Handling Free-Shaped Queries in Large Image Databases Using a Sampling-Based Image Retrieval Technique

Khanh Vu, Kien A. Hua, and Duc A. Tran
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362, U. S. A.
E-mail: {khanh,kienhua,dtran}@cs.ucf.edu

Abstract: The rapid growth of digital image data increases the need for efficient and effective image retrieval systems. Such systems should provide functionality that tailors to the user's need at the time of query. In this paper, we present a new image retrieval technique that allows users to control the relevancy of the results. For each image, the color contents of its regions are captured and used to compute similarity. Various factors, assigned automatically or by the user, allow high recall and precision to be obtained. We also present different 'initial search' strategies for whole and sub-image matching. We implemented the proposed technique for a large database of 16,000 images. Our experimental results show that this technique is not only space-time efficient but also more effective than recently proposed color histogram techniques.

Keywords: image indexing/retrieval, free-shaped queries, quick-and-dirty tests

1. Introduction

The explosion in the amount of digital image data has brought about the need for robust content-based image retrieval (CBIR) systems. This demand has made CBIR a very active area of research in recent years [Swain (1993), Niblack (1993), Faloutsos (1994), Gong (1995), Hsu (1995), Lu (1994), etc.]. In a typical CBIR system, some essential properties of database images are extracted and stored as feature vectors. During the retrieval process, the feature vector of the query image is computed and matched against those in the database. The returned images should be wholly or partially similar to the entire or relevant part of the query image, respectively.

To support image indexing/retrieval, color histogram is commonly used today. A color histogram describes the distribution of colors in the image. An advantage of this approach is that the feature vectors are insensitive to small changes in the viewing position. Representing images by their color histogram alone, however, is prone to false matches due to the lack of spatial information. This approach would treat a blue car at the bottom of the query image as similar to a database image with a blue sky at the top. Recent approaches have attempted to address this problem by integrating the color and spatial information. We discuss some of these techniques below.

A multiple color histogram approach was introduced in [Gong (1995)]. In addition to a global histogram for the entire image, this technique creates a local histogram for each of the nine equal partitions of the image. During a retrieval process, the user has a choice of matching any combination of the histograms. To reduce the expensive cost of matching the histograms, a numerical key is precomputed for each histogram to facilitate simple numerical-key searches.

Correlogram was recently proposed in [Huang (1997)] as a new image feature. This clever technique distills spatial information into the color distribution by counting only pixels of color j at a specified distance k from a pixel of color i. In its restricted version, autocorrelogram, color j and color i are the same. This approach is very robust, able to

tolerate large changes in appearance of the same scene caused by changes in viewing positions, changes in the background scene, etc.

Another approach attempting to integrate color-spatial information was presented in [Chua (1997)]. For each image, its color-spatial information is captured in an image signature obtained by superimposing a set of color signatures. They are computed as follows. We compute one color signature for each primary color in the image. The image is partitioned into k non-overlapping cells of equal size. Each color signature has k bits, one for each of the cells. For each cell, if the percentage of the total number of pixels having a given primary color is greater than a threshold, then the bit corresponding to this cell in the respective color signature is set. During the image retrieval process, the signature of the query image is computed and matched against those in the database to retrieve the desired images. This technique is efficient for large databases. However, it is not shift-invariant and not effective for images without dominant colors.

We note that all of the above techniques construct the feature vector using all the pixels of an image. This approach is not only expensive, but also has the following drawbacks:

1. For most applications, not all pixels are equally important in terms of query processing. Typically, the surrounding background is less important than the objects of interest. Equal contribution of all pixels in the similarity computation would dilute the intended semantic of the query.
2. Due to tight integration of the color-spatial information, there is little flexibility in the way we can use the same feature vectors to support subimage matching, i.e., matching the query image against subpart of a database image. We will give experimental results to show that lacking this capability leads to false exclusion of qualified images.

To support subimage matching, another proposed technique [Wood (1998)] relies on sophisticated segmentation techniques to capture images' contents. Seventeen parameters represent each of one hundred largest regions of the image. Thus, for each image, one hundred feature vectors are stored and compared against for similarity. This technique is highly effective but mainly for relatively small databases due to very high resource requirements.

To handle very large image databases, we propose a technique called Sampling-Based Matching or SamMatch for short [Hua (1999)]. SamMatch is inspired by the digitization of sound in which the magnitudes of a sound wave are sampled at some fixed time interval and stored as numbers. Such numbers capture the characteristics of the sound wave to the details allowed by the sampling rate. Similarly, SamMatch samples an image with respect to space intervals. For each sampling region, we compute the average color of the pixels in the region. These averages form the feature vector of the image. This approach has the following advantages: (i) We will show later that SamMatch requires less space than any of the aforementioned techniques. The smaller feature vectors are less expensive to compare, making this scheme suitable for very large databases. (ii) Unlike local histograms which are tied to predetermined areas of the image, SamMatch enables the users to associate areas with objects in the image. This capability allows areas of interest to be more flexible in shape and size. Furthermore, multiple weights can be assigned to different matching regions to precisely express the intent of the query. (iii) The system can apply different sampling rates (i.e., scaling sampling regions with respect to image size) on the query image to support subimage matching. This can identify images with subparts matching the query image at different scalings.

To further enhance the effectiveness of SamMatch, we also consider the following observations:

- Matches on rare colors are more discriminating.
- Matches at related (e.g., adjacent) sampling groups are more significant.

Our system considers these measures and automatically adjusts the weights of the sampling regions accordingly. We will discuss these features in more detail later.

The remainder of this paper is organized as follows. We describe the details of SamMatch in Section 2. In Section 3, we present search strategies for whole and sub-image matching. The experimental study based on a large database of 16,000 images is presented in Section 4. Finally, we give our concluding remarks and discuss future work in Section 5.

2. Sampling-Based Image Matching

In our approach, a set of fixed sampling locations is predetermined for all database images. To compute the feature vector for each image, we extract the color information for each pixel group (henceforth referred to as *matching region*.) In order to ensure accurate ranking of similar images, many such matching regions may be necessary. In this section, we first present our basic similarity function. We then discuss how to enhance the similarity measure with weight factors.

2.1 Sampling Regions and Their Features

To estimate the number of regions and features needed to yield a good performance, we have analyzed the probability of matching with respect to the number of matching regions. Let n be the number of matching regions in a subimage and k a particular set of feature tuple (e.g. [color_m, texture_q ,...]). We use p^k_i to denote the probability that region i is represented by tuple k, $1 \le i \le n$. To simplify the analysis, let us assume that p^k_i and p^k_j are independent for any i and j. With this assumption, the probability that a subimage is represented by a set of particular tuples at its respective regions is:

$$P_n = \prod_{i=1}^{n} p_i^k$$

This probability should be very small in practice, even with single-feature tuples. For color feature, for instance, its distribution for such regions is depicted in `Figure 1` (dotted line), using the data collected from our database. We observe that p^k_i is not uniform for all colors. Fortunately, p^k_i is generally very small, less than 0.01 according to our database. Even a moderate n, say $n = 25$, can make P_n small enough, $P_n = (1/100)^{25}$, to handle a very large database.

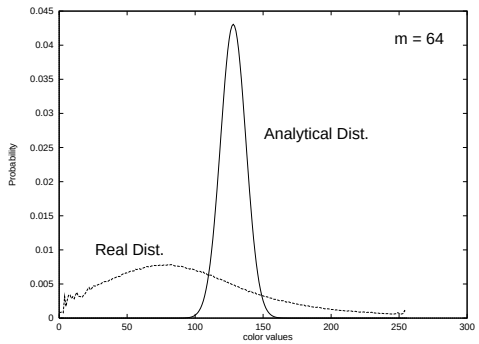


Figure 1: The distribution of the region colors

Although region contents are arguably independently distributed, the above analysis is beneficial for the following reasons:

1. The contents of neighboring regions may be highly related in certain image database domains where most of its images come from the same category. For general image databases, such relationships are greatly reduced. In fact, much research has exhibited statistically that, on the average, the relationship between

pixels tend to decrease after 15 or 20 pixels [Andleigh (1996)]. In our implementation, matching regions are at least 16 pixels apart.

2. In reality, it is difficult to compute the precise probability of a particular feature occurrence, considering databases are constantly growing in size. To counter this fact, our analysis is intended to provide a rough estimate of the minimum number of matching regions, many more of them are actually used to ensure high effectiveness.
3. This analysis also serves as a guide to query the database. For instance, the object of interest should enclose at least a certain number of matching regions.

To support general applications, regions can be sampled uniformly across the image. Matching regions are evenly spread out in the image frame as shown in `Figure 2`. This sampling produces 113 regions. We note that the number of regions is substantially more than what we need to obtain good performance for whole-image queries. The high sampling rate is used to better support subimage matching.

In order to keep the storage overhead low, the representation of the color information for each matching region should be compact. In our system, it is computed as the most dominant Haar wavelet coefficient of the matching region (i.e., the average of the color values) as follows. It is known that Hue, Value and Chroma (HVC) each represents one attribute of human color perception. When representing color as (RGB) data, all three components (RGB) have to be considered in order to change one attribute of H,V,C. The color similarity comparison is, thus, cumbersome at query processing time. We can address this issue by transforming (RGB) color data to *Munsell* (HVC) color data using the mathematical transformation presented in [Hiyahara (1988)]. In this uniform color system, the dissimilarity of two colors is simply the distance between them. We quantized (HVC) data into 256 possible values. The region color is then represented by the most dominant Haar wavelet coefficient of the matching region.

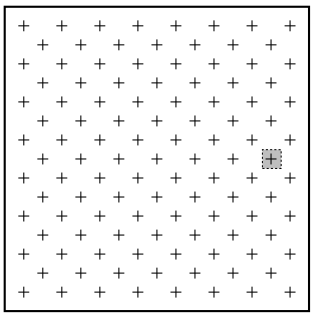


Figure 2: Sampling regions

With n and c determined, the color information of such n regions, referred to as *region colors*, of two images can be compared one-to-one to produce scores. These scores are added up to determine the similarity of the two images.

2.2 Automatic Weight Assignment Using Color Distribution

We note that the distribution of the average color is not uniform. When comparing the region colors of two images, we can give more weight to matches on rare colors since they are more discriminating. To take advantage of this feature, we collected statistics from our image database. The distribution of the average color according to our database is plotted in `Figure 1`, i.e., the dotted line. It shows that the distribution curve has a bell shape with higher frequencies clustered near the middle of the color spectrum. With this distribution, the weight for a match on color c_i is its inverse frequency.

To confirm the above observation, we have analyzed the theoretical distribution of the average color. We first assume that the color values of the pixels are independently distributed. Using the properties of probability generating functions, we can derive the color

distribution of a region of m pixels as follows. The probability generating function of the color distribution of a matching region of one pixel is:

$$P(z) = \left[\frac{1}{c} \cdot z + \frac{1}{c} \cdot z^2 + \frac{1}{c} \cdot z^3 + \ldots + \frac{1}{c} \cdot z^c \right],$$

The sum of the color values of the m pixels has the following probability generating function:

$$\begin{aligned} P(z) &= \left[\frac{1}{c} \cdot z + \frac{1}{c} \cdot z^2 + \frac{1}{c} \cdot z^3 + \ldots + \frac{1}{c} \cdot z^c \right]^m \\ &= a_m \cdot z^m + a_{m+1} \cdot z^{m+1} + a_{m+2} \cdot z^{m+2} + \ldots + a_{c \cdot m} \cdot z^{c \cdot m}, \end{aligned}$$

where a_i is the sum of like terms z^i. This sum of color values ranges from m to $c \cdot m$. Since the color values range only from 1 to c, the sum is quantized to reduce its possible values to c, i.e., $Q : c \times m \rightarrow c$. In our system, $\mathrm{Q(x)} = \lfloor \mathrm{x/m} \rfloor$, i.e., the average of the color values in the region.

The distribution of the region color after quantization is plotted in `Figure 1`, (the solid curve). We see that it is not uniform and has a behavior similar to that of the experimental curve. The discrepancy is due to our assumption that the colors of the neighboring pixels are independently distributed. We note that a very steep curve indicates that the averages share a small set of dominant colors. Under this circumstance, it would be ineffective to rely on average colors to search similar images. Fortunately, the experimental curve does not fall in this category. In fact, it shows a large number of frequent colors making color matching much more discriminating.

We note that the size of the sampling region and the number of possible color c can influence the distribution of the regional average color. To demonstrate these facts, we made the plots shown in `Figure 3` and `Figure 4` using the statistics collected from our database. They show the following properties:

1. A larger region (i.e., larger perimeter) results in a taller and narrower bell shape (see `Figure 3`). This, in turn, causes a vast majority of average color values to fall near the middle of the color spectrum (i.e., low standard deviation). Consequently, the distinguishing power is greatly reduced. Larger matching regions, however, tolerate minor shifting. Interestingly, for the same region size, the number of contributing pixels has very little effect on the distribution. In `Figure 3`, scarcely scattered (m = 64) and more populated (m = 256) regions of the same perimeter (peri. = 128) have nearly identical distributions. This property allows the capture of a region color using all its available pixels without altering the distribution.

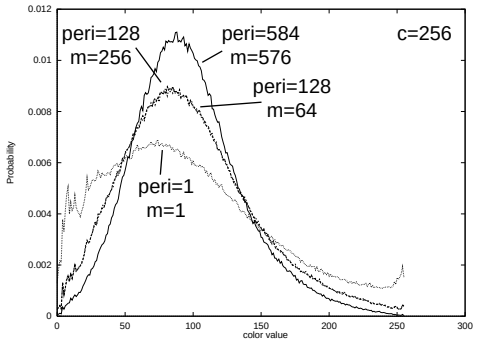


Figure 3: The effect of the region size

2. A smaller *c* results in a steeper bell shape, and therefore less distinguishing power, `Figure 4`. In one extreme, using binary values (i.e., black or white) does not help much in determining similarity. The number of possible colors should be chosen based on various factors, including the number of regions, the space overhead, the nature of the application, etc.

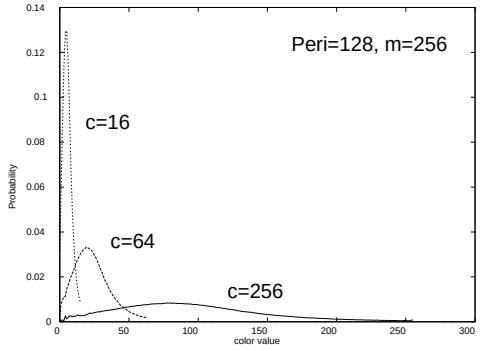


Figure 4: The effect of the number of colors (c)

From the above discussion, the values of *c* and the region area are critical in the effectiveness of the technique. It is important that the appropriate values be selected for the application at hand. In our current application, $c = 256$ and sampling regions is 16x16.

One result of the above observation is to let the system assign more weight to a region which is less likely to match, and is therefore more discriminating. Using its inverse frequency as a weight factor will help improve the matching precision and provide more accurate ranking. Thus, the basic similarity score of two corresponding matching regions in the query image and a database image can be computed as follows:

$$s_i = \frac{a_i}{1+|c_i^D - c_i^Q|},$$

where c_i^D and c_i^Q are the color values of the i^{th} region of a database image and the query image, respectively. Note that '1' is added in the denominator to prevent possible zero division.

2.3 Adjusting Scores Using Spatial Correlation

Matching based on the basic similarity scores discussed above may not accurately reflect the user's perception of similarity. As an example, let us consider a query image with a large red rose surrounded by a green background. Two potentially matched database images are found. One has a pink rose with a blue background; and the other one has scattered green areas along its edges and no rose. Potentially, the second image can be ranked higher because some of its areas match the green background in the query image perfectly. Obviously, the first database image should be ranked higher. This problem is addressed in our system by taking into account the spatial correlation.

We consider a cluster of well-matched regions as more significant than a collection of scattered matched regions. The rationale is that a cluster of matched regions often identifies an object in the image (e.g., a rose), whereas scattered regions are less likely to form a meaningful object. To exploit the spatial correlation, we let the user input a spatial distance *d*. During a search, the system raises/lowers the score of a matching region according to the basic similarity scores of the matching regions within the *d*-distance vicinity. In other words, the matching score of a region is amplified if its neighboring regions also match the query image well. One way to compute the new score of a matching region taking into account the spatial correlation is as follows. Let us consider region *i*. All of its immediate adjacent

matching regions form its 1st circle centered at region *i*. The next outer circle is its 2nd circle, and so on. Let $s^{(k)}$ denote the geometric mean of the scores of the regions on the k^{th} circle centered at region *i*. The new score of region *i* is:

$$s_i^c = \sqrt{s_i \cdot C^{(1)}},$$

where

$$C^{(k)} = \begin{cases} \sqrt{s^{(k)} \cdot C^{(k+1)}} & \text{if } 1 \le k < d \\ s^{(k)} & \text{if } k = d \end{cases}$$

When this feature is disabled, $s_i^c = s_i$. In our implementation, the default is $d = 2$.

2.4 User-defined Spatial Weight

With the region colors captured separately, the contribution of each region to the query can be precisely controlled allowing the user to more accurately describe the rough idea of what the returned images should be. For example, to sketch a query image, the user usually draws only the objects of interest, not the entire image composition. In the case of using an example image as a query, the user is able to specify the areas most appropriate by partially including or excluding various areas of the query image.

In our system, users are allowed to describe their intents by assigning different weights to matching regions of varying significance. The retrieved images are ranked according to these specifications. Let w_i denote the spatial weight factor of region *i* and s^c_i be the score earned by region *i*. The scoring function considering spatial significance is:

$$S = \sum_{i=1}^{n} w_i \cdot s_i^c \qquad (1)$$

The application of the spatial weight factors reflects the intent of the user in querying the database.

2.5 Handling Scaling

Another distinct advantage of the sampling-based approach is that the system can apply various sampling rates on the image query to find matches at different scalings. An example is given in `Figure 5`, in which the query image is sampled at three different rates (b, c, d in this figure) in order to match larger, same size, and smaller objects, respectively.

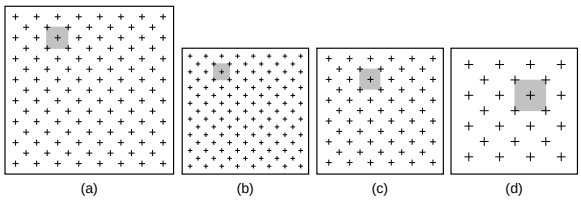


Figure 5: A fixed sampling rate for all database images (a). Three sampling rates for the query image: a higher rate to find larger matching objects (b), the same rate to find matching objects of the same sizes (c), and a lower rate to find smaller matching objects (d).

We observe that the more sampling rates on the query image, the finer scales to be compared at the query time. Since only one sampling rate is necessary for each database image no matter how many scales to be searched, there is no storage overhead penalty for more sampling rates of the query image.

3. Image Retrieval

In this section, we discuss our retrieval procedure in detail. Since our approach is capable of handling both whole and sub-images (of any shape) matching, we present different strategies to perform initial search prior to applying the detailed similarity measure.

3.1 'Quick-and-dirty' Filtering

When we are interested in whole matching (at similar locations), SamMatch enables an effective initial search for qualified images. Normally, to enable the most accurate retrieval possible, the detailed similarity computation, Equation (1), should be carried out on all database images. However, such sequential matching can be avoided using a two-pass approach widely adopted [Chua (1997), Hafner (1995), Faloutsos (1997), etc.] in the trade-off of effectiveness and efficiency. In the first pass, a set of potential image candidates are retrieved using an inexpensive search criterion, called the *quick-and-dirty* test. Equation (1) is then applied on this set to further filter it and rank the remaining.

To facilitate the filtering process in Pass 1, some values (e.g., average values of color components [Faloutsos (1997)]) can be computed for each image and saved in the database. The advantage of this approach is that the filtering parameters can be precomputed. Existing quick-and-dirty tests are not very effective. They usually filter out less than 50% of the database images [Chua (1997)]. This is not acceptable for large databases considering the high cost of similarity computation in Pass 2. To address this issue, we use the weighted color c_q of all the region colors of the query image as the filtering factor, i.e.,

$$c_q = \sum_{i=1}^{n} w_i \cdot c_{q_i}$$

where w_i is the user-assigned spatial weight of region i whose color is c_{qi} . To see how c_q can be affective in filtering irrelevant images, consider a case of c_q, the expansion of which is:

$$c_q = \sum_{i=1}^{n} w_i \cdot c_{q_i} = w_1 \cdot c_{q_1} + \mathrm{K} + w_k \cdot c_{q_k} + \mathrm{K} + w_l \cdot c_{q_l} + \mathrm{K} + w_n \cdot c_{q_n}$$

where $c_{ql} > c_{qk}$ and region k is more relevant to the query than region l, i.e., $w_k > w_l$.
Consider a database image the weighted color c_I of which is:

$$c_I = \sum_{i=1}^{n} w_i \cdot c_{I_i} = w_1 \cdot c_{I_1} + \mathrm{K} + w_k \cdot c_{I_k} + \mathrm{K} + w_l \cdot c_{I_l} + \mathrm{K} + w_n \cdot c_{I_n}$$

Without loss of generality, suppose $c_{Ik} = c_{ql}$, $c_{Il} = c_{qk}$, and $c_{Im} = c_{qm}$ for $1 \le m \le n$, $m \ne k$, $m \ne l$. Thus,

$$\begin{aligned}
\left|c_I - c_q\right| &= \left|w_k \cdot c_{Ik} - w_k \cdot c_{q_k} + w_l \cdot c_{Il} - w_l \cdot c_{q_l}\right| \\
&= \left|w_k \cdot c_{q_l} - w_k \cdot c_{q_k} + w_l \cdot c_{q_k} - w_l \cdot c_{q_l}\right| \\
&= \left|w_k \cdot (c_{q_l} - c_{q_k}) - w_l \cdot (c_{q_l} - c_{q_k})\right| \\
&= \left|(w_k - w_l) \cdot (c_{q_l} - c_{q_k})\right| \\
&= (w_k - w_l) \cdot (c_{q_l} - c_{q_k}) \\
&> \varepsilon
\end{aligned}$$

for some proper tolerance ε value. Therefore, this image is excluded from further similarity computation. It is easy to see that the image would have passed through had we used the simple sum of all color values, i.e., $w_i = 1$, $1 \le i \le n$, since

$$\begin{aligned} \left|c_I - c_q\right| &= (w_k - w_l) \cdot (c_{q_l} - c_{q_k}) \\ &= 0 \cdot (c_{q_l} - c_{q_k}) \\ &= 0 \\ &< \varepsilon \end{aligned}$$

Although this scheme incurs computation overhead in pass 1, it is outweighed by the benefit of having a very high concentration of good candidates in the filtered set. Our experimental results indicate that about 90% of the database images can be eliminated. Exploiting the user's intent (i.e., spatial weight factors) in the filtering process also minimizes the false dismissal of potential candidates.

We observe that in pass 1, the search criterion or the *filter* should be an interval [$c_q - \varepsilon$, $c_q + \varepsilon$] should be carefully determined to avoid false dismissal of potential candidates. If the search color c_q is near the two ends of the color spectrum (which shares the same characteristic as the one shown in `Figure 1`), we want to use a larger threshold, i.e., a wider filter, because there are substantially fewer database images having the weighted color in this range. On the other hand, if the searched value is near the middle of the color spectrum, we want to consider a narrower filter in order to be more restrictive since there are many images in this color range. In our implementation, we control ε in order to fix the size of the filtered set at 10% the size of the database. The tolerance ε for a given search color c_q can be determined as follows:

$$\begin{aligned} 10\% &= 2 \cdot \varepsilon \cdot p(c_q) \\ \Rightarrow \varepsilon &= \frac{10\%}{2 \cdot p(c_q)} \end{aligned} \tag{2}$$

where $p(c_q)$ is the frequency of the weighted color c_q.

The above approach is able to filter images effectively to the user intent for whole image matching. In the following subsection, we present a more elaborate indexing scheme to enable even faster initial search to support our retrieval of free-shaped queries at different locations and scales.

3.2 Indexing and Retrieval of Free-Shaped Queries

To search for a matching subimage in a database image, we can "slide" the query image over the database image. At each sliding location, we compare the sampling regions of the query with the corresponding sampling regions of the subimage. If their total score is greater than a threshold, the containing image is included into the result set. This procedure can be repeated for every image in the database to retrieve all qualified images. Avoiding this straightforward sequential scanning is a major challenge for matching free-shaped subimages in large databases. To enable indexing, we slide windows of a predetermined *base* shape to extract subimages' indexing values from images (Note that the base shape is used for the indexing purpose; the final detailed comparison Equation (1) is performed on the original query). For general applications, we select the square shape as the base shape. Our approach can be easily modified to handle any shape. To support our indexing technique, we build our access structure in three steps as follows:

1. We compute the feature vector for each database image as described earlier.
2. We slide base-shaped windows of various sizes over each database image. At each location, the sliding window encloses and defines a subimage for the purpose of indexing. We derive the feature vector for this subimage by extracting the corresponding components of the feature vector of the whole image. In addition, we compute a signature which is a vector of some essential features of

the subimage (e.g using DCT, DFT, Haar wavelet transformation, or average-variance pairs).

3. For each database image, we map the signatures of its subimages into points in the corresponding vector space, and cluster them into a fixed number of minimum bounding regions (MBRs). These MBRs collectively represent the database image, and are inserted into an R* tree [Beckman (1990)].

Using the access structure discussed above, our free-shaped query matching procedure can be summarized as follows:

1. Query Preprocessing: (i) We apply different sampling rates to the query image, each designed to match against subimages of a predetermined size. For each sampling rate, we compute one feature vector for the query image. (ii) We identify a base-shaped area that covers most of the query within the noise limit. We compute the signature of this area using the components of the image feature vector with the highest sampling rate.
2. Initial Search: We retrieve the qualified MBRs from the R* tree using the signature as the search key.
3. Detailed Comparison: Each subimage in the returned set is compared against the original free-shaped query image using Equation (1). This detailed comparison is based on their feature vectors. If the comparison is positive, the database image containing the matching subimage is returned as a query result.

4. Experimental Study

We evaluate the effectiveness of our image retrieval system using a database of 15,808 images. They consist of a variety of categories. Our workload consists of 30 queries. Each is an image from the database, and used as an example to retrieve similar images. The database was inspected to determine relevant answers. These queries were selected to retrieve objects of various characteristics, such as a rose, a bird, a plane, balloons, buildings, skies, sand, statues, food, etc. We organized them into three groups as follows:

1. The first set consists of fifteen queries. Each has a set of correct answers. These answers have some parts very similar to its query.
2. The second set consists of nine queries. The relevant answers are images containing some parts similar to the selected regions of interest, which can be the entire image.
3. The third set is intended to evaluate our technique in handling queries whose sizes are different from those of the database images. Many existing image retrieval techniques (including those in our comparative study) do not support this realistic querying environment.

Performance Metrics Let A_1, A_2,..., A_q denote the q relevant images in response to a query Q. The recall R is defined for a scope S as:

$$R/S = \frac{\left|\{A_i \mid rank(A_i) \le s, s > 0\}\right|}{q}$$

This measure, denoted by *R/S*, indicates the percentage of the returned images ranked within the specified scope. For instance, if a query returns 30 out of 40 total relevant images in some scope s, then its *R/S* is computed as 30/40 or 0.75. The rationale for factoring in the scope is that result images that fall far behind in the ranking often do not make it to the user. In this study, we use average *R/S* as the performance metric. This metric was also used in [Huang (1997)]. We adopt it for our study due to its effectiveness in evaluating image retrieval techniques. We discuss our performance results in the following subsections.

4.1 Comparative Study

We investigate three versions of SamMatch:

- SamMatch^w or SM^w: This version considers only the spatial weight.
- SamMatch^c or SM^c: This version considers only the spatial correlation.
- SamMatch or SM: This is the full-fledged SamMatch. It takes into account both the spatial weight and spatial correlation.

Studying the first two versions helps us understand the benefits of each individual feature. Evaluating the third version allows us to assess the overall performance of the proposed technique. We compared SamMatch against Correlogram [Huang (1997)], and Color Histogram (CH). The rationale for our choices is as follows:

- We believe Correlogram is one of the most effective image retrieval techniques today. It is very robust and not very sensitive to overall color composition.
- CH is one of the most space-time efficient matching algorithms, although it does not perform as well as Correlogram.

By comparing SamMatch against these two techniques, both its time-space efficiency and retrieval effectiveness can be examined. For CH, we used the histogram intersection formula given in [Smith (1996)]. For the convenience of the reader, we repeat this formula below:

$$d_I(h,g) = \frac{\sum_{i=1}^{c} \min(h_i, g_i)}{\min(\sum_{j=1}^{c} h_j, \sum_{k=1}^{c} g_k)}$$

This function computes the distance between two color histograms *h* and *g*.

4.2 Performance under Set-1 Queries

The *R/S* averages for set-1 queries are presented in `Table 1`. In this type of query, SamMatch returns the correct answers in the top five of the list, while both CH and Correlogram retrieve only about 50% of them in the scope of 500. Although Correlogram performs better than CH, it could not exclude the background colors in the similarity computation due to the tight integration of the pixel information. An example is given in `Figure 6`.

The intent of this query was to retrieve images of the bird regardless of the background. Since Correlogram considers the unmatched background, it ranks the correct answers unacceptably low. On the contrary, SM^w maintains regional information allowing the user to specify the weight for each matching region. This approach allows it to outperform CH and Correlogram by a significant margin. SM^c does not rely on user-assigned spatial weights. Instead, it enhances the scores automatically by evaluating the spatial correlation of the regional matches. The performance results indicate that this automatic feature is also very effective. It is noticeably better than CH and Correlogram. When combining these two weighting features, the full-fledged SamMatch offers from 43% to 70% improvements over Correlogram according to our experiments. It is shown in `Figure 6` that only SM successfully ranks the correct answers in the top.

Table 1: Average R/S in close matches

Scope	*CH*	*Corr.*	*SM^c*	*SM^w*	*SM*
5	0.12	0.29	0.70	0.80	0.95
50	0.23	0.44	0.79	0.85	1.0
500	0.44	0.57	0.85	0.95	1.0

4.3 Performance under Set-2 Queries

In this experiment, the objects in the result images are similar to those in the query image (`Figure 7`) or are different from the query image due to changes in the viewing positions (`Figure 8`). The results of this study are summarized in `Table 2`. Again, we see that each of the two weighting techniques offers significant improvement over CH and Correlogram. Combining these two schemes provides a technique which is 30% to 50% better than Correlogram.

Table 2: Average R/S in similar matches

Scope	*CH*	*Corr.*	SM^c	SM^w	*SM*
5	0.12	0.14	0.17	0.18	0.20
50	0.24	0.35	0.62	0.65	0.71
500	0.44	0.61	0.72	0.85	0.90

To illustrate the experimental results, we present some queries and interesting answers in `Figure 7`, `Figure 8` and `Figure 9`. We note that CH and Correlogram only recognize whole matches with Correlogram performing significantly better. Only SamMatch performs well in all three queries.

4.4 Time and Space

Unlike recent histogram-based schemes incorporating spatial information into color composition to achieve higher effectiveness, our approach does not require high resources. In this subsection, we analyze some basic resource requirements of the studied schemes.

In the above implementation, the image size is 256×256, the number of possible colors is 256, the distance set D is {1, 3, 5, 7} (i.e., four color histograms) for Correlogram, and there are 113 sampling regions for SamMatch. The space requirement for each image's feature vector is:

- 113 bytes in SamMatch,
- $256 \cdot 2 = 512$ bytes for one single color histogram in CH, 2 bytes to record up to 65536 pixel counts for each color.
- $4 \cdot 256 \cdot 2 = 2048$ bytes in Correlogram, since there are 4 color histograms in its implementation.

This space requirement does not include the storage for filtering values. A histogram filter is used in Correlogram [Huang (1997)], the average color of the image x = (R_{avg}, G_{avg}, B_{avg})[Faloutsos (1997)] can be used for CH, and none needed in SamMatch (see 'quick-and-dirty test' Filtering section).

With respect to average response time, the ratio obtained from our experiment with no filtering is Corr.:CH:SamMatch = 2:1:1. That is, Correlogram requires twice as much time compared to CH and SamMatch to answer the same query on the same hardware.

In summary, CH is very efficient. It requires only 1/4 the storage space by Correlogram, and is twice as fast in terms of execution time. SamMatch, however, is even more efficient. It runs as fast as CH, and uses only 1/4 the storage space required by CH.

4.5 Filtering Effectiveness

Our filtering mechanism proves to be very effective in pruning away irrelevant images. By applying Equation (2) to compute an appropriate tolerance value for specific queries, only a small set of image candidates for each query is returned. The sizes of these sets range from 6.25% to 25% of the database depending on the weighted color c_q of the query image. These sizes are controlled by our intention to keep a minimal loss of qualified images, less than 2% in our study. Small filtered sets help reduce the response time, which is less than 4 seconds on average. Our filtering mechanism can also serve to remove false matches from the final

similarity computation. The relevant answers in the final set have slightly higher ranks if filtering is employed.

4.6 Performance under Set-3 Queries

We note that both CH and Correlogram were not designed to handle queries whose images are of different sizes from those of the database images. In contrast, our technique is highly effective in supporting this type of queries. Figure 10 shows an example of SamMatch's effectiveness in handling different-size queries. It demonstrates that SamMatch is robust to scaling and translation of objects. We observe that the objects in the retrieved images are of different sizes and at different locations. Furthermore, large irrelevant content (occupying as high as 75%) of a database image does not affect its ranking. Notice that the 19th ranking of the rightmost image in Figure 10 appears to be low. This is due to the misalignment of the 'apple' and the sliding windows in the region. Such misalignments can be minimized by using more window sizes and smaller sliding steps. We feel that the current configuration is sufficient for many applications.

5. Concluding Remarks

We have presented SamMatch, a new content-based image retrieval technique for large image databases. The color-spatial information of the image is captured in a set of region colors. This approach enables users to partially include or exclude areas of images in similarity matching. Our scoring function is also enhanced by two additional features:

1. Automatic weight assignment using color values
2. Adjusting scores by using spatial correlation

We also address efficiency by employing different efficient search mechanisms for whole matching and subimage matching. Experiments on a large image database show that this approach performs very well. While SamMatch requires less space and time than existing schemes, it achieves greater effectiveness for whole image matching. Our technique is also able to effectively handle different-size queries.

In the above implementation, we have employed an exhaustive scanning algorithm to detect the base area of the free-shaped queries in subimage matching. While the algorithm is fast (less than one half of a second), it is not fast enough for much higher sampling rates. To speed up the detection time, we have been working on a new detection algorithm that is able to reduce the time complexity by several orders of magnitude. We will utilize this efficient algorithm in our future implementation.

References

Andleigh, P.K., and Thakrar, K. (1996). *Multimedia Systems Design*. Prentice Hall, New York.

Beckman, N., Kriegel, H.P., Schneider, R., and Seeger, B. (1990). The r*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD*, 322--331.

Chua, T. S., Tan, K., and Ooi, E. C. (1997). Fast signature-based color-spatial image retrieval. *Proc. Computer Vision and Pattern Recognition*, 362--369.

Faloutsos, C., Equitz, W., Flickner, M., Niblack, W., Petkovic, D., and Barber, R. (1994). Efficient and effective querying by image content. *Journal of Intelligent Information Systems* vol 3(3),231-262.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *ACM SIGMOD*, 419--429.

Faloutsos, C. (1997). *Searching Multimedia Databases by Content*, Kulwer Academic Publishers, 71-75.

Gong, Y., Chua, H., and Guo, X. (1995). Image indexing and retrieval based on color histogram. *Proc. of the 2nd Int. Conf. on Multimedia Modeling*, 115--126.

Hafner, J., et al. (1995). Efficient color histogram indexing for quadratic form distance function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 729--736.

Hiyahara, M. and Yoshida, Y. (1988). Mathematical transform of (r, g, b) color data to munsell (h, v, c) color data. *SPIE Visual Communications and Image Processing*, 650--657.

Hsu, W., Chua, T., and Pung, H. (1995). An integrated color-spatial approach to content-based image retrieval. *ACM Multimedia*, 305--313.

Hua, K.A., Vu, K., Oh, J.W. (1999). SamMatch: A flexible and efficient sampling-based image retrieval technique for large image databases. *ACM Multimedia*, 225--234.

Huang, J., Kumar, S. R., Mitra, M., Zhu, W., and Zabih, R. (1997). Image indexing using color correlograms. *Proc. Computer Vision and Pattern Recognition*, 762--768.

Lu, H., Ooi, B., and Tan, K. (1994). Efficient image retrieval by color contents. *Proc. of the 1994 Int. Conf. on Applications of Databases*, 95--108.

Niblack, W., Barber, R., Equitz, W., Flicker, M., Glasman, E., Petkovic, D., Yanker, P., and Faloutsos, C. (1993). The QBIC project: Query images by content using color, texture and shape. *SPIE* V1908.

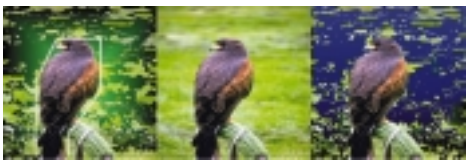
Smith, J.R. and Chang, S.F. (1996). Tools and techniques for color image retrieval. *SPIE*, 29-40.

Swain, M. and Ballard, D. (1991). Color indexing. *Journal of Computer Vision* vol 7(1),11-32.

Swain, M. (1993). Interactive indexing into image database. *SPIE* V1908.

Wood, M.E.J., Campbell, N.W., and Thomas, B.T. (1998). Iterative refinement by relevance feedback in content-based digital image retrieval. *ACM Multimedia*, 13--20.

Figure 6: Correct answers in set-1 queries

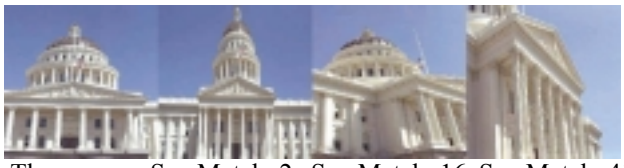


The query | SamMatch: 2 Corr.:1965 CH.:3260 | SamMatch: 3 Corr.:2336 CH: 1738

Figure 7: Some answers in set-2 queries

The query | SamMatch: 2 Corr.: 2 CH: 2 | SamMatch: 21 Corr.: 89 CH: 201

Figure 8: Some relevant answers in set-2 queries. Lower is better.



The query | SamMatch: 2 Corr.: 2 CH: 2 | SamMatch: 16 Corr.: 9 CH: 93 | SamMatch: 41 Corr.: 223 CH: 1258

Figure 9: Some of the highest ranked (left-to-right) images returned by SamMatch (first row), by Corr. (second row) and CH (third row) in response to query (left): Retrieve images containing a yellow flower

Figure 10: Some images (ranked, left-to-right, 1, 2, 3, 4, 5, and 19) returned in response to query (left): Retrieve images containing an apple of any size at any location