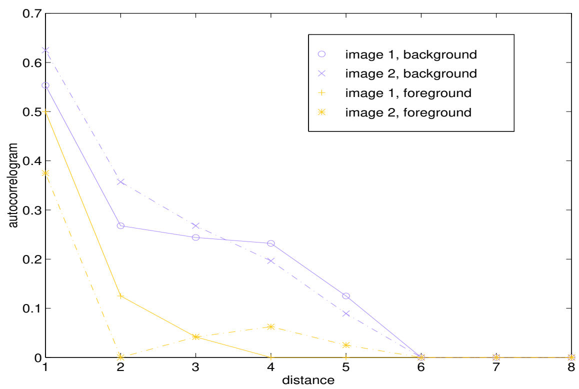

Figure 2. Autocorrelograms for images in Figure 1.

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

First, to compute the correlogram, it suffices to compute the following count (similar to the *cooccurrence matrix* defined in [5] for texture analysis of gray images)

$$\Gamma^{(k)}_{c_i,c_j}(\mathcal{I}) \triangleq \left|\{p_1 \in \mathcal{I}_{c_i}, p_2 \in \mathcal{I}_{c_j} \mid |p_1 - p_2| = k\}\right| \quad (4)$$

for, $\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) = \Gamma^{(k)}_{c_i,c_j}(\mathcal{I}) / (h_{c_i}(\mathcal{I}) \cdot 8k)$. The denominator is the total number of pixels at distance k from any pixel of color c_i. (The factor $8k$ is due to the properties of L_∞-norm.) The naive algorithm would be to consider each $p_1 \in \mathcal{I}$ of color c_i and for each $k \in [d]$, count all $p_2 \in \mathcal{I}$ of color c_j with $|p_1 - p_2| = k$. Unfortunately, this takes $O(n^2 d^2)$ time. To obviate this expensive computation, we define the quantities

$$\lambda^{c,h}_{(x,y)}(k) \triangleq |\{(x+i, y) \in \mathcal{I}_c \mid 0 \le i \le k\}| \quad (5)$$

$$\lambda^{c,v}_{(x,y)}(k) \triangleq |\{(x, y+j) \in \mathcal{I}_c \mid 0 \le j \le k\}| \quad (6)$$

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

Our algorithms work by first computing $\lambda^{c_j,v}_p$ and $\lambda^{c_j,h}_p$. The following two sections give separate algorithms for the case when d is small (running time is $O(n^2 d)$) and when d is large (running time is $O(n^3 d^{\omega-3})$ where $\omega \in [2,3)$ is the exponent of the fastest algorithm for matrix multiplication).

2.1. Dynamic Programming: when d is small

The following equation is easy to check

$$\lambda^{c,h}_{(x,y)}(k) = \lambda^{c,h}_{(x,y)}(k-1) + \lambda^{c,h}_{(x+k,y)}(0) \quad (7)$$

with the initial condition

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

Now, $\lambda^{c,h}_p(k)$ is computed for all $p \in \mathcal{I}$ and for each $k = 1, \ldots, d$ using Equation 7. The correctness of this algorithm is obvious. Since we do $O(n^2)$ work for each k, the total time taken is $O(n^2 d)$.

In a similar manner, $\lambda^{c,v}_p$ can also be computed efficiently. Now, ignoring boundaries, we have

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

This computation takes just $O(n^2)$ time.

The hidden constants in the overall running time of $O(n^2 d)$ are very small and hence this algorithm is extremely efficient in practice for small d.

2.2. Matrix Multiplication: when d is large

When d is, say, $O(n)$, the dynamic programming algorithm given in the previous section is sub-optimal. We resort to a more sophisticated dynamic program – algorithms for fast matrix multiplication. We now sketch the main ideas involved.

The first observation is an alternate way to compute $\lambda^{c,h}_p(k)$ via matrix multiplication. Let $\mathcal{I}_c$ be an $n \times n$ 0-1 matrix such that $\mathcal{I}_c(p) = 1 \iff \mathcal{I}(p) = c$. Now, define N_1 to be the $n \times nd$ matrix whose $(nk+y)$-th column

$y \in [n], k \in [d]$ is given by $[\overbrace{0,\ldots,0}^{y-1}, \overbrace{1,\ldots,1}^{k}, 0,\ldots,0]^T$. It is easy to see that $(\mathcal{I}_c N_1)[x, nk+y] = \lambda^{c,h}_{(x,y)}(k)$.

Now, the second observation is to use another $nd \times d$ matrix N_2 that will accumulate $\sum_{x=1}^{n} \lambda^{c,h}_{(x-k,y+k)}(2k)$ from the product $\mathcal{I}_c N_1$ for each distance $k \in [d]$ and column $y \in [n]$, To accrue this sum, the k-th column in N_2 looks (approximately) like $[1, 2, \ldots, 2k+1, 2k+1, \ldots, 2k+1, 2k, \ldots, 1]$. When $\mathcal{I}_c N_1$ is right-multiplied by N_2, the resulting $n \times d$ product represents the above sum, i.e., $((\mathcal{I}_c N_1)N_2)[y,k] = \sum_{x=1}^{n} \lambda^{c,h}_{(x-k,y-k)}(2k)$.

Note that $(\mathcal{I}_c N_1)N_2 = \mathcal{I}_c(N_1 N_2)$. In other words, $N_1 N_2 = N$ is a fixed $n \times d$ matrix. So, the algorithm will precompute N for a given n and use the fast matrix multiplication to compute $\mathcal{I}_c N$. By adding each column in this product, we can get $\sum_{(x,y)\in\mathcal{I}} \lambda^{c,h}_{(x-k,y+k)}(2k)$, the first term in the equation for $\Gamma^{(k)}_{c_i,c_j}(\mathcal{I})$. Using similar ideas, the other three terms can be computed.

The total time taken by this algorithm is the time taken to multiply $\mathcal{I}_c$ and N, i.e., an $n \times n$ 0-1 matrix and a fixed $n \times d$ integer matrix. Using a block $d \times d$ matrix multiplication, this can be achieved in time $O(n^3 d^{\omega-3})$ where $\omega \in [2,3)$ is the exponent for the fastest matrix multiplication algorithm (for instance, for Strassen's algorithm, $\omega \approx 2.7$). With the availability of fast hardware to perform matrix multiplication, this method promises to be attractive. One could also use several existing optimal techniques which parallelize matrix multiplication to implement this algorithm efficiently in practice.

3. Distance Measures

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

The L_1 and L_2 distance measures are commonly used when comparing two feature vectors. In practice, the L_1 distance measure performs better than the L_2 distance measure because the former is statistically more robust to outliers [11]. Hafner *et al.* [4] suggest using a more sophisticated quadratic form of distance measure, which tries to capture the perceptual similarity between any two colors. To avoid intensive computation of quadratic functions, they propose to use low-dimensional color features as filters before using the quadratic form for the distance measure.

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

$$|\mathcal{I} - \mathcal{I}'|_{h,L_1} \triangleq \sum_{i\in[m]} |h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')| \tag{8}$$

$$|\mathcal{I} - \mathcal{I}'|_{\gamma,L_1} \triangleq \sum_{i,j\in[m],k\in[d]} |\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) - \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')| \tag{9}$$

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

Example. Consider two pairs of images $\langle\mathcal{I}_1, \mathcal{I}_2\rangle$ and $\langle\mathcal{I}'_1, \mathcal{I}'_2\rangle$. Let $h_{c_i}(\mathcal{I}_1) = 1000$, $h_{c_i}(\mathcal{I}_2) = 1050$, $h_{c_i}(\mathcal{I}'_1) = 100$, and $h_{c_i}(\mathcal{I}'_2) = 150$. Even though the absolute difference in the pixel count for color bucket i is 50 in both cases, clearly the difference is more significant for the second pair of images. Thus, the difference $|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|$ in Equation (8) should be given more importance if $|h_{c_i}(\mathcal{I}) + h_{c_i}(\mathcal{I}')|$ is small and vice versa. We could therefore replace the expression $|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|$ in Equation 8 by

$$\frac{|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|}{1 + h_{c_i}(\mathcal{I}) + h_{c_i}(\mathcal{I}')} \tag{10}$$

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

This intuition has theoretical justification in [6] which suggests that sometimes, a "relative" measure of distance d_μ is better. For $\mu > 0, r, s \geq 0$, d_μ is defined by

$$d_\mu(r,s) = \frac{|r-s|}{\mu + r + s}. \tag{11}$$

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

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

$$|\mathcal{I} - \mathcal{I}'|_{h,d_1} \triangleq \sum_{i\in[m]} \frac{|h_{c_i}(\mathcal{I}) - h_{c_i}(\mathcal{I}')|}{1 + h_{c_i}(\mathcal{I}) + h_{c_i}(\mathcal{I}')} \tag{12}$$

$$|\mathcal{I} - \mathcal{I}'|_{\gamma,d_1} \triangleq \sum_{i,j\in[m],k\in[d]} \frac{|\gamma^{(k)}_{c_i,c_j}(\mathcal{I}) - \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')|}{1 + \gamma^{(k)}_{c_i,c_j}(\mathcal{I}) + \gamma^{(k)}_{c_i,c_j}(\mathcal{I}')} \tag{13}$$

4. Performance Measures

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

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

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

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

Recall *vs.* **Scope.** Let $\mathcal{Q}$ be a query and let $\mathcal{Q}'_1, \ldots, \mathcal{Q}'_a$ be multiple "answers" to the query ($\mathcal{Q}$ is called a *category query*). Now, the *recall* r is defined for a *scope*, to be $s > 0$ as $|\{\mathcal{Q}'_i \mid \operatorname{rank}(\mathcal{Q}'_i) \leq s\}|/a$. This measure is simpler than the traditional *recall vs. precision* but still evaluates the effectiveness of the retrieval [12].

5. Experimental Methodology

5.1. Efficiency Considerations

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

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

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

Though in theory the size of a correlogram is $O(m^2 d)$ (the size of an autocorrelogram is $O(md)$), we observe that the feature vector is often sparse. This sparsity could be exploited to cut down storage and speed up computations.

Filtering. A good balance between effectiveness and efficiency can be obtained by adopting a two-pass approach [4]. First, we retrieve a set of N images using an inexpensive search algorithm; next, a more sophisticated matching technique is used to compare only these images to the query. The initial ranking of the images could be poor, but if we ensure that the initial set contains the answer images, these images are likely to be highly ranked in the final ranking. The choice of N is important here: the initially retrieved set should be large enough to contain the answers and should be small enough so that the total retrieval time is reduced.

5.2. Experimental Setup

We have implemented correlograms and autocorrelograms on a large image database and use them for image retrieval. The parameters of our experiments are listed below.

Database. The database consists of 14,554 color JPEG images of size 232 × 168. This includes 11,667 images used in Chabot [7], 1,440 images used in QBIC [3], 1,005 images from Corel, a few groups of images in PhotoCD format, and a number of MPEG video frames from the web. The database is thus quite heterogeneous.

Featurebase. We consider the RGB colorspace with color quantization into 64 colors. To improve performance, we first smooth the images by a small amount. We compute the autocorrelogram, histogram, CCV, and CCV/C for each image in the database. (We did not have to compute the correlogram here, as the autocorrelogram itself was sufficient to produce good results.) We use the dynamic programming algorithm with the distance set $D = \{1, 3, 5, 7\}$ for computing the autocorrelograms. For such a small-sized D, the computation time is small. The size of this feature is also the same as that of CCV/C. We construct the featurebase in parallel.

Queries. Our query set consists of 77 queries, each with a unique correct answer. The queries are chosen to represent various situations like different views of the same scene, large changes in appearance, small lighting changes, spatial translations, etc. Examples of some queries and answers (and the rankings according to the histogram, CCV, CCV/C, and autocorrelogram methods) are shown in Figure 3 (the color versions are available at `http://www.cs.cornell.edu/home/huang/`). We use both the L_1 and d_1 distance measures for comparing feature vectors and use the sparsity of the feature vectors to speed up processing. The query response time for autocorrelograms is under 2 sec on a Sparc-20 workstation. We also ran 4 category queries, each with $a > 1$ answers – Query 1 ($a = 22$ owl images), Query 2 ($a = 17$ fox images), Query 3 ($a = 6$ movie scenes), and Query 4 ($a = 6$ moving car images).

hist: 367. ccv: 230. ccv/c: 245. auto: 1. hist: 119. ccv: 36. ccv/c: 25. auto: 2.

hist: 310. ccv: 177. ccv/c: 160. auto: 6. hist: 388. ccv: 393. ccv/c: 314. auto: 2.

Figure 3. Sample queries and answers with ranks for various methods. (Lower ranks are better.)

6. Results

r- and p_1-measures. The overall performance of the autocorrelogram, histogram, CCV, and CCV/C using 64 color bins is compared in Table 1. The L_1 distance measure is used. We can see that autocorrelograms perform the best

Method	hist	ccv	ccv/c	auto
r-measure	6301	3934	3272	172
avg. r-measure	82	51	42	2
p_1-measure	21.25	27.54	31.60	58.03
avg. p_1-measure	0.28	0.36	0.41	0.75

Table 1. Performance of various methods.

both in the r- and p_1-measures.

For 73 out of 77 queries, autocorrelograms perform as well as or better than histograms. The change in the rank of the correct answer, averaged over all queries, is an improvement of 80 positions. In the cases where autocorrelograms perform better than color histograms, the average improvement in rank is 104 positions. In the four cases where color histograms perform better, the average improvement is just two positions. Autocorrelograms, however, still rank the correct answers within top six in these cases.

We adopt the approach used in [8] to analyze the statistical significance of the improvements. We formulate the null hypothesis H_0 which states that the autocorrelogram method is as likely to cause a negative change in rank as a non-negative one. Under H_0, the expected number of negative changes is $M = 38.5$, with a standard deviation $\sigma = \sqrt{77}/2 \approx 4.39$. The actual number of negative changes is 4, which is less than $M - 7\sigma$. We can reject H_0 at more than 99.9% standard significance level.

For 67 out of 77 queries, autocorrelograms perform as well as or better than CCV/C. On an average, the autocorrelogram method ranks the correct answer 40 positions higher. In the cases where autocorrelograms perform better than CCV/C, the average improvement in rank is 66 positions. In the ten cases where CCV/C perform better, the average improvement is two positions. Autocorrelograms, however, still rank the correct answers within top 12 in these cases. Again, statistical analysis suggests that autocorrelograms are better than CCV/C.

>From a user's point of view, these results can be interpreted as follows: given a query, the user is guaranteed to locate the correct answer by just checking the top two search results (on average) using autocorrelograms. On the other hand, the user needs to check at least the top 80 search results (on average) to locate the correct answer in the case of histogram (or top 40 search results for the CCV/C). In practice, this suggests that the former is a more "usable" image retrieval scheme than the latter two.

Recall Comparison. Table 2 shows the performance of three features on our four category queries. The L_1 distance measure is used. Once again, autocorrelograms perform the best.

Relative Distance Results. Table 3 compares the results obtained using d_1 and L_1 distance measures on different features (64 colors). Using the d_1 distance measure is clearly superior. The improvement is specially noticeable for histograms and CCV/C. A closer examination of the results shows, however, that there are instances where the d_1 distance measure performs poorly compared to the L_1 distance measure on histograms and CCV/C. It seems that the failure of the d_1 measure is related to large changes in overall image brightness. Autocorrelograms, however, are not affected by d_1 in such cases. Nor does d_1 improve the performance of autocorrelograms much. In other words, autocorrelograms

	Recall					
	Query 1			Query 2		
Scope	hist	ccv/c	auto	hist	ccv/c	auto
10	.14	.19	**.24**	.13	.19	**.38**
30	.19	.19	**.38**	.31	.38	**.63**
50	.19	.24	**.57**	.31	.38	**.75**
	Recall					
	Query 3			Query 4		
Scope	hist	ccv/c	auto	hist	ccv/c	auto
10	.20	.20	**1.0**	.20	.20	**.60**
30	.40	.20	**1.0**	.20	.20	**.80**
50	.40	.60	**1.0**	.20	.20	**.80**

Table 2. Scope *vs.* recall results for category queries. (Larger numbers are better.)

Meth. →	hist	ccv/c	auto	hist	ccv/c	auto
Meas. ↓	L_1 distance measure			d_1 distance measure		
r	6301	3272	172	926	326	164
avg. r	82	42	2	12	4	2
p_1	21.3	31.6	58.0	47.9	52.1	59.9
avg. p_1	0.28	0.41	0.75	0.62	0.68	0.78

Table 3. Comparison of L_1 and d_1

seem indifferent to the d_1 distance measure. We need to formally investigate these issues in greater detail.

Filtering Results. The r-measure improves from 172 to 166 and p_1-measure from 58.03 to 58.60 when a histogram filter (with d_1 distance measure) is used before using the autocorrelogram (with L_1 distance measure). This improvement is because of the elimination of false positives. As anticipated, the query response time is reduced since we compare the query to the autocorrelograms of only a small filtered subset of the featurebase.

7. Conclusions and Future Work

We have described a new image feature that can be used to index and compare images. Since this feature captures the spatial correlation of colors in an image, it is effective in discriminating images. It thus rectifies the major drawbacks of the classical histogram method. The correlogram can also be computed efficiently. Our experiments on a large image database evaluated using fair performance measures show that the correlogram performs very well.

The correlogram is powerful and needs to be explored in further detail. One practical question is, can the correlogram be compressed with only a minor loss in quality? It will also be interesting to study the use of correlograms for target search and open-ended browsing of image databases [2].

Some of our results show that when there is a large lighting change between a query and its correct answer, autocorrelograms rank the answer within the top 15 (in these cases, the histogram and CCV/C fail). It will be interesting to try correlograms on color spaces which are stable under lighting change and are also perceptually uniform. Our experiments so far have been based on color quantization in the RGB colorspace.

Acknowledgments. We are grateful to R. Rubinfeld and A. Singhal for their very helpful comments. We thank D. Coppersmith for his answer to our question regarding fast rectangular matrix multiplication. Finally, we would like to thank G. Pass who provided us with the framework for the IR system used in our experiments.

References

[1] Content-based image retrieval systems. *IEEE Computer*, 28(9), 1995.

[2] I. J. Cox *et al.* PicHunter: Bayesian relevance feedback for image retrieval. *International Conference on Pattern Recognition*, 1996.

[3] M. Flickner *et al.* Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995.

[4] J. Hafner *et al.* Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):729–736, 1995.

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

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

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

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

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

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

[11] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, 1987.

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

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

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

[15] G. J. G. Upton and B. Fingleton. *Spatial Data Analysis by Example. Vol I.* John Wiley & Sons, 1985.