Retrieval of translated, rotated and scaled color textures

Cheng-Hao Yao, Shu-Yuan Chen

Department of Computer Engineering and Science, Yuan-Ze University, 135 Yuan-Tung Rd., Nei-Li, Chung-Li, Taoyuan 32026, Taiwan, ROC

Received 18 October 2001; accepted 24 June 2002

Abstract

A new method for color texture retrieval using color and edge features is proposed in this study. The proposed method unifies color and edge features rather than simply analyzing only color characteristics. First, the distributions of color and local edge patterns are used to derive a similarity measure for a pair of textures. Then, a retrieval method based on the similarity measure is proposed to retrieve texture images from a database of color textures. Finally, the similarity measure is extended to retrieve texture regions from a database of natural images. Since the proposed feature distributions can resist variations in translation, rotation and scale, our method has the ability to retrieve texture images or regions that change in translation, rotation and/or scale. The effectiveness and practicability of the proposed method have been demonstrated by various experiments.

Keywords: Content-based retrieval; Color texture retrieval; Color texture segmentation; Similarity measure; Local edge pattern; Texture images; Texture region

1. Introduction

Texture is an important characteristic for the analysis of many types of images. Many papers have proposed methods for either texture classification [1–8] or retrieval [9,10], but a rather limited number of papers for color texture classification [11,12] or retrieval [13,14]. Moreover, these color texture methods combine color and texture in a limited method. For example, they may analyze only color characteristics [11,12]. In this study a new retrieval method of unifying color and edge features is proposed. More importantly, the adopted features can resist variations in translation, rotation and scale so that the texture images or texture regions that change in translation, rotation and/or scale can be retrieved.

Some rotation invariant texture features have been derived including polarograms [1], generalized co-occurrence matrices [2], circular symmetric autoregressive random filed [3], rotation-invariant simultaneous autoregressive model [4], Gabor filter [5], etc. Petikäinen et al. [6] proposed a distribution-based classification approach and a set of recently developed texture measures for rotation-invariant texture classification. However, only a rather limited number of papers simultaneously focus on translation, rotation and scale invariance [7,8]. Kohen et al. [7] used Gaussian Markov random filed models and You et al. [8] used a tuned mask to achieve rotation and scale invariance classification results.

Lin et al. [9] proposed a regular texture image retrieval method using texture primitives and their displacement vectors. Five features are computed from co-occurrence matrices of the texture primitive to characterize textures. Manjunath et al. [10] use texture information for browsing and retrieving from large image databases. They use Gabor wavelet features for texture analysis and retrieval. However, the two methods can be applied to only gray-level images. Moreover, the former manipulates regular textures only.

Some of the methods in the literature manipulate color textures, but their purposes are texture classification [11,12]...

Carson et al. [13] presented a new “blobworld” image representation that provides a transformation from the raw pixel data to a small set of localized coherent regions in color and texture region. The “blobworld” representation is, in turn, used to retrieve texture regions from images. Although this method is rotation and scale invariant, it is too complicated. Zhong et al. [14] addressed the problem of locating objects using color, texture and shape. To speed up processing, they extract color and texture features from the discrete cosine transform which is often used to do image compression. However, since the database images are not presegmented and annotated, the exhaustive search is time consuming.

In this paper we propose a new method for color texture retrieval using color and edge features in a unified way. The distributions of color and local edge patterns (LEPs) are used to derive a similarity measure for a pair of textures which is, in turn, used to retrieve texture images from a database of color textures. Furthermore, the texture similarity measure is extended to retrieve texture regions from a database of natural images. The proposed feature distributions can resist variations in translation, rotation and scale, thus, our method has the ability to retrieve texture images or regions that change in translation, rotation and/or scale.

Texture features and similarity measures are described in Section 2. These are used to retrieve textures from color texture database and then extended to natural images as shown by experimental results in Section 3. Finally, conclusions are given in Section 4.

2. Color texture retrieval based on feature distributions

2.1. Extraction of feature distributions

Two features are proposed in this study to distinguish between color textures, namely a color histogram and an LEP histogram. The former is related to the distribution of colors in a texture region, while the latter is related to the distribution of LEPs in a texture region. Both are extracted from a quantized image. In this study images are simply quantized by partitioning the RGB color space into 64 disjoint regions, each of which is represented by a composite of three bands encoded by two bits. Note that the proposed method is applied to RGB color space rather than a more meaningful color space such as HSV space since color space transformation is not required for this space. This in turn leads to fast response time for on-line query. In addition, the retrieval results based on RGB color space are satisfactory as shown in the experimental results in Section 3.

2.1.1. Color histogram

After quantization, each pixel has a color label from 0 to 63. Accordingly, we can count the frequency of the color label in a texture region \( R \), leading to a color histogram. More specifically, the color histogram \( h_c \) for a texture region \( R \) is obtained by using the following equation:

\[
h_c(i) = \frac{n_i}{N}, \quad i = 0, \ldots, 63,
\]

where \( n_i \) is the number of pixels with color label \( i \), and \( N \) is the number of total pixels in \( R \). It is obvious that the color histogram can resist variations in translation, rotation and scale.

2.1.2. Local edge pattern histogram

The idea of the local binary pattern (LBP) proposed in Refs. [6,15] was adapted to define LEP. The LBP describes the spatial structure of the local texture. The original \( 3 \times 3 \) neighborhood (Fig. 1(a)) is thresholded by the value of the center pixel. The thresholded pixels in a neighborhood (Fig. 1(b)) are multiplied by the corresponding binomial weights in a LBP mask (Fig. 1(c)). The resulting values (Fig. 1(d)) are summed to obtain the LBP value.

In a formal fashion, the LBP value is defined as [16]

\[
LBP(n,m) = \sum_{i,j \in I} k(i,j) \times u(f(n + i, m + j) - f(n,m)),
\]

where \( f(n,m) \) denotes the gray-level input image, \( I \) is a \( 3 \times 3 \) neighborhood, \( k(i,j) \) is the LBP mask, and \( LBP(n,m) \) is the output LBP value at the pixel located at \((n,m)\). Note that the LBP value ranges between 0 and 255.

To compute the LBP value, an edge image must be obtained first. The edge image is obtained by applying the Sobel edge detector to intensity gray level, denoted by \( Y \)-component. The equation to obtain the \( Y \)-component for each pixel can be referred to Ref. [17] and computed by

\[
Y = 0.299 \times R + 0.587 \times G + 0.114 \times B,
\]

where RGB correspond to the quantized color of the pixel. A binary edge image is obtained by setting all pixels with edge values greater than 150 to one and others to zero. From the edge image, the LEP feature can be extracted to describe the spatial structure of the local texture according to the organization of the edge pixels in a neighborhood.

For different goals, two types of LEP histograms are defined, one is LEPSEG for image segmentation, and the other is LEPINV for image retrieval. The former is sensitive to variations in rotation and scale, on the contrary, the latter...
is resistant to variations in rotation and scale. LEPSEG is first defined using an example of an edge image as shown in Fig. 2(a). The binary edge values in a $3 \times 3$ neighborhood are extracted and multiplied by the corresponding binomial weights in an LEP mask (Fig. 2(b)). The resulting values are summed to obtain the LEPSEG value. Note that the LEPSEG value ranges between 0 and 511. After each pixel has a LEPSEG value in a texture region $R$, we can count the frequency of the LEPSEG value, leading to a LEPSEG histogram (Fig. 2(c)).

In a formal fashion, the LEPSEG value is defined as

$$\text{LEPSEG}(n,m) = \sum_{i,j \in I} k_e(i,j) \times e(n,m),$$

where $e(n,m)$ denotes the binary edge image, $I$ is a $3 \times 3$ neighborhood, $k_e(i,j)$ is the LEP mask, and $\text{LEPSEG}(n,m)$ is the output LEPSEG value at the pixel located at $(n,m)$.

Note that the LBP and LEP masks are not exactly the same since LEP mask is specially designed so that an invariant LEP feature can be easily obtained as described later.

Accordingly, the LEPSEG histogram $h_e^{(01)}$ for a texture region $R$ is obtained using the following equation:

$$h_e^{(01)}(i) = \frac{n_i^{(01)}}{N}, \quad i = 0, \ldots, 511,$$

where $n_i^{(01)}$ is the number of pixels with LEPSEG value $i$ and $N$ is the number of total pixels in $R$. It is obvious that the LEPSEG histogram is resistant only to variation in translation.

The LEPSEG feature described above is sensitive to rotation, which is undesirable in retrieval applications. One possible solution [6] is described below. The LEPSEG value $\text{LEPSEG}(n,m)$ as specified by Eq. (4) can be expressed by a binary string $b_8b_7b_6b_5b_4b_3b_2b_1b_0$, since the binomial weights in the LEP mask are ascending in the clockwise order as shown in Fig. 2(b). That is the reason why the LEP mask is designed differently from the LBP mask. After the most significant bit corresponding to the central pixel is excluded, a number of binary shifts is then applied to
the 8-bit binary string “b7b6b5b4b3b2b1b0” until the value represented by the bit string is the least value [18]. After the processing, there are only 36 different least values derived from the 8-bit binary string [6]. Obviously, the 36 values are rotation invariant since only the sequence of the bit string is concerned rather than its starting point. For example, the LEPSEG value 30 in Fig. 2 is expressed by the bit string “000011110”. After the most significant bit corresponding to the central pixel is excluded from “0000011110”, the bit strings “000111100”, “001111000”, “111100000”, “111100001”, “110000011”, “100000111” and “000001111” have the same least value 15. However, the 36 values do not describe whether or not the central pixel is an edge pixel. Thus, if the central pixel is an edge pixel then 36 is added, leading to a LEPROT value. Note that the LEPROT feature is rotation invariant.

In order to resist variation in scale, the LEPROT must be modified in the following way. The LEPROT values are divided into two parts depending on whether or not the central pixel of the neighborhood is an edge pixel. For example, the central pixel of the neighborhood shown in Fig. 2(a) is not an edge pixel. In this way two LEPINV histograms, $he^{(0)}$ and $he^{(1)}$, can be obtained for a texture region $R$ using the following equation:

$$he^{(0)}_i = \frac{n_i}{N^{(0)}}, \quad he^{(1)}_i = \frac{n_i + 36}{N - N^{(0)}}, \quad i = 0, \ldots, 35,$$

(6)

where $n_i$ is the number of pixels with LEPROT value $i$ and $N^{(0)}$ is the number of total non-edge pixels in $R$.

The reason why the LEPINV histogram can resist variation in scale is given. In general, the number of non-edge pixels is proportional to the square of the scale factor, while the number of edge pixels is directly proportional to the scale factor. Similar conjecture can be derived for the numbers of pixels with a specified value LEPROT in the cases of edge and non-edge pixels. Since histograms $he^{(0)}$ and $he^{(1)}$ are individually related to non-edge and edge pixels, they are preserved even with a change in scale.

2.2. Similarity measure based on feature distributions

The histogram intersection technique [19] is used to measure the degree of matching between two texture regions on the basis of the distributions of color and LEP, namely similarity measure. However, the distance function in the feature distributions can also be used to achieve the same goal. The difference is only that the higher the similarity value, the more alike the two distributions, while the distance function is on the contrary. In Ref. [20], the effectiveness of the similarity measure is almost the same as that of the distance function. Thus, we randomly choose the similarity measure to measure the degree of matching between two histograms.

Similarity measure $Hc$ of two color texture regions $R^{(q)}$ and $R^{(l)}$ in the color histogram is computed by

$$Hc = \sum_{i=0}^{63} \min(hc^{(q)}_i, hc^{(l)}_i),$$

(7)

where $hc^{(q)}_i$ and $hc^{(l)}_i$ are the frequencies in bin $i$ of the two color histograms of texture regions $R^{(q)}$ and $R^{(l)}$. Note that the closer $Hc$ is to 1, the more alike the color histograms $hc^{(q)}$ and $hc^{(l)}$.

Similarity measures $He^{(01)}$, $He^{(0)}$, $He^{(1)}$ of two color texture regions $R^{(q)}$ and $R^{(l)}$ in the LEP histograms are computed by

$$He^{(01)} = \sum_{i=0}^{511} \min(hc^{(01)_i}_i, hc^{(01)_i}_i),$$

(8)

$$He^{(0)} = \sum_{i=0}^{35} \min(hc^{(0)_i}_i, hc^{(0)_i}_i),$$

(9)

$$He^{(1)} = \sum_{i=0}^{35} \min(hc^{(1)_i}_i, hc^{(1)_i}_i),$$

where $hc^{(01)}_i$, $hc^{(0)}_i$, $hc^{(1)}_i$ are the LEPSEG and LEPINV histograms as specified by Eqs. (5) and (6) in Section 2.1.2. Again, the closer $He^{(01)}$, $He^{(0)}$, $He^{(1)}$ are to 1, the more alike the LEP histograms $hc^{(01)}$, $hc^{(0)}$, $hc^{(1)}$.

Finally, texture similarity measures are computed as weighted sums of similarity values in color histogram ($Hc$) and similarity values in LEP and LEPINV histograms ($He^{(01)}$, $He^{(0)}$, $He^{(1)}$). Thus, there are two texture similarity values

$$H_{seg} = wc \times Hc + we \times He^{(01)}$$

(10)

$$H_{int} = wc \times Hc + we \times (we^{(0)} \times He^{(0)} + we^{(1)} \times He^{(1)}).$$

(11)

The weights $wc$, $we$, $we^{(0)}$ and $we^{(1)}$ were determined experimentally and set to 0.6, 0.4, 0.2 and 0.8, respectively, in Section 3.

Note that $H_{int}$ is resistant to variations in translation, rotation and scale. So, $H_{int}$ is used as texture similarity measure for retrieval hereafter. On the other hand, $H_{seg}$ is sensitive to both rotation and scale. Hence, $H_{seg}$ is used for region segmentation in Section 3.2 since the textures different in orientation and scale should be distinguished into different regions.

3. Experimental results

The proposed method has been implemented on an IBM compatible PC with a single Intel Pentium III 800 MHz CPU and 384 Megabytes SDRAM. The operating system is Microsoft Windows 2000 Server Chinese version Service Pack 2. The program was developed in the
C++ language and compiled under Borland C++ Builder version 4.0. The experimental results include two parts, one for color texture images, and one for natural images.

### 3.1. Color texture retrieval for texture images

The texture database used in the experiments consists of 200 different texture classes. The images of the 200 \( 512 \times 512 \) classes were collected from different sources including the Ulead PhotoImpact Chinese version 6.0 image database and web site Vistex texture database, (URL: http://www-white-media.mit.edu/vismod/imager/VisionTexture/vistex/html). Each of the \( 512 \times 512 \) images is divided into nine \( 128 \times 128 \) non-overlapping sub-images. The nine sub-images belonging to the same texture class are called related images and are shown in Fig. 3. There are different local distortions corrupting in the related images. Consequently, we obtain a database of \( 1800 (=9 \times 200) \) \( 128 \times 128 \) color texture images. Each texture image is then regarded as a whole region from which color and LEPINV histograms are extracted by using Eqs. (1) and (6), respectively, and only the histograms are stored into the database. Some texture images from the database are shown in Fig. 4.

Fig. 4. Sample images from texture database.

The experiments are designed to retrieve similar textures for the query texture image. Like database image, respective color and LEPINV histograms are also extracted from each query texture image. The similarity measure \( H_{inv} \) between the query texture image and each database image can then be computed by using Eq. (11). All the database images are then sorted by the respective similarity values. The rank denotes the position of the database image in the sorted list and the database image with the lowest rank is regarded as most similar to the query image. Since the related images are most alike to the query texture image, retrieval performance is evaluated using two measures based on the ranking of the related images. One is the ratio of AVRR to IAVRR (denoted by \( AVRR/IAVRR \) [22]), where \( AVRR \) is the average rank of all the related images, and \( IAVRR \) is the ideal average rank when all the related images are ranked at the top, i.e., at positions 1, 2, 3, \ldots and so on. They can be calculated, respectively, as follows:

\[
AVRR = \frac{\sum_{i=1}^{M} i \times d_i}{N_r}, \quad IAVRR = \frac{\sum_{i=1}^{N_r} i}{N_r},
\]

where \( M \) is the number of total images in the database, and \( N_r \) is the number of related images in the database for a query image. For the color texture database in our experiment, \( M \) and \( N_r \) are set to 1800 and 9, respectively. The value \( d_i = 1 \), if the \( i \)th retrieved image is related; \( d_i = 0 \), otherwise. Perfect performance would result in the minimum value \( AVRR/IAVRR \) of 1.

The other measure is in terms of precision \( P(T) \) and recall \( R(T) \). They can be evaluated as follows:

\[
P(T) = \frac{\sum_{i=1}^{T} d_i}{T}, \quad R(T) = \frac{\sum_{i=1}^{T} d_i}{N_r},
\]

where \( N_r \) is the number of related images in the database and \( T \) is the number of retrieved images. The higher the values \( P(T) \) and \( R(T) \), the better the performance; both have maximum values 1. From Eq. (13), it is easy to find that the point of equivalent precision and recall rates occur when \( T = N_r \). This point has been mentioned in the figure of precision/recall as shown in Figs. 7 and 22.

Figs. 5 and 6 show two results in our retrieval experiments. In these figures query texture images are selected from among the 1800 texture database. The query images in Figs. 5(a) and 6(a) were rotated and scaled from the original texture image. If the proposed method is effective in resisting variations in translation, rotation and scale, all the related images of the original texture must be retrieved. Obviously, the retrieval results meet the requirement as can be seen in Figs. 5(b) and 6(b). All the retrieved images are displayed in the ascending order of similarity values from
left to right and top to bottom. The precision/recall of our method is shown in Fig. 7. From Figs. 5–7, we conclude that the proposed texture similarity measure, $H_{inv}$, is indeed resistant to variations in translation, rotation and scale.

Two earlier methods are implemented in this study for comparison. One uses color moments to classify color textures [11], denoted by Color Moment, and the other uses LBP and VAR to classify gray-level textures [6], denoted by LBP + VAR. The comparisons include retrieval performance for the original textures, transformed textures and distorted textures.

A comparison of our method with the other two methods for the original and transformed textures is shown in Fig. 8 and Table 1, which indicate that our method is the most effective and the LBP + VAR is the worst. Since LBP + VAR is sensitive to variation in scale, we altered the scale factors and compare our method with the Color Moment method only. Clearly, our method still has the best performance as shown in Fig. 9.

On the other hand, the LBP + VAR method is designed for gray-level images, while our method is designed for color images. The comparison seems a little unfair. So we...
Our method
Color moment
LBP + VAR

Rotate 30°
Rotate 45°
Rotate 60°

Contrast
Enlarge
Original

Table 1
Comparison of our method with Color Moment and LBP + VAR in retrieval performance

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Enlarge</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>2.49/87.56%/96.94%/98.44%</td>
<td>3.19/81.22%/95.44%/97.44%</td>
<td>2.42/82.78%/96.39%/97.61%</td>
</tr>
<tr>
<td>Color Moment</td>
<td>7.53/66.22%/86.11%/92.00%</td>
<td>6.11/67.28%/87.44%/93.17%</td>
<td>6.41/56.72%/84.28%/92.56%</td>
</tr>
<tr>
<td>LBP + VAR</td>
<td>7.97/78.89%/89.78%/92.89%</td>
<td>126.82/3.94%/12.06%/18.22%</td>
<td>30.74/33.33%/61.72%/70.67%</td>
</tr>
</tbody>
</table>

Rotate 30°
Rotate 45°
Rotate 60°

Our method
Color Moment
LBP + VAR

Enlarge
Contract

Table 1
Comparison of our method with Color Moment and LBP + VAR in retrieval performance

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Enlarge</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>2.70/84.39%/96.61%/98.22%</td>
<td>2.74/84.17%/96.67%/98.39%</td>
<td>2.89/83.39%/96.06%/98.06%</td>
</tr>
<tr>
<td>Color Moment</td>
<td>7.83/61.83%/85.06%/91.61%</td>
<td>7.94/60.94%/84.39%/91.50%</td>
<td>7.49/63.06%/86.11%/91.56%</td>
</tr>
<tr>
<td>LBP + VAR</td>
<td>20.32/59.11%/75.72%/82.11%</td>
<td>16.95/57.83%/78.78%/85.39%</td>
<td>20.62/58.94%/76.11%/82.44%</td>
</tr>
</tbody>
</table>

*Each entry (w/x/y/z) denotes \( AVRR/IAVRR, R^{(10)}, R^{(50)}, R^{(100)} \).
perform another kind of comparison. For each color texture image, the \( Y \)-components are quantized into 64 gray levels. The proposed retrieval method is then applied to the gray-level texture images by replacing the color histogram by the histogram of the gray-level values. Comparisons of our method with the LBP + VAR method in gray-level version are shown in Fig. 10. Obviously, our method is still better than the LBP + VAR method.

We also conducted experiments to compare different methods in the degree of resistance to distortion. The types of distortions include Gaussian blurring and Gaussian noises. We use Photoshop Chinese Version 6.0.1 to blur images and add noise. Some distorted textures are shown in Fig. 11. Comparisons of the results of our method with the other two methods for these distorted textures are shown in Figs. 12 and 13. Since the LBP + VAR method is sensitive to the distortions, we compare our method with the Color Moment method only in the cases of more serious distortions. The results are shown in Figs. 14 and 15. For most serious distortions, the degree of resistance of our method is higher, except for some serious Gaussian noise.

As mentioned in Section 2.2, there are four weights \( w_c \), \( w_e \), \( we^{(0)} \) and \( we^{(1)} \) involved in \( H_{\text{inv}} \) as specified by Eq. (11). An experiment is conducted to show that the values 0.6, 0.4, 0.2 and 0.8 for \( w_c \), \( w_e \), \( we^{(0)} \) and \( we^{(1)} \), respectively, are appropriate. The weights \( w_c \) and \( w_e \) are set to seven pairs of values as shown in Fig. 16 and Table 2. Note that the pair (1.0, 0.0) is excluded in Fig. 16 so that the difference between other pairs can be signified since this pair has too large value of \( AVRR/IAVRR \) as listed in Table 2. From the experimental results, we find that \( H_{\text{inv}} \) is insensitive to weights provided that \( w_c \) is not less than \( w_e \). Under such condition, it is preferred that the difference between the weights \( w_c \) and \( w_e \) is as small as possible so that the significane of the color histogram is almost the same as that of the LEPINV histogram. Since the pair (0.6, 0.4) is most close to (0.5, 0.5) and leads to small value of \( AVRR/IAVRR \), we set the weights of \( w_c \) and \( w_e \) to 0.6 and 0.4, respectively. As regards the weights \( we^{(0)} \) and \( we^{(1)} \), four pairs are tested as shown in

Fig. 10. Comparison of our method with LBP + VAR in \( AVRR/IAVRR \) for gray-level images.

Fig. 17. According to the same determination rule for the weights \( wc \) and \( we \), we set \( we^{(0)} \) and \( we^{(1)} \) to 0.2 and 0.8, respectively.

3.2. Color texture retrieval for natural images

The proposed color texture retrieval algorithm for natural images consists of two stages: database creation and query matching. The former segments each database image into several homogeneous texture regions. The latter matches the queried texture region to each texture region of a database image so that those database images containing the queried texture region can be retrieved.

The segmentation method proposed in Refs. [15,21] is adopted. The method consists of two parts: hierarchical splitting and agglomerative merging. First, hierarchical splitting is used to divide the image into regions of roughly uniform texture. An agglomerative procedure then merges similar adjacent regions until a stopping criterion is met. This results in different homogeneous texture regions present in the image.

The hierarchical splitting algorithm recursively splits the original image into four rectangular blocks of varying size. The decision whether or not to split a block is based on a similarity test. A block is first divided into four non-overlapping subblocks. Color histogram, \( hc \), and LEPSEG histogram, \( he^{(01)} \), as specified by Eqs. (1) and (5), respectively, are computed for each subblock. The similarity measure, \( H_{\text{seg}} \), as specified by Eq. (10), is then used to measure the six pairwise similarities of the four subblocks. The largest and smallest among the six \( H_{\text{seg}} \) values are denoted by \( H_{\text{seg}}^{\text{max}} \) and \( H_{\text{seg}}^{\text{min}} \), respectively. The block is declared to be inhomogeneous, and is thus split into four subblocks when the relative dissimilarity within the block is greater than a threshold, i.e.,

\[
H_{\text{seg}}^{\text{max}} > X \times H_{\text{seg}}^{\text{min}},
\]

where \( X \) is empirically set to 1.02 [21]. In general, the smaller the value of \( X \) is, the more split is performed. However, when the term \( H_{\text{seg}}^{\text{min}} \) of Eq. (14) is zero, split is compulsive no matter what value of \( X \) is. The term \( H_{\text{seg}}^{\text{min}} \) of
Eq. (14) will be zero only when all of the $3 \times 3$ neighborhoods in two subblocks are different. This situation indicates that two of the four subblocks are absolutely different. Thus, splitting the block into four subblocks is imperative.

Once the image has been split into blocks of roughly homogeneous texture, we apply an agglomerative procedure to merge similar adjacent regions until one of the two stopping criteria is satisfied. At each stage we merge the pair of adjacent regions which have the largest merger importance (MI) value, where $MI$ computed from

$$MI = \frac{1}{\sqrt{p}} \times H_{seg},$$

where $p$ is the number of pixels in the smaller of the two regions, and $H_{seg}$ is the similarity of a pair of regions as calculated by Eq. (10). Note that the higher the $MI$ value, the more the pair of adjacent regions is preferred to be merged. In other words, it is preferred to merge a pair of adjacent
regions when they are alike and one of them has large size. Once the pair of adjacent regions with the largest $MI$ value has been found, they are merged, and their color and LEP histograms, $hc$ and $he^{(01)}$, are summed to form the histograms of the newly merged region. Merging proceeds until either of the two stopping rules is true

$$MI_{cur} < Y \times MI_{\min}, \quad MI_{cur} < Z \times MI_{\max}.$$  \hspace{1cm} (16)

Here, $MI_{\min}$ is the smallest of all the proceeding merger importance values, $MI_{\max}$ is the largest, and $MI_{cur}$ is the merger importance for the current best merge. The values of thresholds $Y$ and $Z$ were determined experimentally to be 0.675 and 0.25, respectively [21]. An example of segmentation is shown in Fig. 18.

Users can choose any texture region from the query image for query matching. For example, the texture region may be a leopard skin pattern or that from a tiger, zebra, bird, etc. The query texture region is then matched to all texture regions segmented from each database image. The similarity between the query and each region of database image is
Table 2

Retrieval performance of using different weights \( wc \) and \( we \)

<table>
<thead>
<tr>
<th>( wc )</th>
<th>( we )</th>
<th>Original</th>
<th>Enlarge</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>3.02/82.28%/95.67%/97.61%</td>
<td>3.12/81.61%/94.94%/97.06%</td>
<td>2.28/79.83%/96.28%/98.50%</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>2.81/84.06%/96.06%/97.94%</td>
<td>3.01/82.78%/95.00%/97.44%</td>
<td>2.14/81.78%/96.61%/98.50%</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>2.64/85.28%/96.50%/98.28%</td>
<td>2.96/82.45%/95.39%/97.61%</td>
<td>2.13/82.61%/96.78%/98.50%</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3</td>
<td>2.53/86.50%/96.78%/98.33%</td>
<td>3.00/82.00%/95.56%/97.50%</td>
<td>2.21/83.00%/96.72%/97.72%</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>2.49/87.56%/96.94%/98.44%</td>
<td>3.19/81.22%/95.44%/97.44%</td>
<td>2.42/82.78%/96.39%/97.61%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>2.54/88.22%/97.28%/98.33%</td>
<td>3.68/78.78%/95.33%/97.28%</td>
<td>2.90/81.50%/95.06%/97.50%</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>9.62/69.33%/86.83%/92.33%</td>
<td>83.85/3.11%/16.00%/28.39%</td>
<td>42.98/18.61%/42.44%/56.89%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( wc )</th>
<th>( we )</th>
<th>Rotate 30°</th>
<th>Rotate 45°</th>
<th>Rotate 60°</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>3.00/81.56%/95.61%/97.83%</td>
<td>3.11/81.28%/95.28%/97.78%</td>
<td>3.27/81.17%/94.78%/97.00%</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>2.85/82.78%/96.11%/97.94%</td>
<td>3.07/82.06%/95.94%/98.00%</td>
<td>2.94/82.61%/95.72%/97.17%</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>2.73/83.72%/96.56%/98.06%</td>
<td>2.80/83.28%/96.17%/98.17%</td>
<td>2.92/83.22%/96.00%/97.78%</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3</td>
<td>2.66/84.39%/96.67%/98.17%</td>
<td>2.72/84.17%/96.44%/98.33%</td>
<td>2.84/83.50%/96.06%/98.11%</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>2.70/84.39%/96.61%/98.22%</td>
<td>2.74/84.17%/96.67%/98.39%</td>
<td>2.89/83.39%/96.06%/98.06%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>2.99/82.78%/96.06%/97.78%</td>
<td>2.88/83.28%/96.56%/98.39%</td>
<td>3.16/82.28%/96.00%/97.72%</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>46.83/38.72%/61.39%/68.00%</td>
<td>39.64/36.56%/59.78%/67.89%</td>
<td>47.13/38.39%/60.72%/67.50%</td>
</tr>
</tbody>
</table>

Each entry \((w/x/y/z)\) denotes \((AVRR/IAVRR,\ R^{(10)},\ R^{(50)},\ R^{(100)})\).

Fig. 16. Comparison of \( AVRR/IAVRR \) with respect to different weights of \( wc \) and \( we \).

Fig. 17. Comparison of \( AVRR/IAVRR \) with respect to different weights of \( we^{(0)} \) and \( we^{(1)} \).
calculated. In other words, if the database image is segmented into \( n \) regions, \( n \) similarity values are obtained. The largest value among these \( n \) similarity values represents the similarity between the query and the database image. All the database images are then sorted by their highest similarity values. The rank denotes the position of the database image in the sorted list and the database image with the lowest rank is regarded as most similar to the query image.

An example to illustrate the query matching strategy is shown in Fig. 19. The left column in Fig. 19(a) is the query image with the query texture region as the background texture region. The right column of Fig. 19(a) has three database images each of which has been segmented into homogeneous regions. For example, Images 1, 2 and 3 have 4, 2 and 3 regions, respectively. Thus, the query texture region is compared to each region of Images 1, 2 and 3, respectively, resulting in 4, 2 and 3 similarity values. The largest similarity values of Images 1, 2 and 3 are \( H^1_4 \), \( H^2_2 \) and \( H^3_3 \), respectively. Assume that \( H^2_2 \) \( > \) \( H^3_3 \) \( > \) \( H^1_4 \) as shown in Fig. 19(b), thus the ranks of Images 1, 2 and 3 are 3,
Fig. 20. Retrieval results for an original zebra image: (a) Query image and (b) retrieval results.

1 and 2, respectively. In other words, Images 1, 2 and 3 are sorted in the order of Images 2, 3 and 1 in the retrieval list.

The proposed query matching incorporates two similarity measures, one related to region texture attribute, and the other to a region position attribute. The former is actually a special case of matching color textures by replacing an image by a region. Thus, the texture similarity of each pair of texture regions can be computed by $H_{\text{inv}}$, as specified by Eq. (11). On the other hand, interesting objects are usually located at the center of images. Thus, the similarity measure based on the region position attribute is computed by

$$H_{\text{pos}} = 1 - \frac{|x_0 - x_i|}{\text{Width}} - \frac{|y_0 - y_i|}{\text{Height}},$$

(17)

where $(x_0, y_0)$ is the center of the object, $(x_i, y_i)$ is the center of the image, $\text{Width}$ is the width of the image, and $\text{Height}$ is its height.

The proposed method uses the weighted sum of the two similarity measures, $H_{\text{inv}}$ and $H_{\text{pos}}$, to get final similarity value by

$$H = w_{\text{inv}} \times H_{\text{inv}} + w_{\text{pos}} \times H_{\text{pos}},$$

(18)
where $w_{inc}$ and $w_{pos}$ are determined experimentally as 0.7 and 0.3 in this section.

The proposed color texture retrieval algorithm is then applied to an image database containing 650 color images of animals, birds, flowers, outdoor and indoor scenes, etc. They were collected from different sources including the IMAGEMORE image database (URL: http://www.imagemore.com), web site Electronic Zoo/Net Vet-Animal Image...
Fig. 22. Precision/recall of our method for color natural images. The point of equivalent precision and recall is denoted by \((x, x)\). For zebra and leopard images, \(x = 0.37, 0.33, 0.38, 0.34, 0.39\) and \(0.47, 0.47, 0.47, 0.50, 0.49\) in the cases of original, enlarge, contract, rotate \(45^\circ\) and \(90^\circ\), respectively: (a) For zebra images and (b) for leopard images.

Fig. 23. Comparisons of our method with Color Histogram in \(AVRR/IAVRR\): (a) For zebra images and (b) for leopard images.
Table 3
Comparison of our method with Color Histogram in retrieval performancea

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Enlarge</th>
<th>Contract</th>
<th>Rotate 45°</th>
<th>Rotate 90°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zebra image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our method</td>
<td>7.59/32.67%/58.00%/76.67%</td>
<td>10.72/26.67%/49.33%/65.33%</td>
<td>6.80/31.33%/63.33%/78.67%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Color histogram</td>
<td>13.45/26.00%/48.00%/58.67%</td>
<td>15.29/23.33%/43.33%/56.00%</td>
<td>13.39/22.00%/43.33%/58.67%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leopard image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our method</td>
<td>4.84/23.67%/62.33%/78.00%</td>
<td>5.28/22.00%/61.00%/76.33%</td>
<td>4.76/23.67%/60.33%/78.67%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Color histogram</td>
<td>5.91/21.33%/56.67%/74.00%</td>
<td>5.71/21.00%/57.00%/76.67%</td>
<td>6.03/23.67%/56.67%/75.00%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Zebra texture
- Our method: 7.88/32.00%/60.67%/76.67%
- Color histogram: 14.87/23.33%/39.33%/53.33%

Leopard texture
- Our method: 4.75/23.33%/63.67%/81.00%
- Color histogram: 5.78/23.33%/58.00%/73.33%

aEach entry (w/x/y/z) denotes (AVRR/IAVRR, $P^{(10)}$, $P^{(50)}$, $P^{(100)}$).

![Figure 24](http://example.com/fig24.png)

Fig. 24. Comparison of $AVRR/IAVRR$ with respect to different weights of $w_{inv}$ and $w_{pos}$: (a) For zebra images and (b) for leopard images.

Collection (URL: http://netvet.wustl.edu/pix.htm), and the COREL photo database. PhotoShop Chinese version 6.0.1 was used to reduce the image size. The reduction criteria are, first, the width to height aspect ratio for each image must be retained, and second, the maximum value of width or height must be fixed at 512 pixels. Figs. 20 and 21 show two results from our retrieval experiments. In these figures, query images are selected from the natural image database. The query images in Figs. 20(a) and 21(a) are an original zebra and a rotated leopard image. The results are shown in Figs. 20(b) and 21(b), respectively.

We conducted more experiments with the original and transformed images to demonstrate the effectiveness of the proposed method. The transformed images include enlarged, contracted and rotated images. The natural images are enlarged by a factor of 2, contracted by a factor of 1/2, and rotated by angles of 45° and 90°. Retrieval performance is evaluated using $AVRR/IAVVR$ and precision/recall, specified by Eqs. (12) and (13), except that the number of related natural images, $N_r$, is 15 for zebra and 30 for leopard and the number of database images, $M$, is 650 for zebra and leopard. The precision/recall of the proposed method with respect to original and transformed images are shown in Fig. 22.

One earlier method [19] is implemented in this study for comparison, denoted by the Color Histogram method. The comparisons of our method with the Color Histogram method are shown in Fig. 23 and Table 3, which indicate that our method is more effective than the Color Histogram method.

As mentioned above, there are two weights $w_{inv}$ and $w_{pos}$ involved in $H$, specified by Eq. (18). Experiments are conducted by querying original and transformed images to show that the values of 0.7 and 0.3 for $w_{inv}$ and $w_{pos}$ are suitable. The weights of $w_{inv}$ and $w_{pos}$ are set to five pairs of values...
as shown in Fig. 24. From the experimental results, we find that the (0.7, 0.03) pair has the smallest value of $AVRR/IAVRR$ for Zebra, but the (0.8, 0.2) pair, for leopard. When both zebra and leopard images are concerned, since the value of $AVRR/IAVRR$ zebra is higher, the decision should be pro to zebra. So we decide to set $w_{obs}$ and $w_{pos}$ to 0.7 and 0.3, respectively.

4. Conclusions

In this study, a new color texture retrieval method using a combination of color and LEP histograms is proposed. We have developed two kinds of retrieval methods, color texture image retrieval and color texture region retrieval. From the results presented in Section 3, we conclude that our method has promise in the following situations:

(1) The proposed method based on color and LEP histograms is suitable for color texture retrieval.
(2) The proposed method can resist variations in rotation, translation, and scale.
(3) The proposed method is simple, yet effective.

Future work can be directed to the following topics:

(1) Threshold value must be adjusted to get better segmentation results.
(2) The color texture region retrieval method can be expanded to multiple region selection so that we can choose multiple texture regions for retrieval.

References