A New Approach to Image Retrieval with Hierarchical Color Clustering

Xia Wan and C.-C. Jay Kuo, *Senior Member, IEEE*

***Abstract*—After performing a thorough comparison of different quantization schemes in the $RGB, HSV, YUV,$ and $CIEL^*u^*v^*$ color spaces, we propose to use color features obtained by hierarchical color clustering based on a pruned octree data structure to achieve efficient and robust image retrieval. With the proposed method, multiple color features, including the dominant color, the number of distinctive colors, and the color histogram, can be naturally integrated into one framework. A selective filtering strategy is also described to speed up the retrieval process. Retrieval examples are given to illustrate the performance of the proposed approach.**

***Index Terms*—Color quantization, content-based retrieval, image database, image indexing, image retrieval, query processing.**

I. Introduction

ADVANCES in modern multimedia technologies have led to huge and ever-growing archives of images, audio, and video in diverse application areas such as medicine, remote sensing, entertainment, education, and on-line information services. This is similar to what occurred in the early computer development stage, during which the amount of alphanumeric data increased rapidly and many practical issues in the database management system (DBMS) arose. In the past, DBMS was designed to organize alphanumeric data into structured records indexed by key attributes so that information retrieval and storage could be done conveniently and efficiently. However, traditional DBMS does not work well for multimedia data due to difficulties in several aspects, which include the diversity of the data type (e.g., image, video, audio), the large capacity of the unit record (e.g., a raw 8-bit gray-level image of size 512×512 has 2.1 Mbits before compression), and the lack of semantic meaning of the data at the physical level (e.g., no semantic meaning at the pixel level for images). To exploit the full benefit of the explosive growth of multimedia data, there is a strong demand for developing efficient techniques for their storage, browsing, indexing, and retrieval [1]–[9].

Effective retrieval of image data is an important building block for general multimedia information management. For an image to be searchable, it has to be indexed by its content, which is manually annotated by keywords or automatically extracted by visual features. Although it seems effortless for a human being to pick out photos of horses from a collection of pictures, object recognition and classification are still among the most difficult problems in image understanding and computer vision. For a small image database, it would be easier to manually annotate a picture of horses by the keyword "horse" than to recognize a horse by visual feature analysis at different occlusion conditions and viewpoints. However, for a large image database, a prohibitive amount of labor will be involved for the annotation of objects in all images. In addition, a limited number of keywords is usually not sufficient to describe the details in a content-abundant image. To access images based on their content, low-level features such as colors [10]–[12], textures [13]–[15], and shapes of objects [16], [17] are widely used as indexing features for image retrieval to bypass the difficulties of image understanding.

Among various low-level features, the color information has been extensively studied because of its invariance with respect to image scaling and orientation. Color features used in image retrieval include global and local color histograms, the mean (i.e., average color), and higher order moments of the histogram [18]. Average and dominant colors can be used to filter out irrelevant images without too much computational cost. However, they do not support a detailed comparison of the color appearance among images. The global color histogram provides a good approach to the retrieval of images that are similar in overall color content. There has been research to improve the performance of color-based extraction methods. For example, the QBIC (query by image content) [4] system supports color feature extraction of manually outlined objects. An evaluation study made by Zhang and Smoliar [11] showed that the fixed-size local histogram is computationally simple and efficient in some applications. The color indexing method proposed by Stricker and Dimai [19] extracted color features defined in fuzzy regions adaptive to image content.

There are common issues underlying all color-based retrieval methods: the selection of a proper color space [20], the use of a proper color quantization scheme to reduce the color resolution, and the development of efficient feature representations to support a robust and flexible query process. The effect of color quantization on the performance of image retrieval has been reported by authors in [20] and [21]. It has been observed that the fixed color quantization scheme, which is commonly used in computing global and local histograms, has one major drawback. That is, similar colors might be quantized to different buckets in the histogram, thus leading to false misses. In this work, we propose to use a new color

Manuscript received October 31, 1997; revised May 30, 1998. This work was supported by Hewlett Packard and the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, with additional support from the Annenberg Center for Communication at the University of Southern California and the California Trade and Commerce Agency. This paper was recommended by Associate Editor S. Panchanathan.

The authors are with the Integrated Media Systems Center and the Department of Electrical Engineering—Systems, University of Southern California, Los Angeles, CA 90089-2564 USA.

Publisher Item Identifier S 1051-8215(98)06963-8.

feature obtained by hierarchical color clustering. This color feature is different from the multiresolution color histogram described in our previous work [21] in the sense that it allows natural color clustering according to the content of an image (i.e., image adaptive). We also describe a set of filtering methods based on the new color feature to facilitate the retrieval process. They include filtering by the dominant color, by the color width, and by hierarchical color distribution. A combination of these methods allows a prompt access to images in a large image database.

This paper is organized as follows. Image similarity measurements with color features are discussed in Section II. The retrieval performances of different quantization schemes in different color spaces are compared in Section III. A hierarchical color clustering algorithm is presented in Section IV. Indexing and retrieval schemes based on the clustering algorithm are proposed in Section V. Concluding remarks are drawn in Section VI.

II. Similarity Measurements with Color Features

Similarity measurements of images can be classified into three levels: pixel matching, feature matching, and semantic meaning. The pixel-based similarity measurements obtained via L_1- or L_2-norm distance are straightforward. However, since they are sensitive to image scaling, rotation, and translation, such measurements are seldom used in practice. Similarity comparisons with considering semantic meaning are ideal for retrieval purposes. Nevertheless, they are difficult to be implemented due to the lack of good techniques in image understanding. We focus on similarity measurements based on low-level features in this paper.

A. Fixed Quantization Case

The color histogram of an image describes its color distribution. Every pixel in the image corresponds to a point in a three-dimensional (3-D) color space. A similar image set can be selected based on the color distribution

$$\{T \mid \text{dist}\,(H_Q, H_T) < \varepsilon\}$$

where H_Q and H_T are color histograms of the query and target images, respectively, at the finest resolution level. That is, if a pixel is described by R, G, and B color components of n bits each, then H_Q and H_T are defined on the cubic lattice of $2^n \times 2^n \times 2^n$ points. However, the resolution of H_Q and H_T is too high to be used in practice. To simplify the computation, the color space has to be quantized to reduce the resolution. Thus, histograms on the quantized space are used to define the similarity set. That is, a new similar image set is obtained based on

$$\left\{T \mid \text{dist}\left(\hat{H}_Q, \hat{H}_T\right) < \varepsilon\right\}$$

where $\hat{H}_Q$ and $\hat{H}_T$ are quantized histograms of the query and target images, respectively. The quantization scheme in obtaining $\hat{H}_Q$ and $\hat{H}_T$ should be the same, i.e., one color quantization scheme is used for all images within the database.

A quantized histogram is usually represented by an N-dimensional vector, where N is the total number of quantization bins. For example, an RGB color histogram, which has been quantized into k bins for R, l bins for G, and m bins for B, can be represented as a vector of dimension $N = klm$. Different similarity metrics for histograms have been studied. One example is the histogram intersection method by Swain and Ballard [10]:

$$\frac{\sum_{i=1}^{N} \min\left(\hat{H}_Q(i), \hat{H}_T(i)\right)}{\sum_{i=1}^{N} \hat{H}_Q(i)}$$

where $\hat{H}_Q(i)$ and $\hat{H}_T(i)$ denote, respectively, numbers of pixels in the query and target images of the same bin with index i. Another similarity metric [22], which takes into account the perceptual similarity between bins of histograms, was proposed by Hafner *et al.* [22]. It is of the form

$$\begin{aligned}\text{dist}\left(\hat{H}_Q, \hat{H}_T\right) &= \left(\hat{H}_Q - \hat{H}_T\right)^T \boldsymbol{A}\left(\hat{H}_Q - \hat{H}_T\right) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}\left(\hat{H}_Q(i) - \hat{H}_T(i)\right)\left(\hat{H}_Q(j) - \hat{H}_T(j)\right)\end{aligned}$$

where matrix $\boldsymbol{A} = [a_{ij}]$ contains similarity weighting coefficients between colors corresponding to bins i and j.

B. Hierarchical Case

There are several drawbacks in the fixed quantization method. First, as an indexing feature, the discriminating ability of the color histogram is determined by the selection of the quantization method (i.e., color resolution). The computational complexity increases quickly as the resolution of color feature increases. This can be a major problem in applications where the desired performance requires a high resolution of color features. Second, the histogram obtained by a single resolution quantization is not efficient in the sense that many buckets are empty since it is often that colors in a given image only occupy a small subspace of the entire color space. Third, it is observed that results of quantized images are very sensitive to the location of quantization boundaries. Fig. 1 shows the histogram of two images similar in color. A uniform quantization scheme (two quantization levels for each component of RGB) is used to compute the histogram. The color of image A (127, 127, 127) is quantized to (0,0,0), the color of image B (128, 128, 128) is quantized to (1,1,1). Similar colors which are located at the different side of the quantization boundary are quantized into different bins. As a result, image B cannot be included in the similarity set of image A. To overcome these problems, it is desirable to develop a hierarchical feature representation and comparison scheme for image retrieval. In this work, we consider the following metric:

$$\text{dist}\,(H_Q, H_T)^{(k)} = \text{dist}\left(f_Q^{(k)}, f_T^{(k)}\right)$$

where $f_Q^{(k)}$ and $f_T^{(k)}$ represent color features of the query and target images at the kth resolution, respectively. Comparison

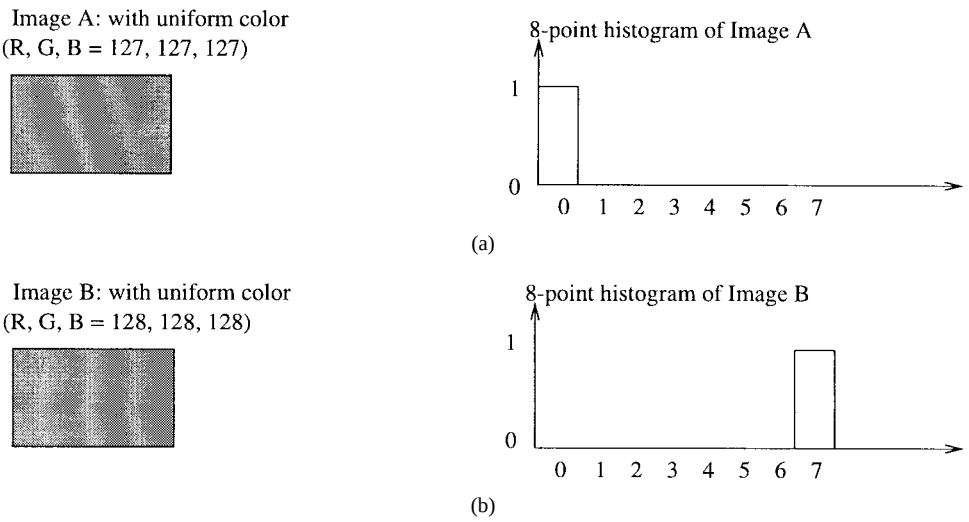


Fig. 1. Illustration of the histogram mismatching problem.

with the coarsest resolution feature can be used to get a candidate-image set with a very low computational complexity. Comparison of higher resolution features is then performed within the candidate-images set to reduce the computational cost.

To avoid the problem of putting similar colors into different buckets, we propose to use a color-clustering technique rather than the fixed quantization boundary in obtaining the color features $f_Q^{(k)}$ and $f_T^{(k)}$ at the kth resolution. The construction of a new color feature by hierarchical color clustering will be described in detail in Section IV.

III. Comparison of Image Retrieval with Different Color Quantization Schemes

To compare different quantization strategies, we perform a study on image retrieval based on global color histograms with different resolutions in four color spaces (i.e., RGB, YUV, HSV, and $CIEL^*u^*v^*$) in this section [20]. The similarity between histograms is measured with the histogram intersection method.

Our experimental database consists of 2119 images, including natural scenes, animals, plants, architectures, and people. Large varieties of our image database prevent the bias on a particular type of images. The same database is used in experiments reported for the rest of the paper. "Sunset" and "stained-glasses" image sets were used as query sets, with eight images in the "sunset" image set and five images in the "stained-glasses" image set. The "sunset" image set is selected because the images in this set are different from each other with slight changes in hue and brightness. It is desirable to test the robustness of color quantization schemes on such images. "Stained-glasses" images are selected because they contain rich colors in different hues, brightness, and saturations. Testing with these images will provide a reasonable comparison among different schemes without overemphasis on any particular color.

Retrieval results are evaluated by precision versus recall curves. Recall is the proportion of relevant images in the database that are retrieved in response to a query. In our experiment, it is the ratio of the number of retrieved "sunset" or the "stained-glass" image to the size of the query image set. Precision is the proportion of retrieved images that are relevant to the query. In our experiment, it is the ratio of the number of retrieved "sunset" or "stained-glass" images to the total number of retrieved images. The ideal result would be that all "sunset" were ranked at the top eight positions and all "stained-glasses" images ranked at the top five positions for respective queries. With such an outcome, the precision versus recall is a constant 1. The precision versus recall curves for different color quantization schemes are compared as shown in Figs. 2–4.

A. Color Distributions in Different Spaces

Color is a visual sensation produced by the light in the visible region of the spectrum incident on the retina. Since the human visual system has three types of color photoreceptor cone cells, three components are necessary and sufficient to describe a color. There are several systems of color spaces, such as $CIEXYZ, RGB, YUV, HSV, CIELAB, CIEL^*u^*v^*$, Munsell system, etc. Described below are the color systems selected for our study.

- RGB: Digital images are normally represented in the RGB space. CRT's also use the RGB system to display a color pixel on the screen by excitations of red, green, and blue phosphors.
- YUV: The YUV space is widely used in image compression and processing applications. Y represents the luminance of a color, while U and V represent the chromaticity of a color. The luminance component is separated from the chrominant components in this space.
- $CIEL^*v^*v^*$: Color differences in an arbitrary direction are approximately equal in the $CIEL^*u^*v^*$ space. Thus, the relative distance of two colors can be determined by the Euclidean distance.
- HSV: HSV (hue, saturation, value) provides an intuitive color space. Each component in this space contributes directly to visual perception. Other similar color spaces

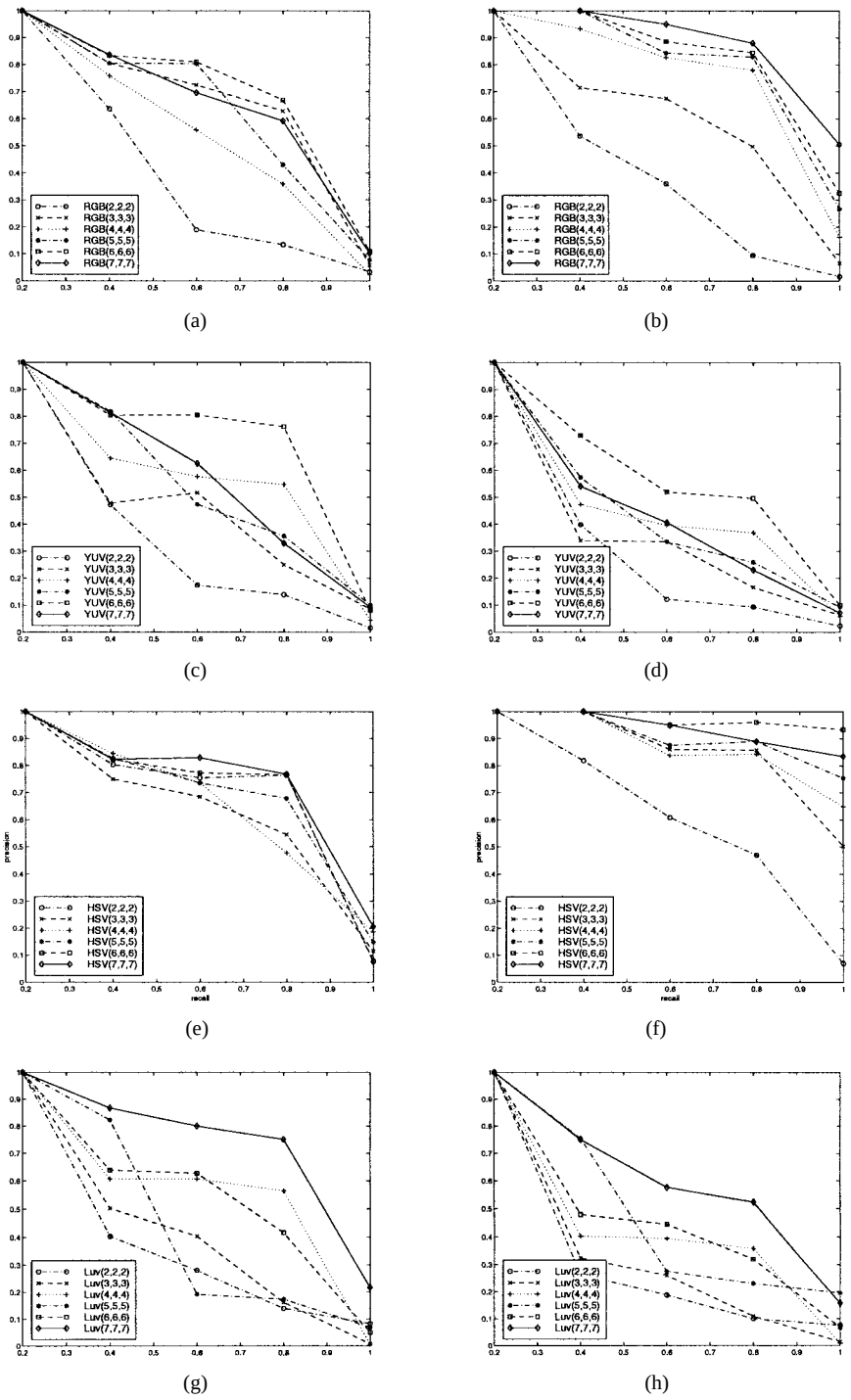


Fig. 2. Precision versus recall using uniform quantization schemes for (a) "sunset" in RGB, (b) "stained glass" in RGB, (c) "sunset" in YUV, (d) "stained glass" in YUV, (e) "sunset" in HSV, (f) "stained glass" in HSV, (g) "sunset" in Luv, and (h) "stained glass" in Luv.

include HSI and HSL, where I and L denote intensity and lightness, respectively.

Color distributions of our test image database with respect to these color spaces are shown in Figs. 5–8.

B. Color Quantization Schemes

1) Uniform Quantization: In uniform quantization, each axis of the color space is uniformly divided into a certain number of bins. It has been shown in the previous section that color distributions are often nonuniform, and therefore, a simple uniform quantization scheme is inefficient for some color spaces. The advantage of uniform quantization is that it is a straightforward and natural choice in the absence of *a priori* information about the color distribution of the image database. Generally, the retrieval performance gets better as the number of quantization bins increases, as shown in Fig. 2. Some exception may occur when similar colors under the same subset of a coarse quantization scheme are divided into two different subsets of a finer quantization.

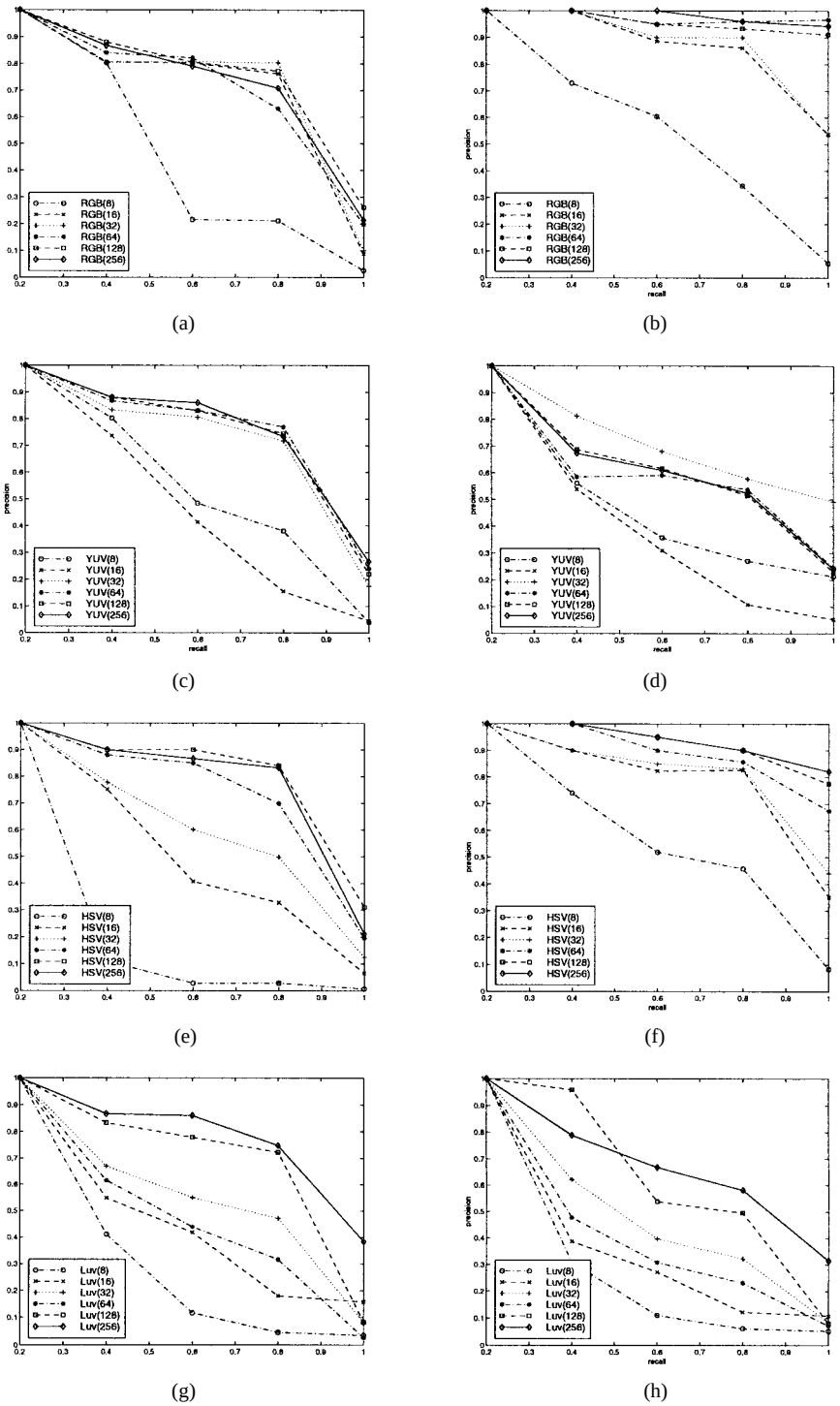


Fig. 3. Precision versus recall using vector quantization schemes for (a) "sunset" in RGB, (b) "stained glass" in RGB, (c) "sunset" in YUV, (d) "stained glass" in YUV, (e) "sunset" in HSV, (f) "stained glass" in HSV, (g) "sunset" in Luv, and (h) "stained glass" in Luv.

2) Standard Vector Quantization (VQ): The standard VQ is a method of partitioning the vector space by minimizing the mean-squared error (MSE) with respect to a set centroid points (i.e., codewords). This procedure can be achieved with a proper initialization and iteration by using the generalized Lloyd–Max algorithm (GLA). When applied to color space quantization, VQ divides the color space into a prespecified number of subspaces so that the resulting quantization error of all pixels of all images with respect to quantized color representations in the database is minimized. Compared with uniform quantization, the histogram bins are optimally selected in VQ so that the same number of bins can lead to a smaller quantization error at the expense of higher processing complexity. As shown in Fig. 3, the retrieval performance of VQ is better than that of uniform quantization as the quantization level increases. It is interesting to point out that the retrieval performance cannot be further improved when the number of quantization bins is larger than a certain threshold.

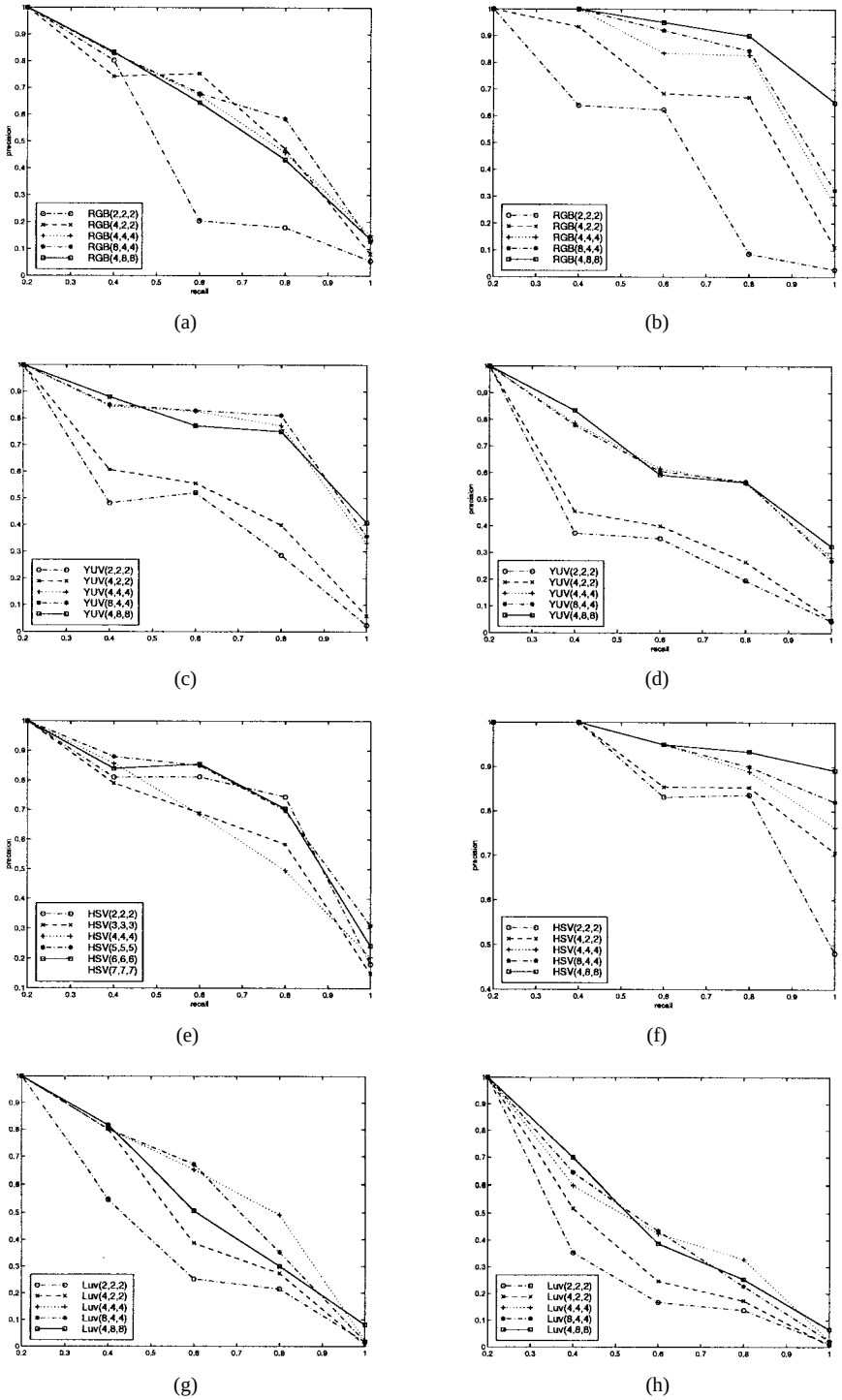
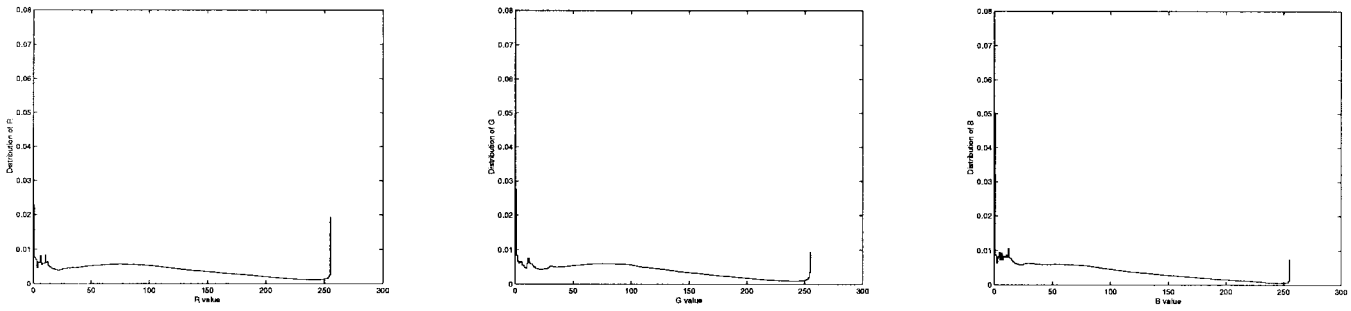


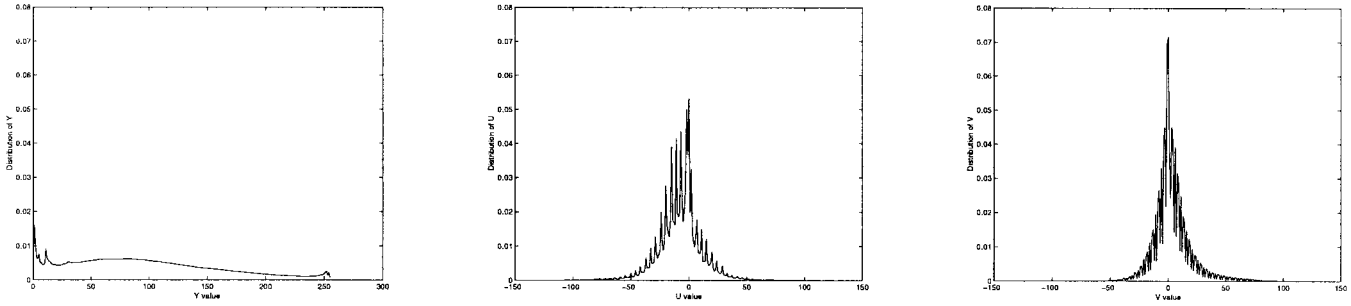
Fig. 4. Precision versus recall using product VQ schemes for (a) "sunset" in RGB, (b) "stained glass" in RGB, (c) "sunset" in YUV, (d) "stained glass" in YUV, (e) "sunset" in HSV, (f) "stained glass" in HSV, (g) "sunset" in Luv, (h) "stained glass" in Luv.

3) Product Vector Quantization: Although VQ results in an optimal partitioning of the color space, it is not suitable for large image databases due to the high computational complexity. A simpler but suboptimal quantization method can be adopted to reduce the complexity. That is, we can consider partitioning perpendicular to the axis of the color space. This method is known as product vector quantization. In particular, the Lloyd–Max quantizer can be applied to each axis of the color space separately to optimize the mean-square quantization error along each axis independently so that the computational complexity of the histogram extraction can be reduced. The product vector quantization with the Lloyd–Max quantizer results in a better partitioning of the color space as compared to uniform quantization without a tremendous increase in computational complexity. As shown in Fig. 4, the retrieval performance in the RGB space does not increase much by using product VQ due to its nearly uniform color distributions. In contrast, the retrieval performance in



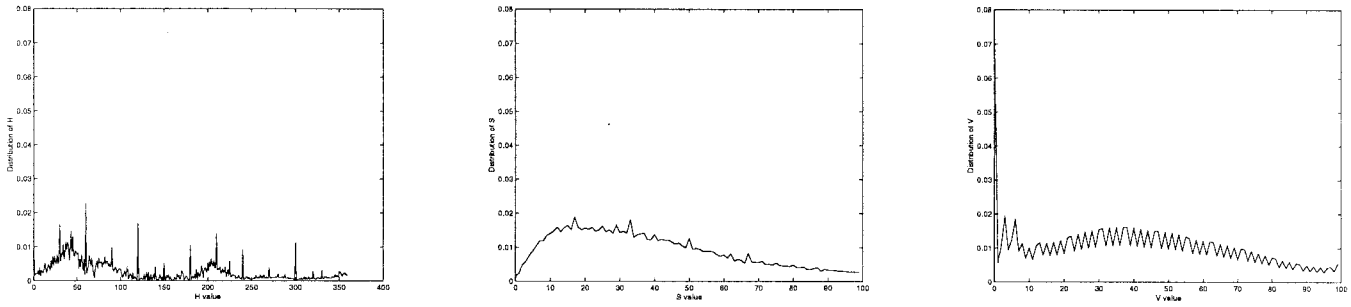
(a) (b) (c)

Fig. 5. Color distribution in the RGB space.



(a) (b) (c)

Fig. 6. Color distribution in the YUV space.



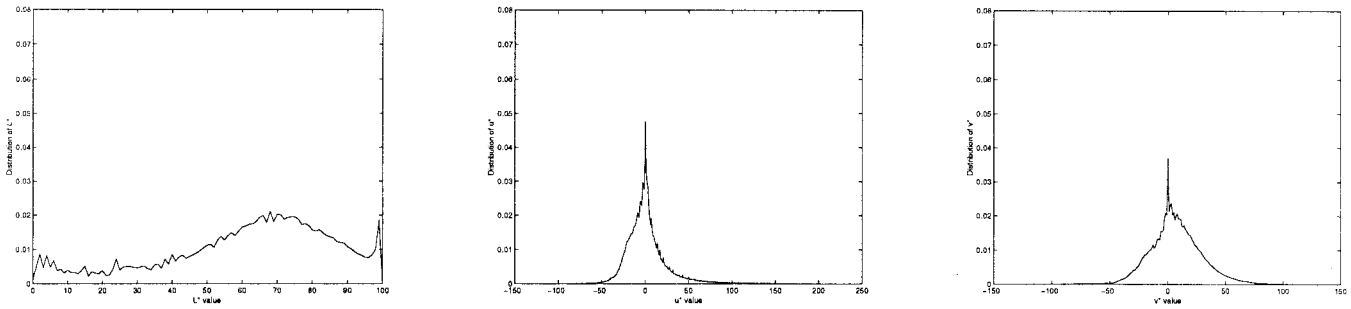
(a) (b) (c)

Fig. 7. Color distribution in the HSV space.

the HSV space is better, especially when the number of quantization bins is small.

C. Quantization Schemes in Different Color Spaces

The RGB space can be uniformly quantized into $N \times N \times N$ bins because the distributions of all three color components are almost flat, except at two endpoints which correspond to the black and white colors contributed mostly by the image background (see Fig. 5). For the YUV space, as shown in Fig. 6, nonuniform quantization schemes are needed for good performances. Product VQ might be a good choice because the U and V components can be modeled by the Laplacian distribution and the Y component can be uniformly quantized for simplicity. The uniform quantization does not work well for the $CIEL^*u^*v^*$ space either (see Fig. 8). A sophisticated quantization scheme based on vector quantization (VQ) can lead to the best performance because of the linear color difference property of this space. For the HSV space, the distribution of its color component suggests a product VQ scheme, where H is quantized with a resolution finer than S and V because the human visual system is more sensitive to the hue than to the saturation and the value (see Fig. 7). It is interesting to point out that the distribution of H peaks every 30° for our experimental database, so that the best quantization should be multiples of 30°. The distribution of V is flat, except for the region around zero, which is the background of an image where a uniform quantization may be appropriate. In the distribution of S, lower saturation bins have higher probability than higher saturation bins; however, because the human visual system is more sensitive to colors with a higher saturation, uniform quantization may also be appropriate. For example, this space can be simply $6 \times 2 \times 2$ or $12 \times 5 \times 5$ uniformly quantized along each axis.



(a) (b) (c)

Fig. 8. Color distribution in the $CIEL^*u^*v^*$ space.

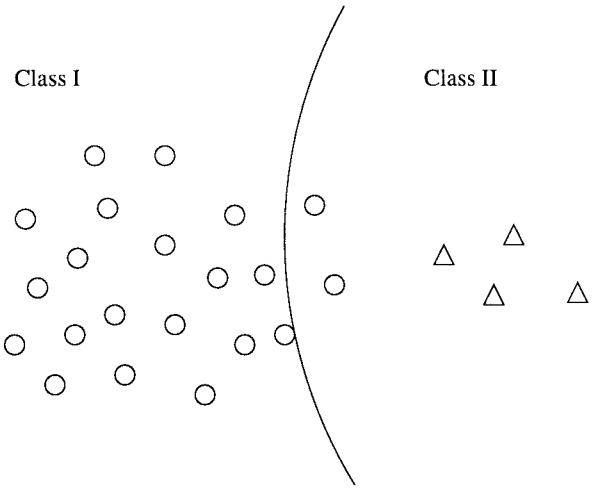


Fig. 9. Example of misclassification based on the mean-squared error classification criterion.

IV. Hierarchical Color Clustering

In the above discussion, we attempted to determine good single-resolution quantization schemes for different color spaces. There are however problems. Given a set $\mathcal{X}$ of n samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$, the objective of VQ is to find c representative vectors which minimize the averaged quantization error. However, a good number of natural color clustering usually varies with images. If the selected value of c is not equal to the number of natural color clusters for a given image, similar colors might lie across boundaries of quantization bins, and false misses might happen as a result in the retrieval process. Furthermore, a large cluster might be split to maintain the minimum mean-square error as shown in Fig. 9. We propose to use a set of new features based on hierarchical color clustering which represents the natural color feature of each individual image in this section to overcome these problems. Also, this new color feature extraction scheme provides a balance between the performance in terms of discriminant capability and the computational complexity.

A. Clustering Strategy

The target of clustering is to partition a set $\mathcal{X}$ of n samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ into c disjoint subsets. There are two families of unsupervised clustering techniques: heuristic methods and criterion-based methods [23]. Heuristic clustering methods include the chain rule, hierarchical clustering, graphic-theoretic methods, etc. Criterion-based methods are often developed via iterative optimization of a certain cost function, which requires a high computational complexity. Several widely used criteria for single resolution clustering are described as follows.

- Sum of Squared Errors:

$$J_e = \sum_{i=1}^{c} \sum_{x \in \mathcal{X}_i} ||x - m_i||^2$$

  where $m_i = n^{-1} \sum_{x \in \mathcal{X}_i} x$. For a given cluster $\mathcal{X}_i$, the mean vector m_i is the best representative of samples in $\mathcal{X}_i$ in the sense that it minimizes the sum of squared errors.
- Related Minimum Variance:

$$J_r = 2^{-1} \sum_{i=1}^{c} n_i s_i$$

  where s_i can be either the average squared distance between points in the ith cluster or the median (or maximum) distance between points in a cluster.
- Scattering Measures: The within-cluster scatter matrix is defined as

$$S_W = \sum_{i=1}^{c} n_i \sum_{x \in \mathcal{X}_i} (x - m_i)(x - m_i)^t$$

  while the between-cluster scatter matrix is defined as

$$S_B = \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^t.$$

  Criteria derived from scatter matrices are their traces, determinants, and other invariant measures.

Since the number c of clusters is unknown, a series of cost functions $J^{(c=i)}$ with $i = 1, 2, 3, \cdots$ has to be calculated to determine its value. Clearly, $J^{(c=i)}$ deceases monotonically with the increment of c. If the samples are naturally grouped into $\hat{c}$ compact and well-separated clusters, $J^{(c=i)}$ decreases fast when $i \leq \hat{c}$ and slowly when $i > \hat{c}$. To avoid difficulties in determining c, a hierarchical clustering method is proposed in this section. The proposed scheme consists of two stages. First, we use the octree color quantization to get the initial hierarchical clustering. Then, we adopt an agglomerative hierarchical method for octree pruning. They are detailed below.

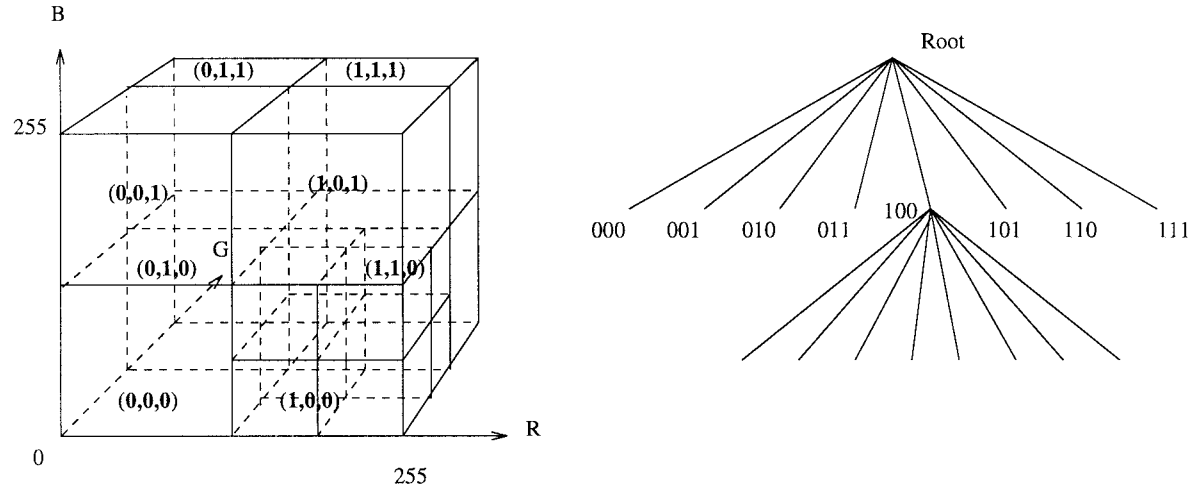


Fig. 10. Illustration of the octree structure.

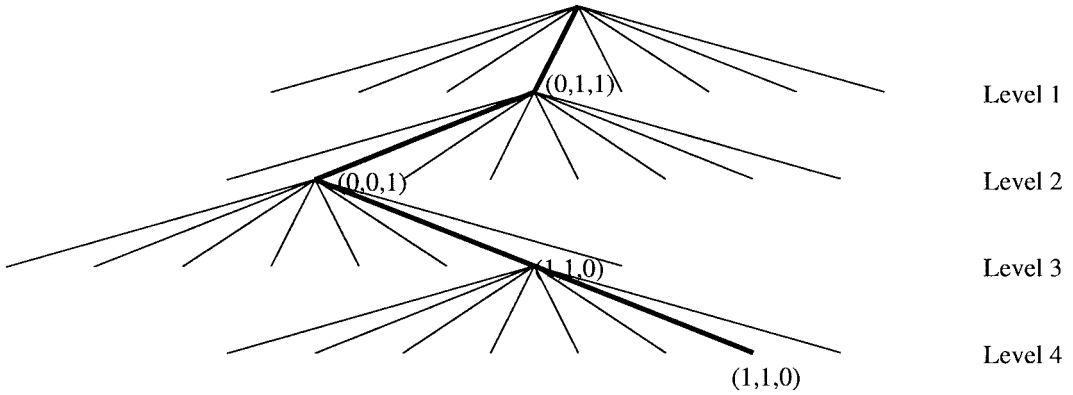


Fig. 11. Example of inserting a color point with $R = 53$, $G = 187$, $B = 197$ into the octree.

B. Hierarchical Clustering with Pruned Octree

The octree color quantization algorithm proposed by Gervautz and Purgathofer [24] is a two-pass color quantization procedure for the purpose of color display. In the first pass, each pixel of a 24-bit color image is scanned and inserted into the octree data structure. Tree reduction is performed whenever the total amount of inserted colors exceeds the predetermined limit N (normally 256), with the assumption that N colors are sufficient to support the color appearance of an individual image. Colors of leaf nodes form the color palette. In the second pass, the image is scanned again and the color of each pixel is traced from the root to the leaf of the octree. The original color of the pixel is quantized to the color of a leaf node. In this work, we use the octree data structure to speed up hierarchical clustering process.

1) Octree Structure and Insertion: Fig. 10 shows the structure of an octree in the RGB color space. At the first level of the tree, the eight children of the root correspond to the eight subspaces of the entire space. Similarly, each of the eight nodes can have its own eight children corresponding to further divided subspaces. The maximum depth of the octree for a 24-bit image is 8, and each leaf corresponds to one of $16\,777\,216 = 8^8$ colors. Each color defines a path from the root to the leaf. Two quantities can be used to describe the color information within a subspace (i.e., each node). They are the number of pixels passing through the node and the average color of these pixels. In other words, we say that each node has two attributes, i.e., the normalized pass number p (in term of %) and the average color $\vec{C}$ of pixels. When a color is inserted to the octree, its path is traced, and attributes of intermediate nodes are modified accordingly. Similar colors will share a common path up to a certain intermediate node so that color quantization can, in fact, be achieved by mapping similar colors to the color of the shared intermediate node.

The octree of an image can be obtained by scanning the color value of each pixel and inserting it into the octree one by one. The easiest way to explain the insertion process is to consider an example as shown in Fig. 11, where the procedure of inserting a pixel with color components $R = 53$ (00 110 101 in binary), $G = 187$ (1 011 101 in binary), and $B = 197$ (11 001 111 in binary) into a eight-level octree is illustrated. At the first level, the color is located in subspace $\{(R,G,B) : R \in [0,127], G \in [128,255]$, and $B \in [128,255]\}$, which corresponds to child (0,1,1) of the root node. The subspace $\{(R,G,B) : R \in [0,127], G \in [128,255]$, and $B \in [128,255]\}$ is further divided to eight subspaces. At the second level, the color belongs to one of the eight further divided subspaces $\{(R,G,B) : R \in [0,64], G \in [128,128+63], B \in [128+63,255]\}$, which corresponds to

node (0,0,1), one of the child of node (0,1,1). By repeating this procedure, we can locate the pixel to the last level of the tree.

2) Octree Pruning and Hierarchical Clustering: Clearly, a full octree is too large to be stored in the memory. In the method of Gervautz and Purgathofer, node shrinking is performed whenever the total number of leaf nodes exceeds a threshold. With this process, we observed that the octree depth of almost all images is less than or equal to 4. Thus, we limit the maximum level of the tree to 4 so that there are at most $\sum_{i=0}^{i=4} 8^i = 4681$ nodes, which is not too large for temporary storage and manipulation. We will introduce pruning techniques in this subsection to further simplify the octree. It is worthwhile to emphasize that insertion is implemented before any pruning in our scheme to guarantee the uniqueness of the octree representation of a given image. In contrast, insertion and pruning are performed simultaneously in the work of Gervautz and Purgathofer.

We consider a two-step pruning process. In the first step, we perform a vertical pruning in which leaf nodes will be merged to their parent node if the parent node of concern has a small pass number. This is often called "tree shrinking." The main objective of tree shrinking is to reduce the number of leaf nodes to a reasonable size (e.g., ≤ 256). In the second step, we apply a horizontal pruning in which leaf nodes are merged if their distance in terms of average colors is small, and the merge of leaf nodes with different parents and at different levels is allowed. There are two main objectives in the second step. First, reclustering is performed to remove rigid boundaries created in the initial octree quantization process. Second, reclustering is achieved under a sequence of thresholds to generate different sets of leaf nodes of different resolutions.

To implement tree shrinking (i.e., the first-step pruning), we rank parent nodes of all leaf nodes according to their pass numbers. The pruning process begins with the parent node with the smallest pass number. Since its pass number is small, its children should have even smaller pass numbers. Thus, we can simply delete the original leaf nodes, and convert their parent node into a leaf node. Consequently, the total number of leaf nodes is reduced. The above process is repeated until the total number of leaf nodes is equal to or less than a preset number (256 in our implementation).

To implement the second-step pruning, we consider a sequence of merge processes determined by a set of increasing thresholds $T_k, k = 1, 2, 3, \cdots$. We start with the merge process with $k = 1$, and search the nearest pair of distinct clusters. If their distance in terms of average colors is less than T_1, these two clusters are merged. This process is repeated until the distances between all distinct clusters are all greater than T_1. Then, by using T_2 as the new threshold, we repeat the same merge operations until all clusters have a distance greater than T_2. This procedure is continued for $k = 3, 4, \cdots$. The average color and the pass number of leaf nodes at the end of each merge process are recorded as the color feature in different resolutions. The reason to divide the second-step pruning into a sequence of successive merge processes is that it allows us to select hierarchical features for sequential matching and filtering so that the computational complexity of image retrieval can be greatly reduced. Note that it requires $O(c_i c_j)$ operations to compare the similarity of two images with cluster numbers c_i and c_j, respectively. Hierarchical features can help in filtering out irrelevant images as early as possible with a very low computational cost.

One of the key issues here is the computational complexity of the reclustering step. A straightforward way to compute the distance of all cluster pairs is $O(N^2)$, where N is the number of leaf nodes. However, a divide-and-conquer method can be used here to reduce this complexity to $O(N \log N)$. Its basic idea is described as follows. The two leaf nodes with the shortest distance contained by the space represented by the root node have to be one of the following two situations. It can be either the leaf node pair with the shortest distance contained by one of its eight subspaces or two leaf nodes belonging to two different subspaces. For the former case, the shortest distance among each subspace can be computed recursively by treating the corresponding subspace as the root node. Let us assume that these distances are denoted by δ_1, $\delta_2, \ldots, \delta_8$ and let $\delta = \min_i(\delta_i)$. To take care of the latter case, we have to pay attention to leaf nodes located away from the boundaries of the subspace with a distance less than δ. That is, distances between every pair of these leaf nodes also have to be computed. By comparing results from the two cases, we can determine the leaf node pair with the smallest distance.

The closest pair of nodes and their distance within each subspace are the attributes of the corresponding intermediate node. The nearest pair attribute of the root node is the nearest pair of nodes within the entire color space. If these two nodes are merged, their ancestors have to be adjusted accordingly to make sure the average color and the pass number of these nodes are updated. The new closest pair of nodes and their distance should also be updated in order to carry out the next merging process.

An illustration of the merging process is shown in Fig. 12. In this example, Node_$(4, m)$ is merged with Node_$(3, n)$. In this case, since the new node is located in subspace $(4, m)$, its average color becomes

$$\vec{C}_{4,m} \leftarrow \left(\vec{C}_{4,m} * N_{4,m} + \vec{C}_{3,n} * N_{3,n}\right)/(N_{4,m} + N_{3,n})$$

and its pass number is updated to be

$$N_{4,m} \leftarrow N_{4,m} + N_{3,n}.$$

The average color and the pass number of ancestors of Node_$(4, m)$ and Node_$(3, n)$ should be updated accordingly. As shown in the figure, Node_$(3, m')$, Node_$(2, m'')$, Node_$(1, m''')$, Node_$(2, n'')$, and Node_$(1, n''')$ are their uncommon ancestors, while the root is their common ancestor. We only have to update uncommon ancestors with the following rules:

$$\vec{C}_{3,m'} \leftarrow \left(\vec{C}_{3,m'} * N_{3,m'} + \vec{C}_{3,n} * N_{3,n}\right)/(N_{3,m'} + N_{3,n}),$$
$$N_{3,m'} \leftarrow N_{3,m'} + N_{3,n}$$
$$\vec{C}_{2,m''} \leftarrow \left(\vec{C}_{2,m''} * N_{2,m''} + \vec{C}_{3,n} * N_{3,n}\right)/(N_{2,m''} + N_{3,n}),$$
$$N_{2,m''} \leftarrow N_{2,m''} + N_{3,n}$$
$$\vec{C}_{2,n''} \leftarrow \left(\vec{C}_{2,n''} * N_{2,n''} - \vec{C}_{3,n} * N_{3,n}\right)/(N_{2,n''} - N_{3,n}),$$

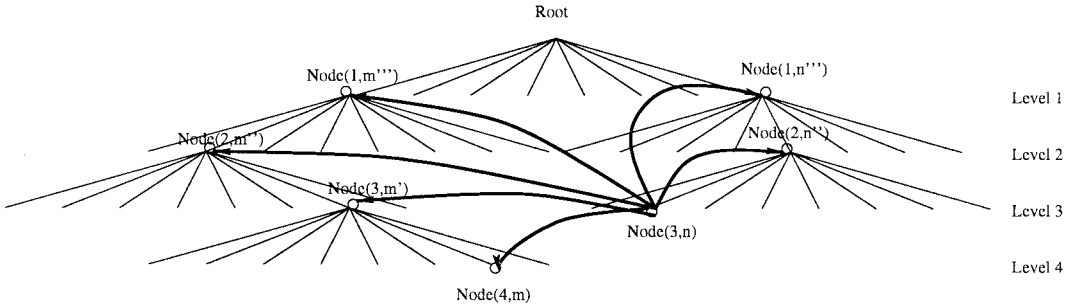


Fig. 12. Illustration of the node-merging process.

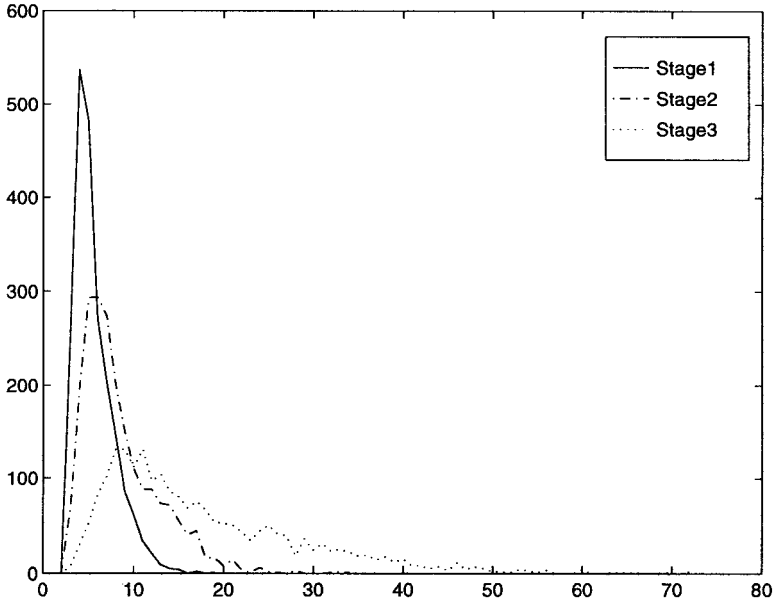


Fig. 13. Distribution of the number of clusters at different resolutions.

$$N_{2,n''} \leftarrow N_{2,n''} - N_{3,n}$$
$$\vec{C}_{1,m'''} \leftarrow \left(\vec{C}_{1,m'''} * N_{1,m'''} + \vec{C}_{3,n} * N_{3,n}\right)/(N_{1,m'''} + N_{3,n}),$$
$$N_{1,m'''} \leftarrow N_{1,m'''} + N_{3,n}$$
$$\vec{C}_{1,n'''} \leftarrow \left(\vec{C}_{1,n'''} * N_{1,n'''} - \vec{C}_{3,n} * N_{3,n}\right)/(N_{1,n'''} - N_{3,n}),$$
$$N_{1,n'''} \leftarrow N_{1,n'''} - N_{3,n}.$$

The hierarchical clustering procedure in the second-step pruning is implemented with $k = 1, 2, 3$ and $T_1 = 16, T_2 = 24$ and $T_3 = 32$. The number of layers and the thresholds are obtained empirically by experiments. We show in Fig. 13 the distribution of the leaf node number for images in our database with respect to each threshold value. The average numbers of clusters (i.e., leaf nodes) are 17.7, 8.48, 5.59, respectively. We record the pass number (1 byte) and the average color (3 bytes) for these clusters as features. Thus, on the average, the indexing file size is $4 \times (17.7 + 8.48 + 5.59) = 124$ bytes per image. The clustering result at different resolutions of the "Sunset" image is shown in Fig. 14.

C. Hierarchical Color Clustering in the $L^*u^*v^*$ Space

Many digital images are represented in the RGB space. However, the color difference computed using the Euclidean distance in the RGB space is not totally consistent with the human visual system (HVS) model. That is, two colors with a large Euclidean distance in the RGB spaces may be perceptually similar. Quantitative measurement of the color distance has been studied extensively, and psychophysical experiments were conducted to determine the just noticeable color differences (JNCD). It is well known that JNCD is not uniform along the three axes in the RGB space. The infinitesimal color difference ds of two neighboring colors can be written as

$$ds^2 = \sum_{i,j=1}^{3} C_{i,j}\, dx_i\, dx_j$$

where metric coefficients $C_{i,j}$ depend on x_i and x_j. To find the difference between two colors, we have to integrate the above equation from one color to the other. The integral is path dependent, and the actual distance is defined to be the integral along the path which yields the minimum distance between these two colors. Since the computational cost is high, an alternative approach is to map the RGB space onto another space with a uniform color difference. Several such spaces have been proposed, including $CIEL^*a^*b^*$, $CIEL^*u^*v^*$, and Munsell color spaces [25]. The Munsell color space was

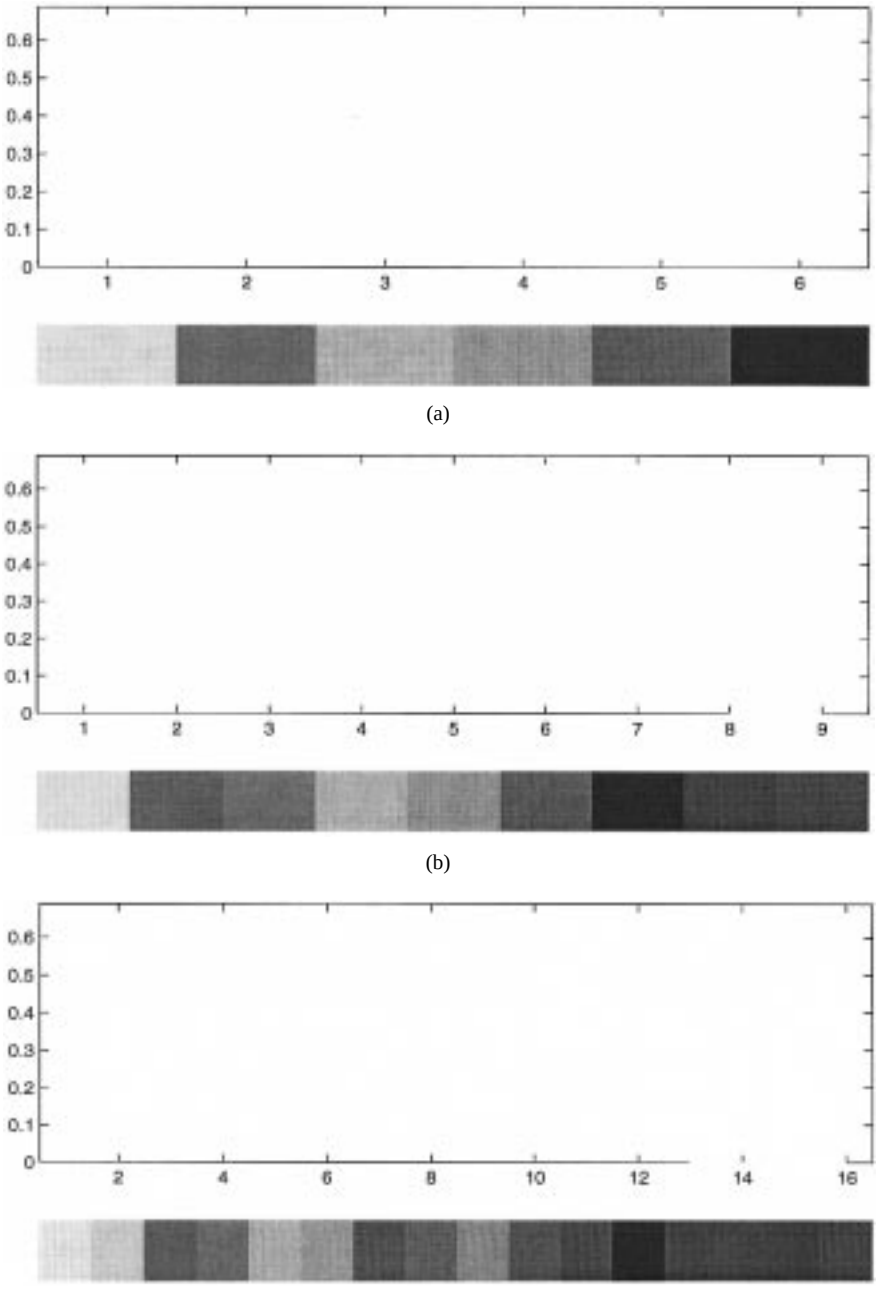


Fig. 14. Color clustering of "sunset" image with three different resolutions.

named after artist Albert Munsell who created a book of colored samples ordered by the constant hue, brightness, and saturation chart. The $L^*a^*b^*$ space was developed to provide a computationally simple measure of colors in agreement with the Munsell space. The $L^*u^*v^*$ space was evolved from the $L^*a^*b^*$ space, and became the CIE standard in 1976.

The transform from the RGB space to the $L^*u^*v^*$ space is

$$
\begin{aligned}
X &= 0.607R + 0.174G + 0.200B \\
Y &= 0.299R + 0.587G + 0.114B \\
Z &= 0.000R + 0.066G + 1.116B \\
u' &= \frac{4X}{X + 15Y + 3Z} \\
v' &= \frac{9Y}{X + 15Y + 3Z} \\
L^* &= \begin{cases} 116[Y/Y_0]^{1/3} - 16, & \text{if } Y/Y_0 \geq 0.008856 \\ 903.3Y/Y_0, & \text{otherwise} \end{cases} \\
u^* &= 13L^*(u' - u'_0) \\
v^* &= 13L^*(v' - v'_0)
\end{aligned}
$$

where u'_0 and v'_0 are obtained with reference to white tristimulus values X_0, Y_0, Z_0.

The gamut of the $L^*u^*v^*$ space translated from the RGB space is shown in Fig. 15. As seen from the figure, not all combinations of hue, chroma, and value are within the gamut.

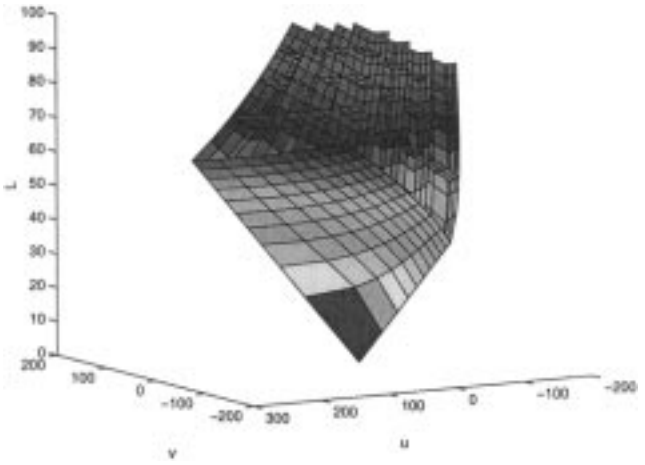


Fig. 15. Gamut of $L^*u^*v^*$ color space and its partitioning planes.

Splitting the color space by a set of planes perpendicular to these axes is not efficient because many subspaces will be empty. It is desirable to develop a splitting scheme suitable to the irregular volume of the $L^*u^*v^*$ space. One possible solution is to map perpendicular splitting planes in the RGB space to the $L^*u^*v^*$ space. We illustrate in Fig. 15 the partitioning planes corresponding to the second level of the octree.

In actual implementation, we do not have to perform the transform and initialize the octree in the $L^*u^*v^*$ space by comparing the value of pixels with boundaries defined by slitting planes. Instead, the RGB representation is used when inserting a color into the octree. However, the average color of pixels in each node is calculated with the $L^*u^*v^*$ representation. It is worthwhile to mention that when the L component is very small, two colors might be perceptually similar, but with a large color distance if their u and v components are very different. To fix this problem, it is convenient to set these colors to black before clustering.

The clustering procedure in the $L^*u^*v^*$ space is summarized as follows.

1) Octree initialization and shrinking
   a) Insert a new pixel in the image from the root to the leaf, and update the pass number and the average color of nodes accordingly based on its RGB color representation. Repeat the process until all pixels are inserted.
   b) If the number of leaf nodes is greater than C ($C = 256$), shrink the tree.
2) Hierarchical clustering
   a) Set the average color of the node whose lightness is less than L_0 ($L_0 = 50$) to black.
   b) Perform the following steps for $k = 1, 2, 3$. Find the nearest pair of distinct clusters based on their means. If their distance is less than T_k, they are merged. This process is repeated until the distance of all distinct pairs is greater than T_k.

Again, $T_1 = 16$, $T_2 = 24$, and $T_3 = 32$ are used in our implementation.

V. Indexing and Retrieval Based on Hierarchical Clustering

A. Indexable Features and Distance Computation

We can derive a set of interesting features based on the hierarchical color clustering feature to speed up the retrieval. They include the following.

- **Average color** (stored with 3 bytes): The average color of the entire image corresponds to that of the root of the octree. The distance between the average colors of the query (Q) and an image in the database (T) is

$$d_{\text{avg}}(Q,T) = d_{\text{Euclidean}}\left(\vec{C}_{0,0}^{(Q)}, \vec{C}_{0,0}^{(T)}\right) = \sqrt{\sum_{i=1}^{3}\left(C_{0,0}^{(Q)}[i] - C_{0,0}^{(T)}[i]\right)^2}.$$

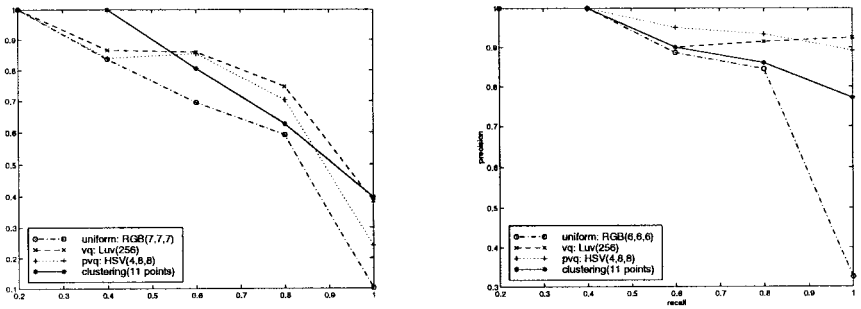
  The average color distance and dominant color distance are computed using Euclidean distance because it is consistent with the human visual perceptual model in the L*u^*v^* space.
- **Dominant color** (stored with 3 bytes): The average color of the node with the largest pass number at the coarsest resolution gives the dominant color of an image. The distance between the dominant colors of the query and target images can be computed via

$$d_{\text{dom}}(Q,T) = d_{\text{Euclidean}}\left(\vec{C}_{k,m}^{(Q)}, \vec{C}_{k,n}^{(T)}\right) = \sqrt{\sum_{i=1}^{3}\left(C_{k,m}^{(Q)}[i] - C_{k,n}^{(T)}[i]\right)^2}, \qquad k = 3$$

  where $\vec{C}_{k,m}^{(Q)}$ and $\vec{C}_{k,n}^{(T)}$ are average colors of dominant nodes of images Q and T at the coarsest resolution, which is 3 in our implementation, respectively.
- **Color width** (stored with 1 byte): The number of leaf nodes at the finest resolution (i.e., $k = 1$) is called the color width of a given image. It indicates the richness of colors. The distance based on the color width is defined as

$$d_{\text{width}}(Q,T) = |W^{(Q)} - W^{(T)}|$$

  where $W^{(Q)}$, $W^{(T)}$ are widths of octrees of images Q and T.
- **Hierarchical color distributions** (stored on the average with 124 bytes per image in our test database): The average color and the pass number of leaf nodes at each merging process lead to a set of hierarchical color distributions. Since the number of clusters and the position of clusters are different from images to images, we have to define the distance between two sets of clustered nodes. Let k represent the resolution level, and $\mathcal{Q}$ and $\mathcal{T}$ denote, respectively, node sets for query image Q and target image T at level k. Nodes from $\mathcal{Q}$ and $\mathcal{T}$ are said to have a match if their distance is less than T_k. Let $\mathcal{M}$ be the set of all matched nodes in $\mathcal{Q}$ and $\mathcal{T}$ at resolution k.



(a) (b)

Fig. 16. Retrieval result comparison for (a) "sunset" and (b) "stained glass."

Then, the distance can be defined as

$$d_{\text{layer}(k)}(Q,T) = \sum_{\mathcal{M}_i \in \mathcal{M}} \left| \sum_{m \in \mathcal{M}_i} p_m^{(Q)} - \sum_{\hat{m} \in \mathcal{M}_i} p_{\hat{m}}^{(T)} \right| + \sum_{u \in \mathcal{Q} - \mathcal{M}} p_u^{(Q)} + \sum_{u \in \mathcal{T} - \mathcal{M}} p_u^{(T)}$$

where $p_m^{(Q)}$ and $p_{\hat{m}}^{(T)}$ are the normalized pass numbers of nodes m and $\hat{m}$ in the same matched set $\mathcal{M}_i$ and belonging to $\mathcal{Q}$ and $\mathcal{T}$, respectively, and $p_u^{(Q)}$ and $p_u^{(T)}$ are the normalized pass-numbers of nodes u in unmatched sets $\mathcal{Q} - \mathcal{M}$ and $\mathcal{T} - \mathcal{M}$, respectively.

As described above, we need 131 $(= 3+3+1+124)$ bytes in total to store the average color, the dominant color, the color width, and the hierarchical color distribution for each image in the database. This is about one half the storage required by the traditional histogram method with 256 quantization bins where 1 byte is used to record the normalized pixel numbers in each bin.

B. Retrieval Examples

Each indexing feature mentioned above carries interesting color information of an image. Filtering by a selected set of simple features such as the average color, the dominant color, and the color width can be performed first to remove irrelevant images. This is particularly useful if the query image has certain prominent features, e.g., a clear dominant color and an unusual color width. Filtering based on the comparison of hierarchical color distributions can be performed at a later stage to refine the candidate image set which contains similar images.

We use several examples below to demonstrate this idea. The experimental database is the same as the one given in Section III. We consider three image sets, i.e., "skiing," "stained-glasses," and "sunset," and use one from each image set as the query image. The comparison of retrieval results is shown in Fig. 16. The query image for each image set is shown in Fig. 17.

Retrieval of "Skiing" Image: Each image in the "Skiing" image set is dominated by the white tone. The dominant color and the percentage of pixels possessing this color are shown in Table I. Retrieval by the dominant color alone can promptly get a very small candidate image set.

(a) (b) (c)

Fig. 17. Query images in (a) "skiing," (b) "stained-glass," and (c) "sunset" image sets.

TABLE I
DOMINANT COLOR OF IMAGES IN "SKIING" SET

Images	Dominant color (Luv)	np	Images	Dominant color (Luv)	np
Ski_0	(86,-18,-19)	0.9135	Ski_5	(98, -8, -5)	0.8999
Ski_1	(96, -9, -3)	0.9077	Ski_6	(87,-14,-12)	0.9431
Ski_2	(92,-11, -8)	0.8667	Ski_7	(87,-22,-25)	0.9121
Ski_3	(91,-14, -7)	0.9490	Ski_8	(91, -10, -9)	0.9283
Ski_4	(89, -10, -4)	0.9334	Ski_9	(86, -10,-8)	0.8376

Retrieval of "Sunset" Image: Each image in this set has a dominant color, but their dominant colors are not very similar. For example, some images are dominated by dark red, while

other images are dominant by dark yellow. Thus, the threshold for filtering with the dominant color has to be set a larger value to avoid false misses and, as a result, the candidate image set is still large. Filtering by hierarchical color distributions can be applied to this set of candidate images. Retrieval results are compared with those of other methods in Fig. 16. The amount of clusters at the third stage of hierarchical clustering for "sunset" image set is 11. Retrieval by this small size of color representation performs reasonably well compared with other methods.

Retrieval of "Stained-Glasses" Image: The color width of the query image is 71, which is very large in comparison with most images in the database. The distribution of the color width of images in our test database at the each resolution of clustering can be found in Fig. 13. As seen from the figure, only a small number of images have a large width. Thus, filtering by the color width helps to narrow down the number of candidate images quickly. Filtering by the color width is ideal for images with rich or few distinctive colors. The precision versus recall curves are shown in Fig. 16.

VI. Conclusion

We investigated the effect of color quantization schemes on the performance of image retrieval, and proposed a new hierarchical color extraction and indexing scheme based on a pruned octree color representation. Color feature obtained by our scheme is more efficient than the color histogram in several aspects. First, it calculates the color feature of each individual image separately, and only a small number of distinctive colors and their corresponding pass numbers are used to describe the color feature of the image. Consequently, the color feature of each image is described more effectively with a smaller storage space. Second, there are no rigid quantization boundaries in quantizing similar colors so that we can get a more robust retrieval result with respect to small color differences among images. Third, more color features such as color width, dominant color, and average color can be obtained as the byproduct. The retrieval process can be speeded up by combining these features properly.

There are several related tasks to be performed in the near future. The proposed hierarchical color indexing file is much smaller than that of the traditional color histogram. As a result, we did not use sophisticated indexing methods such as R^* tree, $k-d$ tree in our retrieval experiment. However, the storage and retrieval efficiency can still be improved by organizing the structure of indexing files effectively. We would also like to consider the combination of color-based and object-based retrieval with the proposed hierarchical color clustering scheme. Object segmentation can be carried out using the octree data structure by projecting colors of distinctive clusters back to the original image. This back-projection process leads to an efficient initial segmentation of the image [26]. Further refinement using other methods, such as edge flow and region growing [27], [28], need to be carried out. Image segmentation using hierarchical color clustering by itself is an interesting research problem. Query analysis for interactive retrieval based on this color feature and other low-level image features such as textures and shapes of objects are our main research topics currently.

References

[1] W. Niblack, R. Berber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker, "The qbic project: Querying images by content using color, texture and shape," in *SPIE 1908, Storage and Retrieval for Image and Video Databases*, Feb. 1993, pp. 172–187.

[2] R. Jain, "Workshop report: NSF workshop on visual information management systems," in *Storage and Retrieval for Image and Video Databases*, SPIE, San Jose, CA, Feb. 1993, pp. 198–218.

[3] A. Pentland, R. Picard, and S. Sclaroff *et al.*, "Photobook: Tools for content-based manipulation of image databases," in *Proc. SPIE Storage and Retrieval for Image and Video Datbases II*, Feb. 1994, vol. 1908, pp. 34–47.

[4] 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*, vol. 28, pp. 23–32, Sept. 1995.

[5] R. Jain, A. Pentland, and D. Petkovic, Eds., *NSF-ARPA Workshop on Visual Information Management Syst.*, Boston, MA, June 1995.

[6] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, F. Jain, and C. Shu, "The virage image search engine: An open frame work for image management," in *SPIE Storage and Retrieval for Image and Video Databases IV*, vol. SPIE 2670, Feb. 1996. pp. 76–87.

[7] D. Forsyth, J. Malik, M. Fleck, H. Greenspan, T. Leung, S. Belongie, C. Carson, and C. Bregler, "Finding pictures of objects in large collections of images," Tech. Rep. CSD-96-905, Dep. EECS, Univ. California, Berkeley, 1996.

[8] H. Chen, B. Schatz, T. Ng, J. Martinez, A. Kirchhoff, and C. Lin, "A parallel computing approach to creating engineering concept spaces for semantic retrieval: The Illinois digital library initiative project," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 771–782, Aug. 1996.

[9] J. R. Smith and S.-F. Chang, "Exploring image functionalities in www applications-development of image/video search and editing engines," in *Proc. ICIP*, Santa Babara, CA, vol. 3, Oct. 1997, pp. 1–4.

[10] M. Swain and D. Ballard, "Color indexing," *Int. J. Comput. Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[11] H. Zhang, C. L. Y. Gong, and S. Smolia, "Image retrieval based on color features: An evaluation study," in *SPIE Digital Image Storage and Archiving Syst.*, vol. SPIE 2606, Oct. 1995, pp. 212–220,

[12] J. Huang, S. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image indexing using color correlograms," in *IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, Puerto Rico, June 1997, pp. 744–749.

[13] F. Liu and R. Picard, "Periodicity, directionality and randomness: Wold features for image modeling and retrieval," M.I.T. Media Lab. and Modeling Group, Tech. Rep. 320, 1994.

[14] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 837–842, Aug. 1996.

[15] L. M. Kaplan, R. Murenzi, and K. R. Namuduri, "Fast texture database retrieval using extended fractal features," in *Storage and Retrieval for Image and Video Databases*, San Jose, CA, vol. 3312, Jan. 1998, pp. 162–175.

[16] R. Mehrotra and J. Gary, "Similar-shape retrieval in shape data management," *IEEE Computer*, vol. 28, pp. 57–62, Sept. 1995.

[17] Z. Lei, T. Tasdizen, and D. B. Cooper, "Object signature curve and invariant shape patches for geometric indexing into pictorial databases," in *Multimedia Storage and Archiving Syst.*, Dallas, TX, vol. 3229, Nov. 1997, pp. 232–243.

[18] M. Stricker and M. Orengo, "Similarity of color images," in *SPIE Storage and Retrieval for Image and Video Datbases III*, vol. SPIE 2185, Feb. 1995, pp. 381–392.

[19] M. Stricker, "Color indexing with weak spatial constraints," in *SPIE Storage and Retrieval for Image and Video Databases IV*, vol. 2670, Feb. 1996, pp. 29–40.

[20] X. Wan and C.-C. Kuo, "Color distribution analysis and quantization for image retrieval," in *SPIE Storage and Retrieval for Image and Video Databases IV*, vol. SPIE 2670, Feb. 1996, pp. 9–16.

[21] ——, "Image retrieval with multiresolution color space quantization," in *Electron. Imaging and Multimedia Syst.*, Nov. 1996.

[22] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient color histogram indexing for quadtratic form distance functions," *IEEE Trans. Pattern Machine Intell.*, vol. 17, pp. 729–736, July 1995.

[23] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley-Interscience, 1973.

[24] M. Gervautz and W. Purgtathofer, *A Simple Method for Color Quantization: Octree Quantization*. San Diego, CA: Academic, 1990.
[25] G. Wyszecki and W. Stiles, *Color Science*. New York: Wiley, 1982.
[26] J. R. Smith and S.-F. Chang, "Single color extraction and image query," in *Proc. ICIP*, 1995.
[27] Y. Rui and T. Huang, "Automated region segmentation using attraction-based grouping in spatial-color-texture space," in *Proc. ICIP*, Lausanne, Switzerland, Sept. 1996, vol. 1, pp. 53–56.
[28] W. Ma and B. Manjunath, "Edge flow: A framework of boundary detection and image segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, Puerto Rico, June 1997, pp. 744–749.

Xia Wan was born in Baotou, China, in 1969. She received the B.S. degree and M.S. degree in electrical engineering from Tsinghua University, Beijing, China in 1991 and 1994, respectively, and the Ph.D. degree in electrical engineering from University of Southern California in 1998.

From 1994 to 1998, she was a Research Assistant with the Signal and Image Processing Institute and Integrated Media Systems Center of USC. Since July 1998, she has been a Senior Software Engineer at SONY's U.S. Research Lab. Her research interests are in video compression and communications, and visual information management.

C.-C. Jay Kuo (S'83–M'86–SM'92) received the B.S. degree from the National Taiwan University, Taipei, in 1980 and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively, all in electrical engineering.

From October 1987 to December 1988, he was Computational and Applied Mathematics (CAM) Research Assistant Professor in the Department of Mathematics at the University of California, Los Angeles. Since January 1989, he has been with the Department of Electrical Engineering-Systems and the Signal and Image Processing Institute at the University of Southern California, where he currently has a joint appointment as Associate Professor of Electrical Engineering and Mathematics. His research interests are in the areas of digital signal and image processing, audio and video coding, wavelet theory and applications, multimedia technologies, and large-scale scientific computing. He has authored more than 300 technical publications in international conferences and journals.

Dr. Kuo is a member of SIAM, ACM, and a Fellow of SPIE. He is the Editor-in-Chief for the *Journal of Visual Communication and Image Representation*, and served as Associate Editor for *IEEE* TRANSACTIONS ON IMAGE PROCESSING in 1995–1998 and *IEEE* TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY in 1995–1997. He received the National Science Foundation Young Investigator Award (NYI) and Presidential Faculty Fellow (PFF) Award in 1992 and 1993, respectively.