Name of workshop: Workshop on Multimedia Technology

Content-based Image Retrieval Using Moment-preserving Edge Detection

Shyi-Chyi Cheng* and Tian-Luu Wu

Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, 1 University Road, Yenchao, Kaohsiung 824, Taiwan,

Tel: +886-7-6011000 Ext. 2025, Fax: +886-7-6011012

e-mail: csc@ccms.nkfust.edu.tw

ABSTRACT

A content-based image retrieval algorithm based on a new edge detection technique is proposed. Both the query and database images are divided into non-overlapping square blocks and coded by the mean in each uniform block and by edge information in each non-uniform block. The coded blocks of a query image are then used to find matches from an image database. The edge feature in a given block is detected by applying the moment-preserving principle to the image data. The edge directions are approximated by multiples of 45º to speed up the matching process without introducing obvious distortion. For a larger database, a selective filtering strategy based on the visual-pattern histograms is also described to further speed up the retrieval process. The solution to the edge detection problem in a given block is also analytic. This algorithm can be performed very fast for large-database applications with no need for special hardware.

Keywords: Content-based image retrieval, Edge feature, Moment-preserving technique.

* To whom all correspondences should be addressed.

1. INTRODUCTION

With the rapidly increasing use of the Internet, the demands for storing multimedia information (such as text, image, audio, and video) have increased. Along with such databases comes the need for a richer set of search facilities that include keywords, sounds, examples, shape, color, texture, spatial structure and motion. Traditionally, textual features such as filenames, captions, and keywords have been used to annotate and retrieve images. As they are applied to a large database, the use of keywords becomes not only cumbersome but also inadequate to represent the image content. Many content-based image retrieval systems have been proposed in the literature [1-4]. To access images based on their content, low-level features such as colors [5-8], textures [9,10], and shapes of objects [11,12] are widely used as indexing features for image retrieval. Although content-based image retrieval is extremely desirable in many applications, it must overcome several challenges including image segmentation, extracting features from the images that capture the perceptual and semantic meanings, and matching the images in a database with a query image based on the extracted features. Due to these difficulties, an isolated image content-based retrieval method can neither achieve very good results, nor will it replace the traditional text-based retrievals in the near future.

Edge features, which are recognized as an important aspect of human visual perception, are commonly used in shape analysis. Decomposition of images into low-frequency blocks and blocks containing visually important features (such as edges or lines) suggest visual continuity and visual discontinuity constraints. A block is visually continuous if the values of all the pixels in the block are almost the same. In contrast, if the variations of the pixel values in the block are noticeable, it is a visually discontinuous block. The mean of a visually continuous block is enough to represent the block. If a block is visually discontinuous and if a strong orientation is associated with it, then it should be coded as a kind of visually important feature. Using coded edges, we can represent the structure of an image without explicitly extracting visual features.

In this paper, a new image retrieval algorithm is proposed based on the application of the moment-preserving technique to detect a visually important feature, namely an edge in a given image block. Each given image is divided into non-overlapping square blocks and coded block by block.

Edge features used to code an ordinary image, produce excellent image quality according to human perception and provide a promising approach for the representation of the image content with a compact code [13]. Furthermore, simple and analytical formulae to compute the parameters of the edge feature for both gray and color images are also derived in this study, which makes the computation very fast. The potential of using edge information for image retrieval was observed by Jain and Vailaya [12]. They proposed a retrieval technique by matching a query image with each image in a database on the basis of the edge direction and color histograms, and the similarity between two images can be assessed by combining the color- and shape-based similarity.

In contrast to the integrated color- and shape-based method mentioned above, in this paper we propose the usage of block-based visual patterns of an image as the features for identification. In this system, thirty-seven types of visual pattern including a uniform pattern and 36 edge patterns are defined to classify 4 x 4 image blocks of a given image. A simple approach for image retrieval on the basis of visual patterns is to encode both a database image and a query image and then compute the difference between the visual-pattern codes of the two images. In certain classes of images, the image details are very rich and do not lend themselves well to being indexed by simple visual features. Such images require a detailed search of the image content to be made as part of the query. For speed and efficiency, we use the visual-pattern histogram of a query image to extract a pool of candidates from a large database, and then to match the visual patterns block by block on the candidates.

The remainder of this paper is organized as follows. Section 2 presents the use of the moment-preserving principle to detect edges. Section 3 presents the similarity measurement with edge information. The proposed image matching strategy is shown in Section 4 followed by some experimental tests to illustrate the effectiveness of the proposed image retrieval method in Section 5. Finally, conclusions are drawn in Section 6.

2. PROPOSED MOMENT-PRESERVING EDGE DETECTION

A given image is partitioned into a set of non-overlapping square blocks. Each block is coded as either a uniform block or an edge block. The edge in each block is detected by the proposed edge detection technique, and the image can be reconstructed according to the parameters of these blocks.

2.1 *Review of moment-preserving principle*

Gray-level moment-based operators have been successfully developed for image processing [14-16]. Delp and Mitchell [14] first proposed block truncation coding (BTC), which used a two-level moment-preserving quantizer to compress monochrome image. Tabatabai and Mitchell [15] also proposed an operator, which can compute edge location by fitting first three gray-level moments to the input data. Compared with traditional edge operators such as those proposed by Robert and Sobel [17], the precision to subpixel accuracy has been reported. Applying this technique to image segmentation, Tsai [16] used the moment-preserving principle to select thresholds of input gray-level image. The threshold values were determined in such a way that the gray-level moments of an input image are preserved. In contrast to Tsai [16], a new moment-based operator, based on a thresholding technique called binary quaternion-moment-preserving (BQMP) thresholding was reported by Pei and Cheng [18]. The BQMP thresholding generalizes conventional gray-level moment-based operators [14-16] to be multi-dimensional by expressing the input color space as a quaternion-level space. An analytic solution for the BQMP thresholding is also obtained by the use of quaternion arithmetic. Pei and Cheng [18] also applied the BQMP thresholding technique to detect color edges with the precision to subpixel accuracy. There are no analytical formulae for the edge detection algorithms proposed in Tabatabai [15] and Pei [18], so their methods are not unified approaches for detecting edges of gray and color images, respectively.

2.2 *Proposed edge detector*

Figure 1 shows the aspects used for edge detection in this approach, including a continuous two-dimensional edge model specified by four parameters, two representative gray (color) values h_1 and h_2, an edge translation l, and an orientation angle θ for an edge in square block B. The edge is simply a step edge transition from representative value h_1 to representative value h_2. The edge translation l is defined as the length from the center of the edge model to the transition, and is confined within the range of -1 to +1. The parameter θ specified the direction of the edge and is confined within the range of 0 to 360 degrees. The value of θ can be computed individually first, as described next.

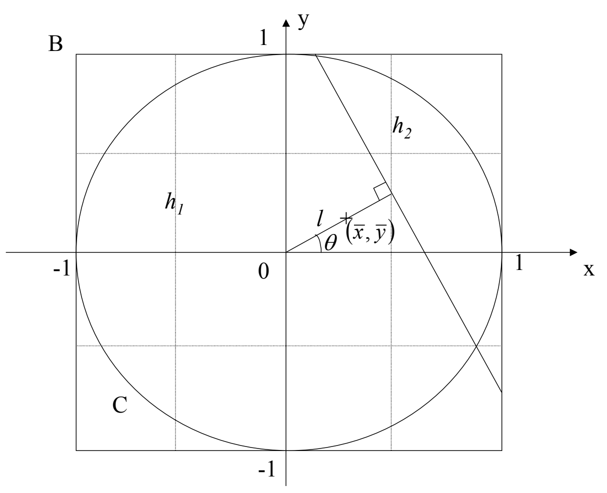


Fig. 1. An edge model in a 4 x 4 block *B*. The circle *C* is inscribed in *B*, and $(\bar{x},\bar{y})$ are the coordinates of the centers of gravity of the gray (color) values inside *C*.

The x-mass moment M_x and y-mass moment M_y defined within the cycle *C* inscribed *B* are as follows:

$$M_x = \iint_C x f(x,y)\,dydx, \tag{1}$$

$$M_y = \iint_C y f(x,y)\,dydx \tag{2}$$

where *f*(*x,y*) is the gray level and the vector norm at (*x,y*) for a gray-level image and a color image, respectively. The vector norm of the pixel at (*x,y*) of a color image is computed as

$$f(x,y) = \sqrt{R(x,y)^2 + G(x,y)^2 + B(x,y)^2} \tag{3}$$

where (*R*(*x,y*), *G*(*x,y*), *G*(*x,y*)) denotes the color value (*R, G, B*) of the pixel at (*x,y*) for a color image. The value of θ can be easily obtained from the values of M_x and M_y. Let $(\bar{x},\bar{y})$ be the coordinates of the center of gravity of the gray (color) values inside *C*. Then

$$(\bar{x},\bar{y}) = (\frac{M_x}{M_0}, \frac{M_y}{M_0}) \tag{4}$$

where M_0 is the mean of *C* and is computed by

$$M_0 = \iint_C f(x,y)\,dydx. \tag{5}$$

It has been found that the direction of the edge is perpendicular to the direction of the vector from the origin to $(\bar{x}, \bar{y})$ [19]. So,

$$\sin\theta = \frac{\bar{y}}{\sqrt{\bar{x}^2 + \bar{y}^2}} \tag{6a}$$

$$\cos\theta = \frac{\bar{x}}{\sqrt{\bar{x}^2 + \bar{y}^2}} \tag{6b}$$

and angle θ can be calculated by

$$\theta = \tan^{-1}\left(\frac{\bar{y}}{\bar{x}}\right) = \tan^{-1}\left(\frac{M_y}{M_x}\right). \tag{7}$$

The moments M_0, M_x and M_y are also useful to judge whether C and therefore B are visually uniform. Notice that if C is uniform, $(\bar{x}, \bar{y})$ will be very close to the origin of C. So, if the distance between the center of gravity $(\bar{x}, \bar{y})$ and the origin of C is less than a threshold τ, then it is considered that there is no edge involved in C, and B is determined to be a uniform block. This criterion for judging the uniformity of a given block is actually equivalent to the following:

$$\sqrt{M_x^2 + M_y^2} < \tau M_0. \tag{8}$$

To compute each of the moments M_0, M_x and M_y, a more convenient way is to correlate the pixel values in the block with a mask. Each mask represents the value resulting from performing the integration (1), (2), or (5) over each pixel in C, assuming *f(x,y)* is constant over that pixel. The circular limits are also included in the integration. The resulting set of masks is shown in Fig. 2.

$$\begin{array}{c|cccc} & 0.0788 & 0.2952 & 0.2952 & 0.0788 \\ \mathrm{M}_0 & 0.2952 & 0.2500 & 0.2500 & 0.2952 \\ & 0.2952 & 0.2500 & 0.2500 & 0.2952 \\ & 0.0788 & 0.2952 & 0.2952 & 0.0788 \end{array}$$

$$\begin{array}{c|cccc} & -0.0426 & -0.0622 & 0.0622 & 0.0426 \\ \mathrm{M}_\mathrm{x} & -0.0842 & -0.0625 & 0.0625 & 0.0842 \\ & -0.0842 & -0.0625 & 0.0625 & 0.0842 \\ & -0.0426 & -0.0622 & 0.0622 & 0.0426 \end{array}$$

$$\begin{array}{c|cccc} & 0.0426 & 0.0842 & 0.0842 & 0.0426 \\ & 0.0622 & 0.0625 & 0.0625 & 0.0622 \\ \mathrm{M}_\mathrm{y} & -0.0622 & -0.0625 & -0.0625 & -0.0622 \\ & -0.0426 & -0.0842 & -0.0842 & -0.0426 \end{array}$$

Fig. 2. Set of three masks for computing moments of a 4 x 4 block.

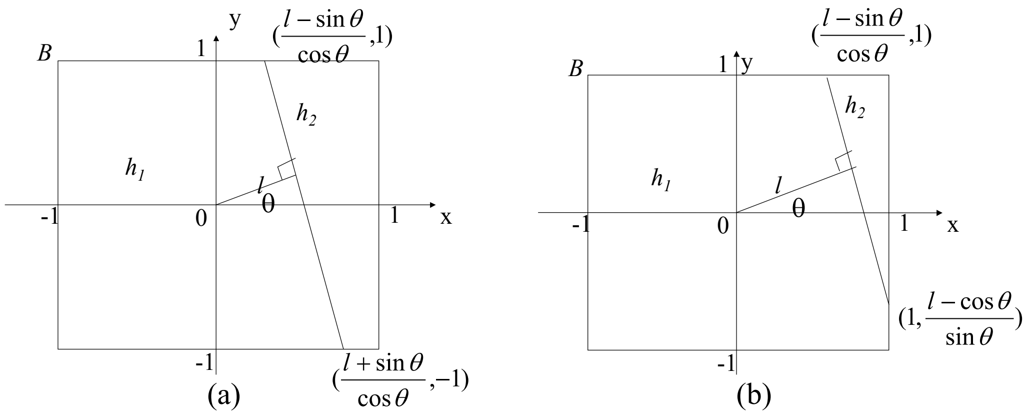
Once the value of θ is obtained, the solutions for h_1, h_2, and l can be found by applying the moment-preserving technique. Given the orientation of an edge of an input block, the edge might intersect the block at either opposite or neighboring sides, according to the variation of the pixel values within the block as shown in Fig. 3. The values of h_1, h_2, and l and the type of intersection can be decided by preserving the first three moments of the input block in the following manner:

$$m_k = p_1 h_1^k + p_2 h_2^k, \quad k=1, 2, 3 \tag{9}$$

where $p_1 = A_1/A$, A_1 is the area of B covered by pixel value h_1, A is the total area of B and is equal to 4, $p_2 = 1 - p_1$, $B = \{(x,y): |x| \le 1, |y| \le 1\}$, and m_k is the *kth* gray (quaternion) moment of the original block B and is computed as follows:

$$m_k = \frac{1}{A'} \iint_B g^k(x,y)\,dydx. \tag{10}$$

The value of A', which is the area of B, is also 4 and $g(x,y)$ is the gray value of (x,y) for gray-level images and the quaternion number of (x,y) for color images. For the methods to model a color vector as a quaternion number and compute the *kth* quaternion moment of a block please refer to [18].



B
y
1
$(\frac{l-\sin\theta}{\cos\theta}, 1)$
h_2
h_1
l
θ
-1
0
1
x
-1
$(\frac{l+\sin\theta}{\cos\theta}, -1)$
(a)
B
1
y
$(\frac{l-\sin\theta}{\cos\theta}, 1)$
h_2
h_1
l
θ
-1
0
1
x
$(1, \frac{l-\cos\theta}{\sin\theta})$
-1
(b)

Fig. 3. Given the orientation θ of an edge of a block, the edge might intersect the sides of the block in the following two ways: (a) the intersection points are located on opposite sides; (b) the intersection points are located on neighboring sides.

By assuming that $g(x,y)$ takes a constant value over each grid, the integral in (10) becomes a weighted sum of the pixel values in block B and can be written as:

$$m_k = \sum_x \sum_y w_{xy} g^k(x,y), \quad k= 1, 2, 3 \tag{11}$$

where w_{xy} is the weighting coefficient associated with (x,y) (if the size of block is 4 x 4, then w_{xy} = 1/16). Applying the moment-preserving [16] and quaternion-moment-preserving [18] thresholding techniques to a gray-level block and a color block, respectively, the unknown parameters h_1, h_2, p_1 and p_2 in equation (9) can be derived in closed form.

It is simple to calculate the remaining unknown parameter l from the value of p_2 in each case of the edge-block intersection patterns. First,

$$p_2 = \frac{A_2}{A} \tag{12}$$

where A_2 is the area covered by the pixel value h_2 in B and $A = 4$ is the area of B. Next, the value of A_2 can be computed by

$$A_2 = 2(1 - l/\cos\theta) \tag{13}$$

for the case as seen from Fig. 3(a), and

$$A_2 = (l - \cos\theta - \sin\theta)^2 / 2\cos\theta\sin\theta \tag{14}$$

for the case as seen from Fig. 3(b). From equations (12), (13) and (14), we get

$$l = (1 - 2p_2)\cos\theta \tag{15}$$

for the former case, and

$$l = \cos\theta + \sin\theta \pm \sqrt{8p_2\cos\theta\sin\theta} \tag{16}$$

for the latter case. Note that the sign of the third term of equation (16) should be negative because the value of θ is within the interval $(0, \pi/2)$ and the value of l is between 0 and $\sqrt{2}$ in Fig.3(b). Moreover, the value of l is negative if the value of p_2 is larger than 0.5.

Although the edges of the block B in Fig. 3 are located in the first quadrant of the x-y coordinates of B, the edge models can be applied to compute the translations of edges located at other quadrants by adjusting the values of edge orientation as follows

$$\theta' = i \times \frac{\pi}{2} - \theta, \text{ if } i = 2, 4, \text{ and} \tag{17a}$$

$$\theta' = \theta - (i-1) \times \frac{\pi}{2}, \text{ if } i = 3 \tag{17b}$$

where the value of *i* denotes which quadrant contains the center of gravity $(\bar{x}, \bar{y})$.

The final question to answer is that given a block, how is it decided should equation (15) or (16) be used to compute the edge parameter *l*? The edge as seen from Fig. 4 intersects the block *B* on opposite sides when the value of translation within the interval (-*t*, *t*) and the value of *t* can be computed as

$$t = \sqrt{2} \times \sin(\frac{\pi}{4} - \theta) = \cos\theta - \sin\theta \, . \tag{18}$$

Combining equations (15) and (18), we get

$$(\sin\theta - \cos\theta) \le (1 - 2p_2)\cos\theta \le (\cos\theta - \sin\theta) \, . \tag{19}$$

And hence, if

$$0.5 \times \tan\theta \le p_2 \le 1 - 0.5 \times \tan\theta \tag{19}$$

then the edge shown in Fig. 4 intersects the block *B* on opposite sides of *B*; otherwise, the edge intersects *B* on neighboring sides of *B*.

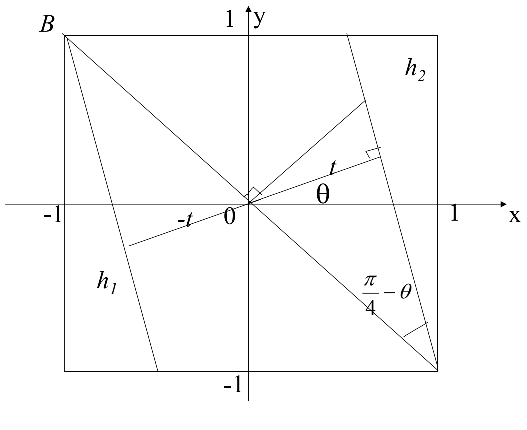


Fig. 4. Edges intersecting the block *B* on opposite sides, with the values of translation confined to the interval (-*t,t*).

An algorithm is given below to summarize the proposed moment-preserving edge detector.

Algorithm 1. Proposed moment-preserving edge detection technique (MPED).

Input. Square block B containing an edge feature.

Output. Parameters of the edge.

Method.

1. Compute the values of M_x, M_y and M_o from the circle inscribed inside B.
2. Calculate the value of θ using equation (7) and adjust θ using equation (17) to shift edges located at other quadrants of the x-y space to the first quadrant.
3. Apply the moment-preserving technique to find the solutions of the parameters h_1, h_2, p_1, and p_2 of the edge. If B is a gray-level block, apply the moment-preserving threshlding(16) technique, else use the quaternion-moment-preserving thresholding [18] technique.
4. Apply equation (19) to test how the edge intersects B. If the edge intersects B at opposite sides, use equation (15) to compute the value of translation l, else use equation (16).

As an example, suppose a gray-level block with size 4 x 4(as shown in Fig. 5(a)) is given and the detected edge and its parameters are shown in Fig. 5(b).

$$\begin{array}{cccc} 23 & 23 & 56 & 57 \\ 22 & 54 & 58 & 56 \\ 20 & 55 & 57 & 53 \\ 21 & 55 & 50 & 54 \end{array} \quad \begin{array}{l} M_0 = 167.11, \\ M_x = 10.61, \\ M_y = -1.66. \end{array} \quad \begin{array}{l} cos\theta = 0.9880, \\ sin\theta = -0.155, \\ \cos(2\pi-\theta) = 0.9880, \\ \sin(2\pi-\theta) = 0.155, \\ \tan(2\pi-\theta) = 0.1569. \end{array}$$

(a)

$$\begin{array}{cccc} 22 & 22 & 55 & 55 \\ 22 & 55 & 55 & 55 \\ 22 & 55 & 55 & 55 \\ 22 & 55 & 55 & 55 \end{array} \quad \begin{array}{l} h_1 = 22, \\ h_2 = 55, \end{array} \quad \begin{array}{l} p_1 = 0.3196 \\ p_2 = 0.6804 \end{array}$$

$$1-0.5*\tan(2\pi-\theta) > p_2 > 0.5*\tan(2\pi-\theta),$$

$$l = -0.3565.$$

(b)

Fig. 5. Applying the proposed moment-preserving edge detector to a gray-level block: (a) the original block and its computed values of moments M_0, M_x, M_y and normal vector ($cos\theta$, $sin\theta$) of the edge; (b) the detected edge and the values of parameters h_1, h_2, p_1, p_2, and l.

2.3 *Visual pattern mapping*

The proposed moment-preserving edge detector can detect an edge along any direction in a given image block. The difference between two edges with very close directions is not perceivable when the image resolution is high and the block size is small. In this paper, the block size 4 x 4 is adopted, which can be assumed to be small for high-resolution images. Instead of representing edges in any directions, the detected edges are mapping to 37 types of visual pattern, as shown in Fig. 6 This assumes that the possible directions of an edge in a 4 x 4 block are limited to multiples of 45°, or equivalently, $i \times 45^\circ$, i= 0, 1, …, 7. If the actual direction of an edge is not a multiple of 45°, it is quantized to be the nearest multiple of 45°. After the direction of an edge is given, we use the value of translation to map the edge to the nearest visual pattern. Note that each pixel in a visual pattern is either classified as a *Class0* pixel or a *Class1* pixel. A visual pattern is coded by a bitmap, consisted of *0*s (pixels in *Class0*) and *1*s (pixels in *Class1*) and two representative gray (color) values h_1 and h_2 for the *Class0* and *Class1*, respectively.

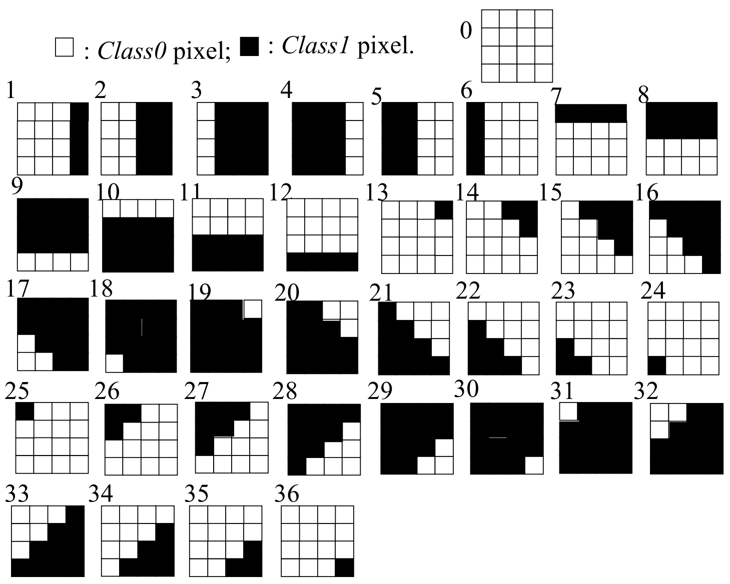


Fig. 6 Possible thirty-seven types of visual patterns in a 4 x 4 block.

The main reason for mapping an edge feature to a visual pattern is to encode an image with visual patterns block by block. The pixels of a visual pattern are divided into two classes, each of which is represented by a representative gray (or color) value. And hence, a given image block can be encoded by two representative gray (or color) values and an index to indicate which visual pattern the block is mapped. Both the edge and intensity (color) features of an image block are included in a visual pattern. Therefore, no extra information needs to be stored in the image database. The block-based visual pattern coding has two clear advantages: short computing time and less storage space (codes can be generated in realtime).

3. SIMILARITY MEASUREMENT WITH VISUAL PATTERNS

Using the scheme discussed in the last section, an image is represented with the visual-pattern codes of image blocks. In other words, the image content can be uniquely determined by the visual-pattern codes. Thus, it is possible to retrieve images that are similar to a query image based on a similarity measurement between the visual patterns of the database images and those of the query image. More identical pieces in the visual-pattern codes suggest more identical blocks, and therefore the two images are more similar. Consequently, the mean square error (MSE) of two visual-pattern codes, in which one is from a database image and the other is from a query image, can measure the similarity of two images with a lower MSE, indicating a greater degree of similarity. A lookup table *L*, referred to as pattern-similarity table can be constructed in advance to record the level of similarity of all pairs of visual-pattern types in terms of bitmaps for the purpose of efficiently computing the value of MSE of two images. Let b_i and b_j be the bitmaps of visual-pattern types *i* and *j*, respectively. The similarity of b_i and b_j can be measured by the following measurement: (1) the count of *1*s in $b_i \wedge b_j$; (2) the count of *1*s in $\neg b_i \wedge \neg b_j$; (3) the count of *1*s in $\neg b_i \wedge b_j$; (4) the count of *1*s in $b_i \wedge \neg b_j$. The larger the values of cases (1) and (2) are, the more similar the two visual-pattern types in terms of bitmaps. In contrast, the larger the values of cases (3) and (4) are, the less similar the two visual-pattern types in terms of bitmaps. We denote the values of the four cases above as $L^{1,1}_{i,j}, L^{1,0}_{i,j}, L^{0,1}_{i,j}$, and $L^{0,0}_{i,j}$, respectively, for illustration convenience. Note that the types of visual pattern can be classified as five classes, namely uniform (type 0), vertical (types 1 to 6), horizontal (types 7 to 12), diagonal (types 13 to 24), and anti-diagonal (types 25 to 36) classes by grouping similar types together.

Suppose two images *A* and *B,* are both coded by *M* visual patterns, the value of MSE between the images A and B is computed as follows:

$$MSE = \frac{1}{M}\sum_i [L^{1,1}_{A_i,B_i} \times (h_2^{A_i} - h_2^{B_i})^2 + L^{0,0}_{A_i,B_i} \times (h_1^{A_i} - h_1^{B_i})^2 + L^{1,0}_{A_i,B_i} \times (h_2^{A_i} - h_1^{B_i})^2 + L^{0,1}_{A_i,B_i} \times (h_1^{A_i} - h_2^{B_i})^2] \quad (20)$$

where A_i and B_i denote the type numbers of the *i*th visual pattern of *A* and *B*, respectively; and $h_1^{A_i}$ and $h_2^{A_i}$ ($h_1^{B_i}$ and $h_2^{B_i}$) are the two representative values of the *ith* visual pattern of the image *A* (*B*). The values of $L^{1,1}_{A_i,B_i}, L^{1,0}_{A_i,B_i}, L^{0,1}_{A_i,B_i}$, and $L^{0,0}_{A_i,B_i}$ can be easily obtained by using the pattern-similarity table mentioned above, and hence the cost to compute the value of MSE can be reduced.

Let us consider two images, I_1 and I_2, where the size of I_1 is not smaller than that of I_2. There are three cases to compare the content of I_1 and I_2: (1) I_1 and I_2 are two identical images; (2) I_2 is a sub-image of I_1; (3) I_1 and I_2 have the same sub-image. Case 1 is the simplest, where the visual-pattern codes at identical locations in both I_1 and I_2 are identical because they have the identical pixel values. And hence, it is easy to retrieve those images, which are the same as the query image from a database, by comparing their visual-pattern codes.

Cases (2) and (3) are more complicated because the blocks to be compared in two images are not identical, although they have an identical sub-image, which leads to an incorrect decision that these two images are not similar or they have a low matching rate. The common blocks of I_1 and I_2 must be aligned properly for these two cases, as shown in Figs. 7(a) and 7(b), respectively. As a rule of thumb, two images are more similar if they have more identical blocks, that is a lower value of MSE.

4. MATCHING STRATEGY FOR IMAGE RETRIEVAL

In order to retrieve images we must be able to efficiently compare two images, where one is a database image and one is the query image to determine if they have similar content. A simple approach for image retrieval with visual-pattern coding is to encode both images and then compute the matching rate between their visual-pattern codes. However, the visual-pattern code of a query image could be very different from those of similar database images because the objects of the query image might be translated, rotated, scaled, or distorted; so this direct method may produce rather poor results.

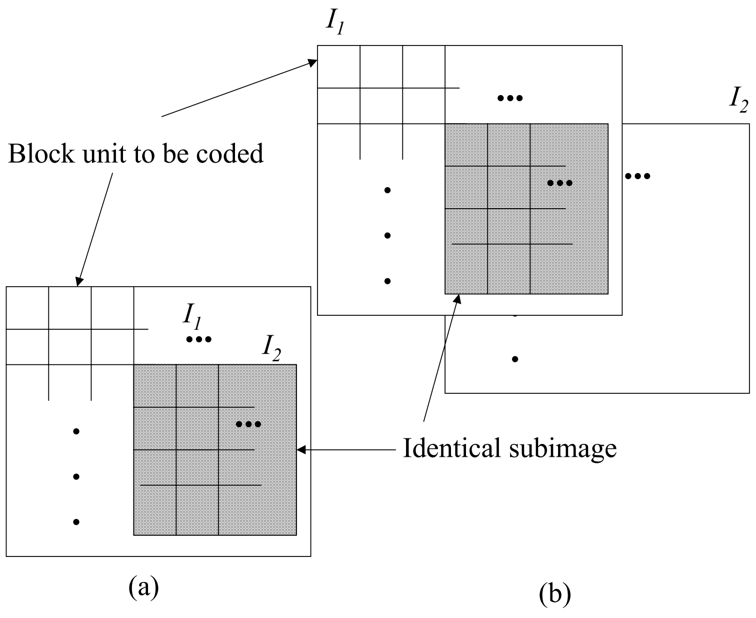


Fig. 7 Aligning image I_2 to I_1 for computing the matching rates: (a) I_2 is a sub-image of I_1; (b) I_1 and I_2 have the same sub-image.

4.1 *Histograms of visual-pattern type*

This block-based visual-pattern coding scheme provides two advantages: lower computational complexity and less storage space. However, it also generates problems in two situations described as follows:

(1) A process to align the common blocks of a query image with those of a database image must be done, as shown in Fig. 7 The representation of an image with visual-pattern code is not rotation or scale invariant, but is translation invariant if the identical sub-image of the two images is aligned properly.

(2) The retrieval by matching the visual-pattern codes of database images and a given query image directly may receive responses from irrelevant matching pairs because if two images have the same sub-image, then parts of the visual-pattern codes for both two images are overlapping. The problem is how to efficiently find the highly co-related parts.

To overcome the problems from both situations above, the 1-D histograms of visual-pattern types are computed from a given query image and database images to eliminate irrelevant matching. If the histograms of two images, in which one is a database image and the other is a given query image are quite different, then the visual-pattern codes between the two images need not match.

Basically, the histogram of visual-pattern types represents the shape attribute of an image and is invariant to translation in the image. However, this also turns out to be a limitation, as two totally different images may yield similar histograms of visual-pattern types. A suitable normalization of the histograms is helpful to provide scale invariance. Let *H(i)* be a histogram of the visual-pattern types of an image, where the index *i* represents a type number. Then, the normalized histogram $\overline{H}$ is defined as follows:

$$\overline{H}(i) = \frac{H(i)}{\sum_{i=0}^{36} H(i)} \quad . \tag{21}$$

A drawback of normalized histograms is the inability to match parts of images. If an image *Q* is a part of an image *I*, then the histogram of *Q* is contained within the histogram of *I*. Normalizing the histograms does not satisfy this property. Furthermore, a histogram of visual-pattern types is also not invariant to rotation. A shift of the histogram bins during matching will partially take into account rotation of the images. For example, if two edges with the same edge translation but different orientations of 30 and 70° will fall into the same histogram bin (one of diagonal types). However, if the same image is rotated by 10°, then these two edges will fall into different bins, in which one is from the horizontal class and the other is from the diagonal class. To reduce this effect of rotation we smooth the histograms using the following equation:

$$\overline{H}_s(i) = \frac{\overline{H}(i) + \sum_{j \in \zeta_i} \overline{H}(j)}{1 + |\zeta_i|} \tag{22}$$

where ζ_i denotes the set of the neighboring histogram bins of the *ith* histogram bin. One histogram bin is said to be a neighbor of another bin if a visual pattern will switch the bins with each other by a rotation. Note that an edge block coded by a visual pattern with its type from the diagonal (or anti-diagonal) class may change the type to the vertical or horizontal classes by a rotation. On the other hand, a type from the vertical (or horizontal) class might change its type to the diagonal or anti-diagonal classes. Let C_H^k, C_V^k, C_D^k, and C_{AD}^k denote the *kth* type from the horizontal, vertical, diagonal, and anti-diagonal classes of visual-pattern types, respectively. There are four cases to determine the neighborhood ζ_i of the *ith* histogram bin, dependent on the class of the *ith* histogram bin associated, and which can be defined as $(C_A^{2k-1}, C_A^{2k}, C_{AD}^{|C_{AD}|-2k+1}, C_{AD}^{|C_{AD}|-2k+2})$, $(C_A^{2k-1}, C_A^{2k}, C_{AD}^{2k-1}, C_{AD}^{2k})$, $(C_H^{\lceil k/2 \rceil}, C_V^{\lceil k/2 \rceil})$, and $(C_H^{\lceil k/2 \rceil}, C_V^{|C_V|-\lfloor k/2 \rfloor})$ for the horizontal, vertical, diagonal, and anti-diagonal classes of visual-pattern types, respectively.

Let $\overline{H}_{Q,s}$ and $\overline{H}_{I,s}$ be the normalized and smoothed histograms for a query image *Q* and a database image *I*, respectively. The similarity in terms of the visual-pattern histogram between *Q* and *I*, $S_h(I,Q)$, is given by the following equation:

$$S_h(I,Q) = 1.0 - \sqrt{\frac{\sum_i (\overline{H}_{Q,s}(i) - \overline{H}_{I,s}(i))^2}{2}} \ . \tag{23}$$

Note that the value of $S_h(I,Q)$ lies in the interval [0,1]. If images *I* and *Q* are identical then $S_h(I,Q) = 1$. Fig. 8 shows three database images [(a) and (b) are similar but (c) is different] and their respective hi stograms of visual-pattern types. Note that $S_h(a,b) > S_h(b,c)$ and $S_h(a,b) > S_h(a,c)$.

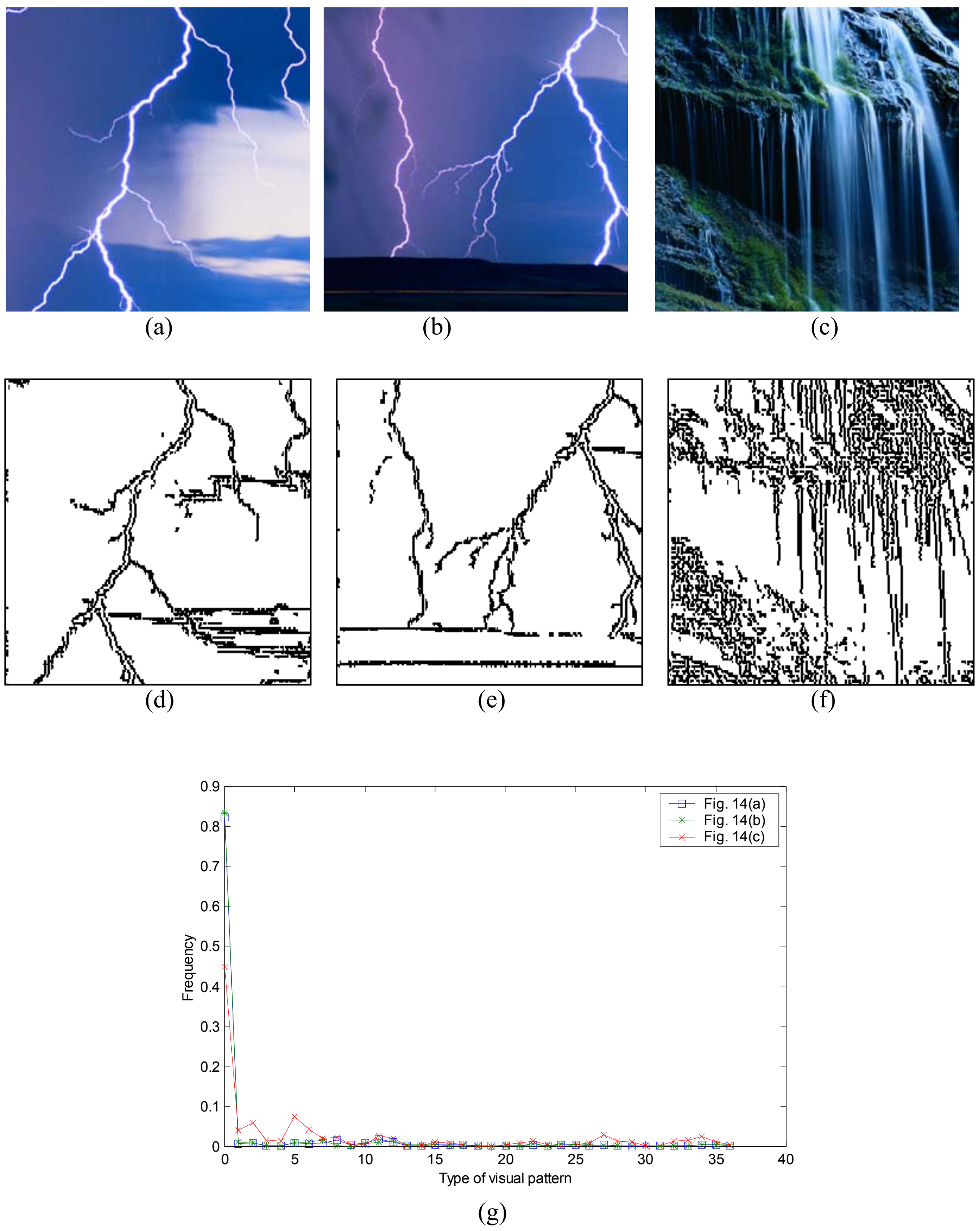


Fig. 8 The visual-pattern histograms for three database images: (a)-(c) show three database images; (d)-(f) show the corresponding edge images; and (g) shows the corresponding histograms: $S_h(a,b)$ = 0.984, $S_h(a,c)$ = 0.724, $S_h(b,c)$ = 0.717.

4.2 *Matching strategy*

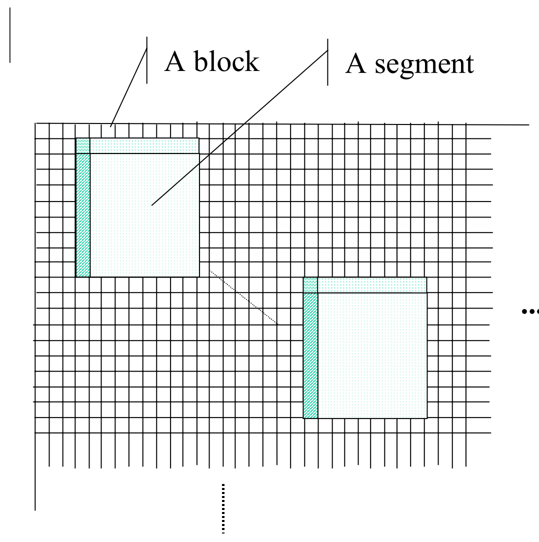
It is not wise to match the visual-pattern codes of the query image globally to those of the database images. We can reduce the number of irrelevant matches by constraining the matching to the segments of the database images. The segment size would depend on the resolution of the query image. Let *B* x *B* be the size of the block and *S* x *S* denote the size of an image segment. The value of S should be chosen as a multiple of the value of B and hence there are $\frac{S}{B} \times \frac{S}{B}$ blocks in a segment. Given a query image *Q* with size $N \times N$, the image is tailored by removing the boundary rows and columns for the purpose of dividing *Q* into $\left\lfloor \frac{N}{S} \right\rfloor \times \left\lfloor \frac{N}{S} \right\rfloor$ segments. Each segment of *Q* is then applied to match the same size segments of a database image, and the average of the matching rates of all the segments of *Q* is then obtained as the matching rate of the whole query image.

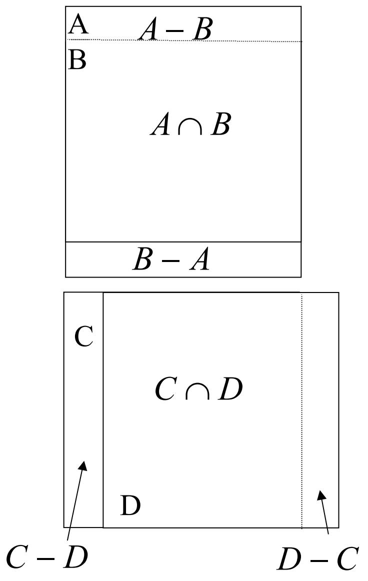
If the size of a database image is $M \times M$, the number of segments to be searched for each segment of *Q*, as seen from Fig. 9a, would be $\left\lceil \frac{M - S}{B} \right\rceil^2$. For example, if the values of *B*, *S* and *M* are 4, 64 and 256, respectively, then the number of segments to be searched for the database image is 2304. This is a large number and hence a long time is required to find the best matches among the segments of a database image for each segment of a query image.

To shorten the searching time for each segment, the matching rates on the basis of visual-pattern histograms for each segment in a database image and a query image are first computed. Only the segments in a database image with sufficiently large values of similarity in terms of histograms are considered for computing the eventual matching scores based on the visual-pattern codes for each query segment. That is, the visual-pattern histogram for each query segment works as a filter to eliminate the irrelevant segments of a database image. As shown in Fig. 9(b), the histogram of visual-pattern types of the segment *B* (*D*) can be constructed directly by replacing the types of blocks from the set *A-B* (*C-D*) with the types of blocks in the set *B-A* (*D-C*) from the histogram of the segment *A* (*C*). This further shortens the time of filtering.

4.3 *Proposed image retrieval algorithm*

In this study, the size of the segment in an image is set to 128 x 128 or 64 x 64 dependent on the resolution of the query image. If the size of the query image is close to 64 x 64, then the size of segments will be set to 64 x 64; otherwise, it will be set to 128 x 128. When the retrieval of larger query image is submitted, the query image is first decomposed into several segments of size 128 x 128, the average of the matching scores (MSEs) of all the segments is then obtained as the matching score of the whole query image.





(a) (b)

Fig. 9 Segment decomposition of a database image: (a) search for the best match segment of the database image for each query segment; (b) the histogram of visual-pattern types of the segment *B* (*D*) can be constructed directly by replacing the types of blocks from the set *A-B* (*C-D*) with the types of blocks in the set *B-A* (*D-C*) from the histogram of the segment *A* (*C*).

The complete matching scheme between a database image *I* and a query image *Q* is given as the following algorithm.

Algorithm 2. Proposed image retrieval technique.

Input. A query image *Q* of size $M \times M$ and a database image *I*.

Output. A set of match images from a database.

Method.

(1) Encode the database image *I* using the proposed visual-pattern coding.

(2) If the value of *M* <128, set the size of an image segment to be 64 x 64; otherwise, set the size of a segment to be 128 x 128.

(3) Tailor *Q* into several segments by removing boundary rows and columns so that the size of *Q* is a

multiple of the size of a segment.

(4) For each query segment Q_s, do the following steps.

(4.1) Encode Q_s with our proposed visual-pattern coding scheme.

(4.2) Compute the visual-pattern histogram of Q_s. Match the histograms of the database image I with that of the query segment Q_s using equation (23), and obtain the segments of I whose values of similarity are larger than a predefined threshold (i.e. 0.97).

(4.3) Match the visual-pattern codes of those segments of I obtained in the step (4.2) with those of the query segment Q_s and obtain the smallest value of MSE (defined in equation (20)).

(5) Average the values of MSE of all query segments as the matching score between the query image Q and the database image I.

We can perform the above operation for each database image and retrieve with the lower values of MSE.

5. EXPERIMENTAL RESULTS

In order to evaluate the proposed approach, a series of experiments was conducted on an Intel PENTIUM-IV 1.5GMhz PC and a color image database consisted of 675 scenery images was used. Each image in the database is first tailored to the size of 256 x 256 for testing the retrieval approach. Query images of different sizes were extracted from these images.

A retrieval method is classified as accurate if, for a given query image, the perceptually (to a human) most similar image in the database is retrieved by the system as the topmost retrieval. Also, a robust system should be stable for all types of queries, i.e., the system must not break down under

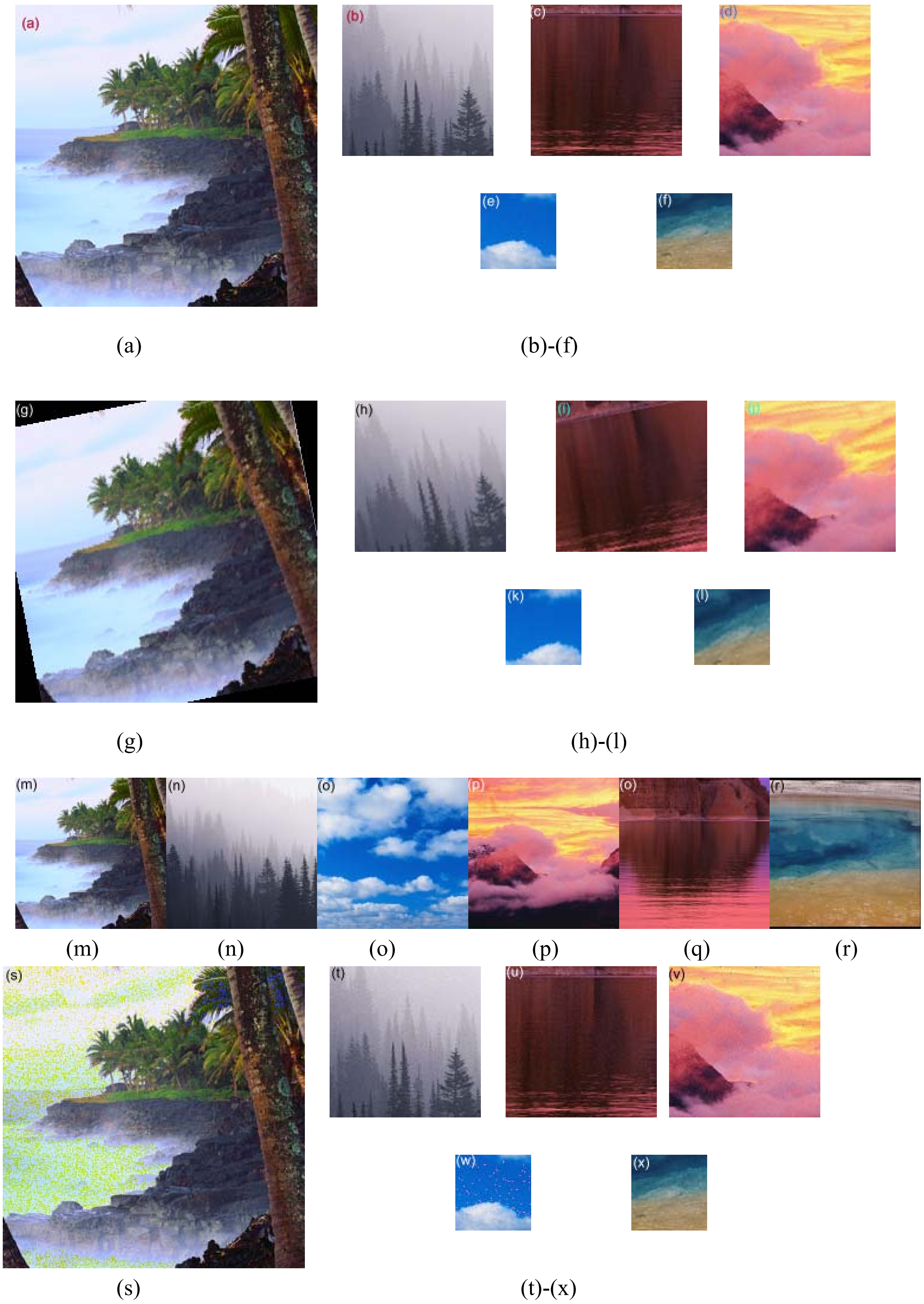


Fig. 10. A sample of test images: (a)-(f) show six database images of different sizes; (g)-(l) show the corresponding rotated images; (m)-(r) show the corresponding scaled images; and (s)-(x) show the corresponding noisy images.

specific cases. In order to test the robustness of the proposed system, not only normal query images, but also rotated, scaled, and noise added query images were used to test the system. Fig. 10 shows a sample of test images.

For retrieval on the basis of visual-pattern coding, the experiments were conducted on the 100 color images randomly selected from our database as the query images. Tables 1 and 2 present the retrieval results on the basis of histograms of visual-pattern types and visual-pattern codes, respectively, where *n* refers to the position of the correct retrieval. The retrieval performance was better in the presence of scale changes than in the presence of rotated or noisy images if histograms of visual-pattern types were used as features. However, if we directly match the

Table1 Image retrieval results using histograms of visual-pattern types: *n* refers to the position of the correct retrieval; the last column indicates the average time taken for a retrieval.

Query images		*n* = 1 (%)	*n* <= 2 (%)	*n* <= 3 (%)	*n* <= 4 (%)	*n* <= 10 (%)	*n* <= 20 (%)	Retrieval time (seconds)
Size	Nature							
256 x 256	Normal	100	100	100	100	100	100	0.804
	Scaled	32	40	48	54	70	89	0.452
	Noisy	32	40	43	50	60	72	0.804
128 x 128	Normal	100	100	100	100	100	100	0.451
	Rotated	31	50	59	64	83	95	0.451
	Scaled	13	20	28	34	56	81	0.307
	Noisy	30	40	42	45	55	65	0.451
64 x 64	Normal	100	100	100	100	100	100	0.307
	Rotated	30	51	57	64	81	96	0.309
	Noisy	27	34	40	44	54	67	0.307

Table 2 Image retrieval results by matching the visual-pattern codes directly: *n* refers to the position of the correct retrieval; the last column indicates the average time taken for a retrieval.

Query images		*n* = 1 (%)	*n* <= 2 (%)	*n* <= 3 (%)	*n* <= 4 (%)	*n* <= 10 (%)	*n* <= 20 (%)	Retrieval time (seconds)
Size	Nature							
256 x 256	Normal	100	100	100	100	100	100	9.374
	Scaled	52	73	81	84	93	97	2.448
	Noisy	100	100	100	100	100	100	9.374
128 x 128	Normal	100	100	100	100	100	100	2.438
	Rotated	86	91	93	98	100	100	2.448
	Scaled	61	73	78	83	93	98	1.372
	Noisy	100	100	100	100	100	100	2.438
64 x 64	Normal	100	100	100	100	100	100	1.372
	Rotated	73	83	90	95	98	98	1.420
	Noisy	100	100	100	100	100	100	1.420

Table3 Image retrieval results of integrating queries on the basis of both the type histograms and visual-pattern codes: *n* refers to the position of the correct retrieval; the last column indicates the average time taken for a retrieval.

Query images		*n* = 1 (%)	*n* <= 2 (%)	*n* <= 3 (%)	*n* <= 4 (%)	*n* <= 10 (%)	*n* <= 20 (%)	Retrieval time (seconds)
Size	Nature							
256 x 256	Normal	100	100	100	100	100	100	3.895
	Scaled	52	70	74	77	91	96	1.021
	Noisy	81	85	86	89	93	95	3.895
128 x 128	Normal	100	100	100	100	100	100	1.023
	Rotated	84	91	93	95	99	99	1.023
	Scaled	55	70	77	80	90	97	0.503
	Noisy	75	83	87	90	92	96	1.512
64 x 64	Normal	100	100	100	100	100	100	0.503
	Rotated	84	93	95	97	99	99	0.503
	Noisy	75	84	87	92	93	97	0.503

visual-pattern codes of two images, the retrieval performance of the proposed system seems more sensitive to scaled images than to rotated or noisy images. The worst-case retrieval accuracy of the system was 97%. Moreover, the time taken for a retrieval on the entire database was much shortened by using histograms of visual-pattern types. Table 3 presents the results of integrating queries on the basis of both type histograms and visual-pattern codes for the 100 color images. The performance is comparable to that of matching visual-pattern codes directly, but the retrieval efficiency is much improved.

The retrieval technique based on edge and color histograms proposed by Jain and Vailaya [12] was also implemented for performance comparison. Before the evaluation, human assessment was done to determine the relevant matches in the database to the query images. The top 100 retrievals from both the color and shape histograms and the proposed approaches were marked to decide whether they were indeed visually similar in color and shape. The retrieval accuracy was measured by precision and recall

$$\text{Precision}(K) = C_K/K \text{ and } \text{Recall}(K) = C_K/M \tag{24}$$

where K is the number of retrievals, CK is the number of relevant matches among all the K retrievals, and M is the total number of relevant matches in the database obtained through human assessment. The average precision and recall curves are plotted in Figs. 11 and 12. It can be seen that the proposed method achieves good results in terms of retrieval accuracy compared to Jain's method.

Let us consider a query image of size 256 x 256, as shown in Fig. 13. The segment size is chosen as 128 x 128. The first 11 retrieval images with the lowest values of MSE as 46.76, 86.79, 97.61, 131.89, 142.85, 152.33, 156.61, 159.54, 162.30, 163.44 and 167.22 are shown in Figs. 13(b), (c), (d), (e), (f), (g), (h), (i), (j), (k) and (l) respectively. All these images have similar sub-images of a stream, which indicates that the proposed retrieval scheme also can retrieve similar images based on an iconic image.

In order to speed up the computationally expensive search over the entire database, the similarity between two histograms of visual-pattern types for a database image segment and a query segment is first used as a filter step. This dramatically shortens the time required to retrieve images without sacrificing the retrieval accuracy. Furthermore, the histograms of segments in a database can first be clustered to further shorten the retrieval time.

6. CONCLUSION

In this paper we have present a block-based edge detection algorithm using the moment-preserving techniques. Based on the edge detector, an image can be coded by visual-pattern codes block by block in real time. A matching strategy based on the visual-pattern codes also makes the retrieval process robust and accurate, and the method using the visual-pattern type histograms to filter out dissimilar images dramatically shortens the time required to retrieve images, without sacrificing the retrieval accuracy.

The speed of retrievals can also be increased without affecting the robustness of the system by clustering the type histograms of a database in advance. Future work will deal with linking edge patterns into objects to the proposed system, and increasing the database size.

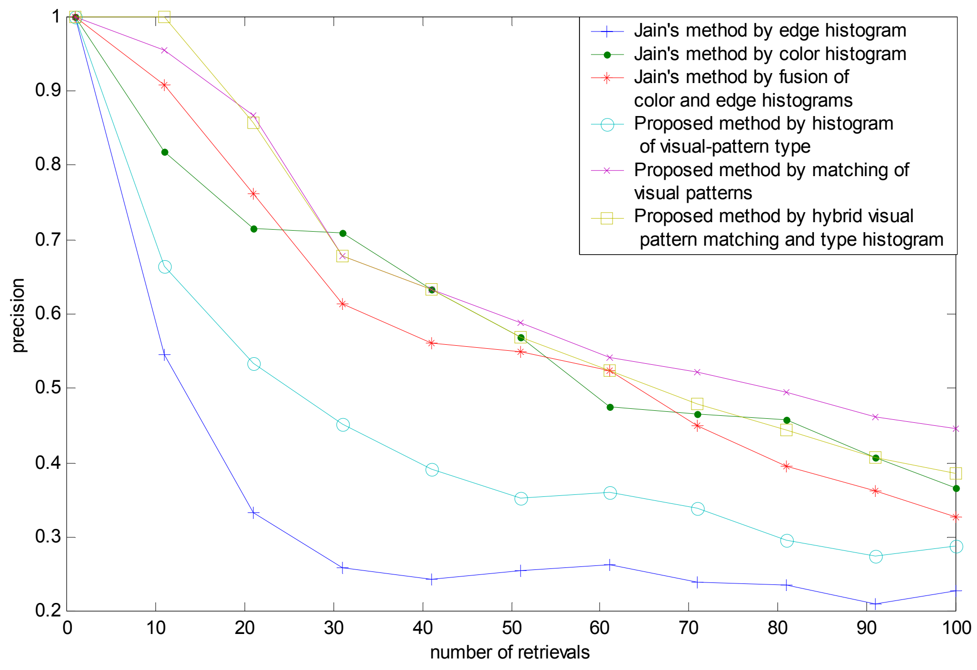


Fig. 11. Average precision versus number of retrievals.

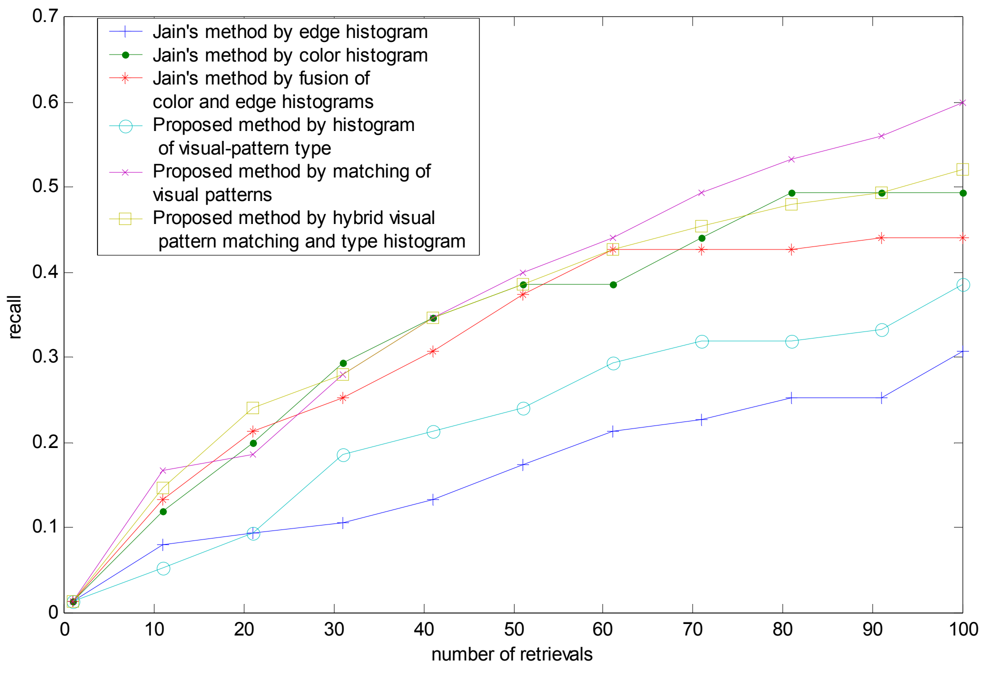


Fig. 12. Average recall versus number of retrievals.

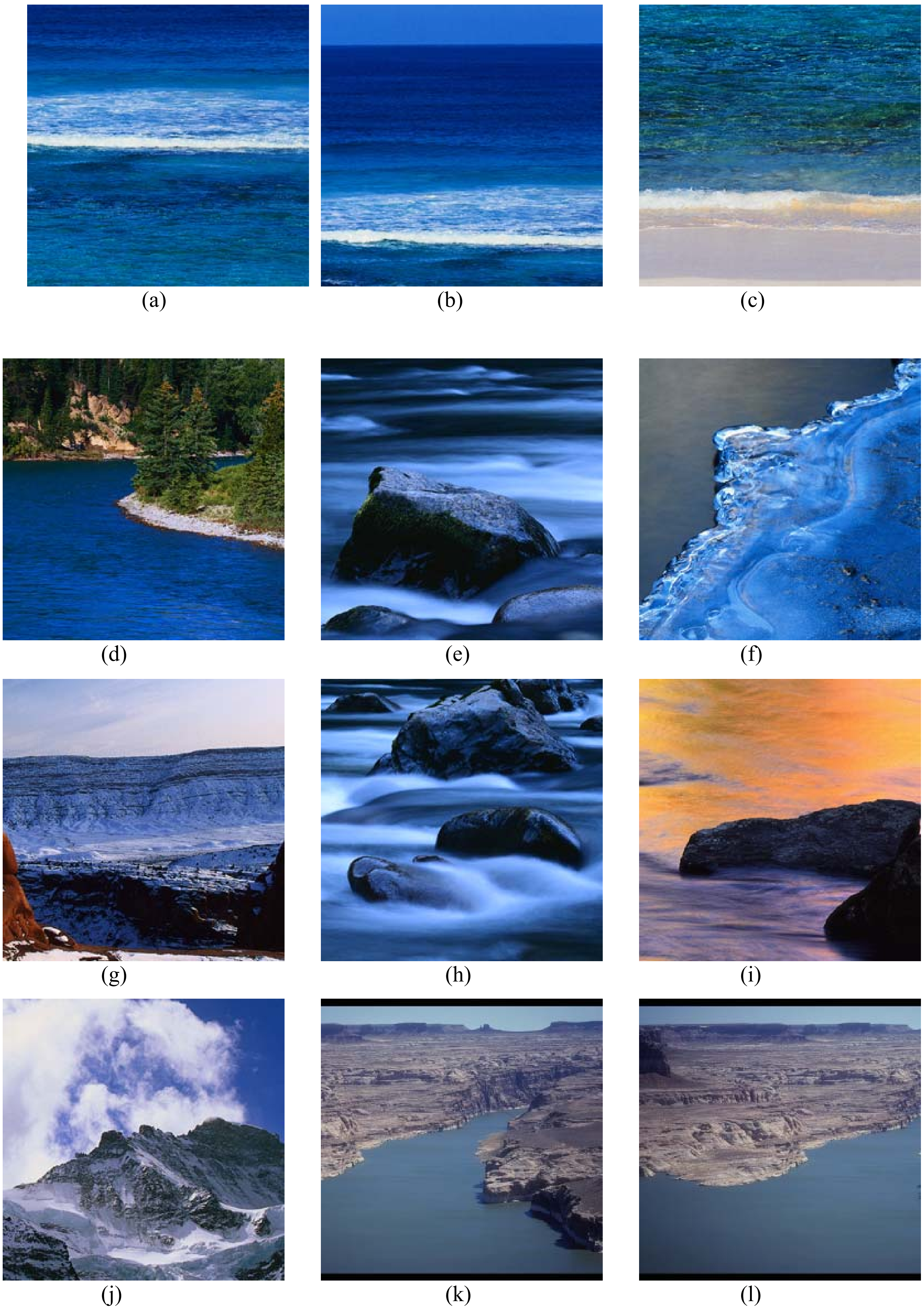


Fig. 13. A query image and retrieved images: (a) the query image, (b)-(l) the first 11 retrieved images with lowest values of MSE as 46.76, 86.79, 97.61, 131.89, 142.85, 152.33, 156.61, 159.54, 162.30, 163.44 and 167.22.

REFERENCES

[1] C. Faloutsos et. al., Efficient and effective querying by image content, *Journal of Intelligent Systems*, 1, 95-108 (1994).

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

[3] A. Pentland, R. Picard, and S. Scalroff, Photobooks: Tools for Content-based manipulation of image databases, *SPIE Conf. On Storage and Retrieval of Image and Video Databases II*, 33-47 (1994).

[4] B. M. Mether, M. S. Kankanhall, and W. F. Lee, Content-based image retrieval using a composite color-shape Approach, *Information Processing and Management,* 34(1) 109-120 (1998).

[5] M. Swain and D. Ballard, Color indexing, *Int. J. Comput. Vision,* 7(1), 11-32 (1991).

[6] 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, 744-749 (1997).

[7] X. Wan and C.-C. Jay Kuo, A new approach to image retrieval with hierarchical color clustering, *IEEE Trans. On Circuits and Systems for Video Technology,* 8(5), 628-643 (1998).

[8] S.C. Pei and C.M. Cheng, Extracting color features and dynamic matching for image data-base retrieval, *IEEE Trans. On Circuits and Systems for Video Technology,* 9(3), 501-512 (1999).

[9] B. Manjunath and W. Ma, Texture features for browsing and retrieval of image data, *IEEE Trans. Pattern Anal. Machine Intell.*, 18, 837-842 (1996).

[10] Z. Wang, Z. Chi and D. Feng, Content-based image retrieval using block-constrained fractal coding and nona-tree decomposition, *IEE Proc.-Vis. Image Signal Process*, 147(1), 9-15 (2000).

[11] R. Methrotra and J. Gary, Similar-shape retrieval in shape data management, *IEEE Computer*, 28, 57-62 (1995).

[12] A. K. Jain and A. Vailaya, Image retrieval using color and shape, *Pattern Recognition*, 29, 1233-1244 (1996).

[13] S. C. Cheng and W. H. Tsai, Image compression by moment-preserving edge detection, *Pattern Recognition*, 27(11), 1439-1449 (1994).

[14] E. J. Delp and O. R. Mitchell, Image compression using block truncation coding, *IEEE Trans. Commun.*, 27, 1335-1341 (1979).

[15] A. J. Tabatabai and O. R. Mitchell, Edge location to subpixel values in digital imagery, *IEEE Trans. Pattern Anal. Mach. Intell.*, 6, 188-201 (1984).

[16] W. H. Tsai, Moment preserving thresholding: A new approach, *Comput. Vis., Graph., Image Process.*, 377-393 (1984).

[17] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice-Hall (1991).

[18] S. C. Pei and C. M. Cheng, Color image processing by using binary quaternion-moment-preserving thresholding technique, *IEEE Trans. Image Process.*, 8, 614-628 (1999).

[19] A. Rosenfield and A. C. Kak, *Digital Picture Processing*, Vol II. Academic Press, New York (1982).

[20] H. C. Lee and D. R. Cok, Detecting boundaries in a vector field, *IEEE Trans. Signal Processing*, 39, 1181-1194 (1991).

[21] P. E. Trahanias and A. N. Venetsanopoulos, Vector order statistics operators as color edge detectors, *IEEE Trans. Syst., Man, Cybern.*, SMC-26, 135-143 (1996).