Int. J. Electron. Commun. (AEÜ) 70 (2016) 833–841

Contents lists available at ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

Robust image hashing based on color vector angle and Canny operator

Zhenjun Tang[a,b,*], Liyan Huang[a,b], Xianquan Zhang[a,b], Huan Lao[a,b]

[a] Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin 541004, China
[b] Department of Computer Science, Guangxi Normal University, Guilin 541004, China

ARTICLE INFO

Article history:
Received 28 August 2015
Accepted 17 March 2016

Keywords:
Image hashing
Color vector angle
Image edge
Canny operator

ABSTRACT

Image hashing is a novel technology of multimedia processing, and finds many applications, such as image forensics, image retrieval and image indexing. Conventional image hashing algorithms have limitations in reaching desirable classification performances between rotation robustness and discrimination. Aiming at this issue, we propose a robust image hashing based on color vector angle and Canny operator. Specifically, our hashing firstly converts input image to a normalized image by interpolation and Gaussian low-pass filtering. And then, color vector angles and image edges are both extracted from the normalized image. Finally, statistical features incorporating color vector angles and image edges are calculated to form image hash. We conduct experiments with 2762 images to validate efficiency of our hashing. The experimental results show that our hashing is robust against normal digital processing, such as image rotation, brightness/contrast adjustment and JPEG compression, and reaches good discrimination. Receiver operating characteristics (ROC) curve comparisons with some state-of-the-art algorithms indicate that our hashing outperforms these compared algorithms in classification performances between robustness and discriminative capability.

1. Introduction

Image hashing is a new and hot topic of multimedia processing. It uses a short string called image hash to denote an image, and has been widely applied to many applications [1], such as image authentication, image forensics, image retrieval, image indexing and image copy detection. In practice, digital images often undergo some normal processing such as JPEG compression, geometric transform, and format conversion. After these processing, visual appearances between the original and processed images are unchanged, but digital representations are quite different. So their image hashes are expected to be the same or very similar. In general, image hashing has two basic properties [1–3] as follows. (1) *Perceptual robustness*: Image hashing should map visually identical images to the same or very similar hashes regardless of their digital representations. In other words, image hash should be robust against normal digital processing, such as image compression and image enhancement. (2) *Discriminative capability*: For different images, image hashing should produce different image hashes. Except these basic properties, image hashing must satisfy other property for some specific applications. For example, it should be secure enough (e.g., controlled by keys) for application to image forensics.

In the past years, many researchers have devoted themselves to developing image hashing. The pioneer work was introduced by Schneider and Chang [4]. From then on, image hashing has attracted much attention in multimedia community. At first, researchers used discrete wavelet transform (DWT), discrete cosine transform (DCT) and discrete Fourier transform (DFT) to develop image hashing. For example, Venkatesan et al. [5] exploited DWT coefficients statistics to construct image hashes. This algorithm is robust against JPEG compression, median filtering and rotation within 2°, but fragile to gamma correction and contrast adjustment. Fridrich and Goljan [6] used DCT coefficients to design hashing function. This method is sensitive to image rotation. Swaminathan et al. [7] used DFT coefficients to produce image hashes. This hashing can resist digital processing such as moderate geometric transforms and filtering. Monga and Evans [8] detected visually significant feature points with the end-stopped wavelet transform and exploited them to construct hashes. Later, Radom transform (RT) was also taken for hash generation. For example, Lefèbvre et al. [9] were the first of using RT to design image hashing. Motivated by RT, Roover et al. [10] introduced a RASH method based on radial projections of image pixels and 1-D DCT. This scheme is robust to rotation, but its discrimination is not good enough. In another work, Ou and Rhee [11] applied RT to input image, randomly selected 40 projections to perform 1-D DCT, and took the first AC coefficient of each projection

* Corresponding author at: Department of Computer Science, Guangxi Normal University, 15 Yucai Road, Guilin 541004, PR China. Tel.: +86 15295990968.
E-mail address: tangzj230@163.com (Z. Tang).

http://dx.doi.org/10.1016/j.aeue.2016.03.010
1434-8411/

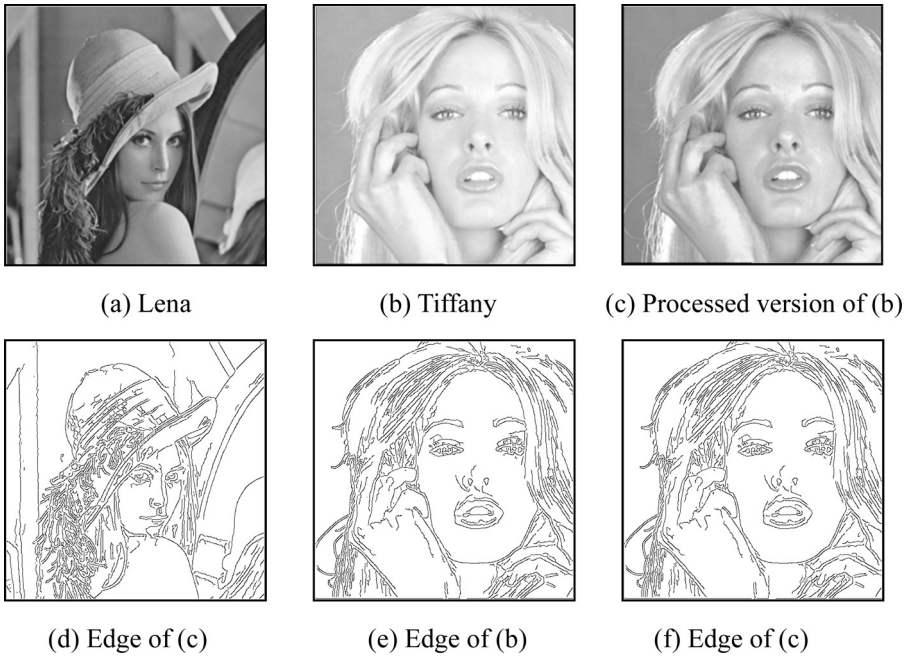


Fig. 1. Standard test images and their edges.

to produce hash. The RT-DCT hashing is resistant to rotation within 5°.

Besides the above strategies, researchers also applied other techniques to image hashing. For example, Kozat et al. [12] viewed image and attacks as a sequence of linear operators, and proposed to calculate hashes with singular value decompositions (SVDs). The SVD–SVD hashing is robust to rotation at the cost of significantly decreasing discrimination. Monga and Mihcak [13] firstly proposed to use non-negative matrix factorization (NMF) to derive image hashing, and obtained a high performance algorithm. This hashing is resilient to geometric attacks, but it cannot resist some normal manipulations, e.g., watermark embedding. Tang et al. [14] designed a lexicographical image hashing based on DCT and NMF. This algorithm is resilient to image rotation within 1°. Recently, Li et al. [15] calculated hashes using random Gabor filtering (GF) and dithered lattice vector quantization (LVQ). The GF–LVQ hashing is resistant to JPEG compression and rotation, but its discrimination is not good enough. Tang et al. [16] calculated histogram of color vector angles (HC) and compressed it by DCT. The HC-DCT hashing can tolerate rotation with any angle, but its discrimination is still not desirable. In another work [17], Tang et al. proposed a robust hashing with local image entropies and DWT. This approach can tolerate rotation within 5°. Laradji et al. [18] exploited quaternion Fourier transform (QFT) to construct image hashes. The QFT hashing is also sensitive to rotation. Zhao et al. [19] exploited Zernike moments (ZM) to calculate image hashes. The ZM-based hashing only tolerates rotation within 5°. Tang et al. [20] presented a robust image hashing with local moment invariants. This method can only tolerate rotation with 2°.

From the above review, it is found that most hashing algorithms are sensitive to rotation or robust against small angle rotation, such as [5,6,8,14,17–20]. Some methods can tolerate rotation with any angle, but their discriminations are not good enough, such as [9,15,16]. Therefore, it is still a challenging task to develop hashing method reaching a desirable trade-off between rotation robustness and discrimination. In addition, most algorithms are designed for gray images. For color images, they often choose the luminance component in YCbCr color space for representation. As other color components are discarded, their discriminative capability is limited. Aiming at these problems, we propose a robust image hashing based on color vector angle and Canny operator. Our algorithm can reach good rotation robustness since our features extracted from circles are invariant to rotation. Moreover, the use of color vector angle provides our algorithm a desirable discrimination. This is because color vector angle takes all RGB color components into account and then makes our method discriminative. We conduct experiments with 2762 images (i.e., 2562 images for robustness and 200 images for discrimination) to validate our efficiency. Experimental results indicate that our hashing reaches good trade-off between rotation robustness and discrimination, and outperforms some state-of-the-art hashing algorithms.

The rest of this paper is organized as follows. Section 2 describes the proposed image hashing. Sections 3 and 4 present the experimental results and performance comparisons, respectively. Conclusions are finally drawn in Section 5.

2. Proposed image hashing

A key step of image hashing is to extract robust feature invariant to normal digital processing. Clearly, edge is an important visual image feature and human visual system (HVS) can distinguish images in terms of their image edges. For example, Fig. 1(a) and (b) are two standard test images and (d) and (e) are their image edges. Although gray information is discarded in (d) and (e), HVS can also recognize that (a) is different from (b) according to their edge pixels' distribution. In addition, we find that image edge is almost kept unchanged after normal digital processing. For example, we decreased the brightness of Tiffany with Photoshop and obtained a processed version as shown in Fig. 1(c). Then, we detected image edge of Fig. 1(c) by Canny operator and generated the edge image as shown in Fig. 1(f). It is observed that there is no significant difference between Fig. 1(e) and (f). In other words, image edges of Fig. 1(e) and (f) are almost the same. Considering that edge can help to distinguish different images and is almost the same after normal digital processing, we investigate the use of image edge and thus propose a robust image hashing with color vector angle and image edge.

Our image hashing consists of three steps, as shown in Fig. 2. In the first step, input image is converted to a normalized image for

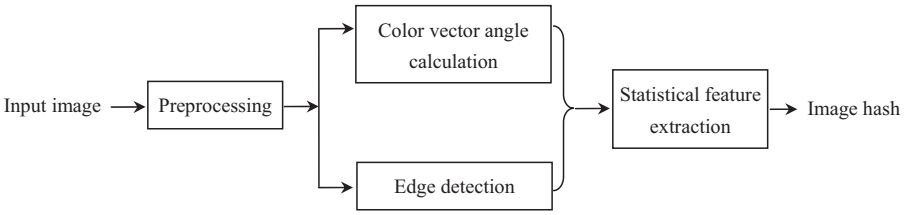


Fig. 2. Block diagram of our robust image hashing.

robust feature extraction. In the second step, the normalized image is processed with two operations separately. One is to calculate color vector angle and the other is to detect image edge. In the final step, statistical features based on image edge and color vector angle are extracted to form image hash. The detailed steps of our hashing are presented in the following subsections.

2.1. Preprocessing

The input image is firstly converted to a fixed size $M \times M$ with bicubic interpolation. The resizing operation is to ensure that those images with different resolutions have the same or very similar hashes. The square image is then blurred with a Gaussian low-pass filter. This operation is to reduce those high-frequent components easily influenced by image modifications, e.g., noise contamination and filtering. It can be done by a convolution mask. Let $T_{\text{Gaussian}}(i,j)$ be the element in the ith row and the jth column of the convolution mask. Thus, it is defined as

$$T_{Gaussian}(i,j) = \frac{T^{(1)}(i,j)}{\sum_i \sum_j T^{(1)}(i,j)} \tag{1}$$

in which $T^{(1)}(i,j)$ is calculated by

$$T^{(1)}(i,j) = e^{\frac{-(i^2+j^2)}{2\sigma^2}} \tag{2}$$

where σ is the standard deviation of all elements in the convolution mask.

2.2. Color vector angle calculation

Luminance, hue and saturation are important features of a color image, where the luminance, also called intensity, is used to indicate brightness, the hue is used to distinguish colors, and the saturation is the amount of white contained in the color. In general, normal digital operations, e.g., brightness/contrast adjustment, only change intensity and keep hue and saturation almost unchanged. To effectively describe color feature, we select color vector angle [21] to represent color image. This is because color vector angle is sensitive to hue and saturation differences but robust against intensity change. Comparing with the Euclidean distance in RGB color space, color vector angle is more effective in evaluating perceptual differences between two colors. Take Fig. 3 for example. $(\mathbf{C}_1, \mathbf{C}_2)$ is a pair of colors and $(\mathbf{C}_3, \mathbf{C}_4)$ is another pair of colors. Although they have perceptual difference, their Euclidean distances are still the same. On the contrary, the color vector angle of $(\mathbf{C}_1, \mathbf{C}_2)$ is 0.3486 radians, and that of $(\mathbf{C}_3, \mathbf{C}_4)$ is 0.6193 radians. Obviously, color vector angle can efficiently distinguish color differences in the RGB color space.

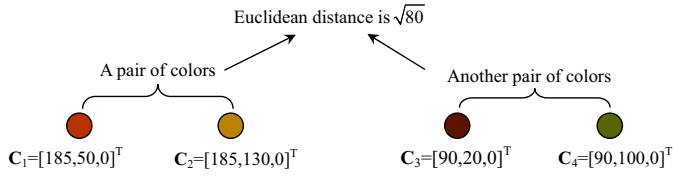


Fig. 3. Two color pairs having perceptual difference with the same Euclidean distance.

Let $\mathbf{P}_1 = [R_1, G_1, B_1]^{\mathrm{T}}$ and $\mathbf{P}_2 = [R_2, G_2, B_2]^{\mathrm{T}}$ be the vectors of two colors, where R_1 and R_2, G_1 and G_2, B_1 and B_2, are their red, green and blue components, respectively. Thus, their color vector angle θ can be calculated by

$$\theta = \arcsin\left(1 - \frac{(\mathbf{P}_1^{\mathrm{T}}\mathbf{P}_2)^2}{\mathbf{P}_1^{\mathrm{T}}\mathbf{P}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{P}_2}\right)^{1/2} \tag{3}$$

where $\arcsin(\cdot)$ is the operation of arcsin. Here, the sine of θ is used to reduce computational cost, which is defined as follows:

$$\sin\theta = \left(1 - \frac{(\mathbf{P}_1^{\mathrm{T}}\mathbf{P}_2)^2}{\mathbf{P}_1^{\mathrm{T}}\mathbf{P}_1\mathbf{P}_2^{\mathrm{T}}\mathbf{P}_2}\right)^{1/2} \tag{4}$$

Clearly, color vector angle calculation needs two colors. To compute color vector angle of each image pixel, a reference color $\mathbf{P}_{\text{ref}} = [R_{\text{ref}}, G_{\text{ref}}, B_{\text{ref}}]^{\mathrm{T}}$ is firstly generated, where R_{ref}, G_{ref} and B_{ref} are calculated by

$$R_{\text{ref}} = \frac{1}{MM}\sum_{i=1}^{M}\sum_{j=1}^{M} R_{i,j} \tag{5}$$

$$G_{\text{ref}} = \frac{1}{MM}\sum_{i=1}^{M}\sum_{j=1}^{M} G_{i,j} \tag{6}$$

$$B_{\text{ref}} = \frac{1}{MM}\sum_{i=1}^{M}\sum_{j=1}^{M} B_{i,j} \tag{7}$$

in which $R_{i,j}$, $G_{i,j}$ and $B_{i,j}$ are the red, green and blue components of the image pixel $\mathbf{P}_{i,j}$ in the ith row and the jth column ($1 \le i \le M$, $1 \le j \le M$). Thus, the color vector angle $\sin\theta_{i,j}$ between $\mathbf{P}_{i,j}$ and $\mathbf{P}_{\text{ref}}$ is calculated and a matrix $\mathbf{A}_{\text{color}}$ defined in Eq. (8) is available. Fig. 4 is an example of conversion from a color image to its color vector angles.

$$\mathbf{A}_{\text{color}} = \begin{bmatrix} \sin\theta_{1,1} & \sin\theta_{1,2} & \cdots & \sin\theta_{1,M} \\ \sin\theta_{2,1} & \sin\theta_{2,2} & \cdots & \sin\theta_{2,M} \\ \cdots & \cdots & \cdots & \cdots \\ \sin\theta_{M,1} & \sin\theta_{M,2} & \cdots & \sin\theta_{M,M} \end{bmatrix} \tag{8}$$

2.3. Edge detection

Generally, edge is a collection of points in a digital image at which the pixel brightness changes sharply or, more formally, has discontinuities. In fact, edge is an important visual feature for HVS to distinguish images and has been widely used in image processing, machine vision and computer vision. In the past, many useful edge detection methods have been reported, such as Sobel operator, Prewitt operator [22], LoG operator [23] and Canny operator [24]. Here, Canny operator is selected. This is because,

(a) Color image (b) Color vector angles

Fig. 4. Conversion from the color image to color vector angles.

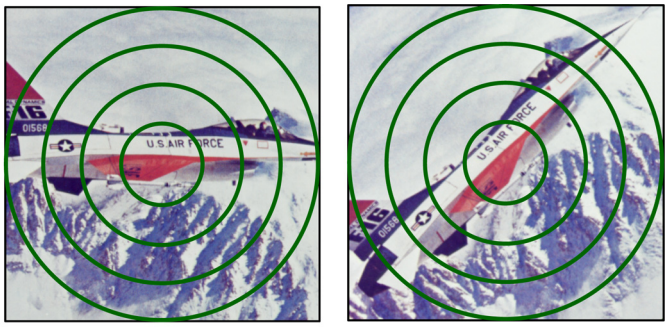
(a) Result with luminance (b) Result with color vector angle

Fig. 5. Edge detection results with Canny operator.

comparing with other operators, it can reach a desirable balance between detection performance and computational cost. Canny operator is a five-step method as follows. (1) Create a smooth image for reducing noise influence on detection result with a Gaussian filter. (2) Calculate the intensity gradients of the smooth image. (3) Exploit non-maximum suppression to get rid of spurious response to edge detection. (4) Determine potential edges with double thresholds. (5) Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges. More details of Canny operator can be referred to [24].

To find image edge, we convert the normalized image into YCbCr color space, take the luminance component for representation and then apply the Canny operator to the luminance component. Here we choose luminance component instead of color vector angle for edge detection. This is because edge detection with luminance is more accurate than the detection with color vector angle. For space limitation, a typical comparison of edge detection is exemplified in Fig. 5, where (a) is the result with luminance and (b) is the result with color vector angle. It is clear that, in Fig. 5(b), there is almost no edge in the marked rectangle regions. But these real edges are found in Fig. 5(a). Let $\mathbf{E}=(E_{i,j})_{M\times M}$ be the result of edge detection with Canny operator, where $E_{i,j}$ is defined as follows:

$$E_{i,j}=\begin{cases}1, & P_{i,j} \text{ is an edge point,}\\ 0, & P_{i,j} \text{ is not an edge point.}\end{cases} \tag{9}$$



(a) Central part of Airplane (b) Rotated by 45°

Fig. 6. Examples of four concentric circles in an image and its similar version.

2.4. Statistical feature extraction

In general, image rotation often takes image center as origin of coordinates and those pixels on the concentric circles (whose circle center is image center) are kept unchanged after rotation. As shown in Fig. 6, (a) is the central part of the standard benchmark image Airplane, and (b) is its similar version obtained by cropping the rotated Airplane. It is clear that image pixels on the concentric circles of Fig. 6(a) are the same with those of Fig. 6(b). Therefore, we can make our hash robust against image rotation by extracting statistical features using those pixels on the concentric circles. Specifically, we divide the color vector angle matrix $\mathbf{A}_{\text{color}}$ into a set of concentric circles, calculate variances of color vector angles of those edge pixels on the concentric circles, and then quantize these variances to make a short hash. The detailed calculation is as follows.

Let K be the number of concentric circles and $\mathbf{E}^{(k)}$ be the set of color vector angles of those edge pixels on the kth concentric circle ($k=1,2,\ldots,K$). Suppose that r_k is the kth radius ($k=1,2,\ldots,K$) labeled from small value to big value (r_1 and r_K are the innermost and outermost radii, respectively). Thus, it can be calculated by the below formula:

$$r_k = kd \tag{10}$$

where d is defined as follows:

$$d = \lfloor M/2K \rfloor \tag{11}$$

where $\lfloor\cdot\rfloor$ means downward rounding. Next, calculate the distance $d_{i,j}$ from the pixel $\mathbf{P}_{i,j}$ ($1\le i\le M$, $1\le j\le M$) to the image center. Thus, edge points on the circle can be determined by the relation between $d_{i,j}$ and the circle radius. Let (x_c, y_c) be the coordinates of the image center. Thus, $x_c=M/2+0.5$ and $y_c=M/2+0.5$ if M is an even number. Otherwise, $x_c=(M+1)/2$ and $y_c=(M+1)/2$. Therefore, $d_{i,j}$ can be calculated by

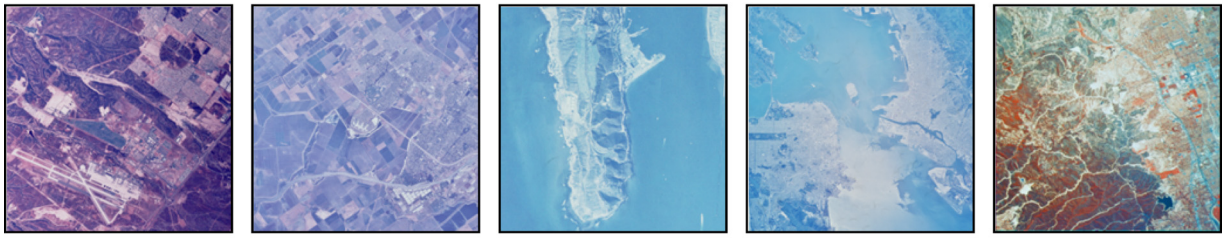
$$d_{i,j}=\sqrt{(i-x_c)^2+(j-y_c)^2} \tag{12}$$

Consequently, $\mathbf{E}^{(k)}$ can be determined by the following equation:

$$\mathbf{E}^{(k)}=\{\mathbf{A}_{\text{color}}(i,j)\,\big|\,|d_{i,j}-r_k|\le\Delta d \quad\text{and}\quad E_{i,j}=1\} \quad (k=1,2,\ldots,K) \tag{13}$$

where Δd is a pre-defined threshold for error control. The use of Δd is based on the consideration that the coordinates of image pixel are discrete and there may be few pixels falling precisely on the circle, i.e., $d_{i,j}=r_k$.

(a) Standard benchmark images



(b) Sample images of the volume '*Aerials*'

Fig. 7. Sample images of our test database for robustness validation.

Having obtained $\mathbf{E}^{(k)}$, we calculate its variance v_k as follows:

$$v_k = \frac{1}{N_k - 1}\sum_{i=1}^{N_k}\left(E^{(k)}(i) - m_k\right)^2 \tag{14}$$

where N_k is the element number of $\mathbf{E}^{(k)}$, $E^{(k)}(i)$ is the ith element of $\mathbf{E}^{(k)}$ $(1 \le i \le N_k)$, and m_k is the mean of $\mathbf{E}^{(k)}$ which can be calculated by the following formula:

$$m_k = \frac{1}{N_k}\sum_{i=1}^{N_k} E^{(k)}(i) \tag{15}$$

To reduce the cost of hash storage, the variance v_k is quantized as follows:

$$h(k) = [v_k \times 10{,}000 + 0.5] \tag{16}$$

where $[\cdot]$ is the rounding operation and $h(k)$ is the kth hash element. After rounding, each hash element only requires 10 bits for storage. This will be validated in Section 3.2. Finally, our image hash $\mathbf{h}$ is available as follows.

$$\mathbf{h} = [h(1), h(2), \ldots, h(K)] \tag{17}$$

Clearly, our hash length is K integers, equivalent to $10K$ bits. In Section 3.3, we will discuss effect of K value on hash performances.

2.5. Similarity evaluation

To measure similarity between two image hashes, the well-known correlation coefficient is taken as similarity metric. Let $\mathbf{h}_1$ and $\mathbf{h}_2$ be two hashes. Thus, the correlation coefficient can be defined as follows.

$$S(\mathbf{h}_1, \mathbf{h}_2) = \frac{\sum_{k=1}^{K}[h_1(k) - \mu_1][h_2(k) - \mu_2]}{\sqrt{\sum_{k=1}^{K}[h_1(k) - \mu_1]^2 \times \sum_{k=1}^{K}[h_2(k) - \mu_2]^2}} \tag{18}$$

where μ_1 and μ_2 are the means of $\mathbf{h}_1$ and $\mathbf{h}_2$, $h_1(k)$ and $h_2(k)$ are the kth elements of $\mathbf{h}_1$ and $\mathbf{h}_2$, respectively. The range of S is $[-1, 1]$. The bigger the S value is, the more similar the corresponding images of the input hashes are. If S is bigger than a pre-defined threshold T, the corresponding images of the input hashes are judged as visually similar images. Otherwise, they are different images.

3. Experimental results

To validate efficiency of our hashing, perceptual robustness and discriminative capability are tested in Sections 3.1 and 3.2, respectively. In the experiments, all images are resized to 512×512 and blurred by a 3×3 Gaussian low-pass mask with a unit standard deviation, the number of concentric circles is 40, and the threshold for error control is 3, i.e., $M = 512$, $K = 40$, and $\Delta d = 3$. Therefore, our hash length is 40 integers. Effect of the circle number on our hash performances is discussed in Section 3.3.

3.1. Perceptual robustness

To validate our perceptual robustness, we collect a test database with 42 color images, including five standard benchmark images sized 512×512 (i.e., Airplane, Baboon, House, Peppers, and Lena), and 37 images (all color images) in the second volume (i.e., '*Aerials*') of the USC-SIPI Image Database [25]. Fig. 7 presents typical images of our test database. To generate visual similar versions of these color images, we exploit Photoshop, MATLAB and StirMark 4.0 [26] to conduct robustness attack. The used operations include brightness adjustment, contrast adjustment, gamma correction, 3×3 Gaussian low-pass filtering, JPEG compression, watermark embedding, scaling, and rotation and cropping. For each operation, different parameters are used. Detailed settings of these operations are presented in Table 1. Since image sizes of the rotated images are significantly expanded and some image regions are padded with black or white pixels, we only take the central parts sized 361×361 of the original images and the rotated images for hash generation. After robustness attack, each original test image has 60 similar images. And then, there are $42 \times 60 = 2520$ pairs of visually identical images in total. Consequently, the number of the used images is $2520 + 42 = 2562$. We extract image hashes of the original images and their similar versions, and calculate their similarities by correlation coefficient S. Statistical results of these S values under different operations are presented in Table 2. Clearly, the mean S values of all operations are bigger than 0.95, except JPEG compression and rotation and cropping, whose values are 0.9399 and 0.9186, respectively. Moreover, the standard deviations of all operations are small and the maximum standard deviation is only 0.0845. Therefore, we can choose the threshold $T = 0.95$ to resist most of the above operations.

Table 1
Digital operations and their used parameter values.

Tool	Operation	Description	Parameter value	Number of images
Photoshop	Brightness adjustment	Photoshop's scale	±10, ±20	4
Photoshop	Contrast adjustment	Photoshop's scale	±10, ±20	4
MATLAB	Gamma correction	γ	0.75, 0.9, 1.1, 1.25	4
MATLAB	3×3 Gaussian low-pass filtering	Standard deviation	0.3, 0.4, . . ., 1.0	8
StirMark	JPEG compression	Quality factor	30, 40, . . ., 100	8
StirMark	Watermark embedding	Strength	10, 20, . . ., 100	10
StirMark	Scaling	Ratio	0.5, 0.75, 0.9, 1.1, 1.5, 2.0	6
StirMark	Rotation and cropping	Angle in degree	±1, ±2, ±5, ±10, ±15, ±30, ±45,±90	16
Total				60

Table 2
Statistics of S values under different operations.

Operation	Max.	Min.	Mean	Standard deviation
Brightness adjustment	1.0000	0.8005	0.9752	0.0360
Contrast adjustment	0.9990	0.7699	0.9700	0.0384
Gamma correction	0.9990	0.7876	0.9653	0.0408
3×3 Gaussian low-pass filtering	1.0000	0.7525	0.9741	0.0459
JPEG compression	0.9980	0.4465	0.9399	0.0824
Watermark embedding	0.9995	0.7538	0.9705	0.0427
Scaling	0.9974	0.7239	0.9575	0.0541
Rotation and cropping	0.9988	0.3023	0.9186	0.0845

3.2. Discriminative capability

To test the discriminative capability, we collect 200 different color images to form a database, where 67 images are downed from the Internet, 33 images are captured by digital cameras and 100 images are taken from the Ground Truth Database [27]. The image sizes range from 256×256 to 2048×1536. Fig. 8 presents typical images in the test database. For each image, we compare it with other 199 images. Therefore, there are $200 \times 199/2 = 19{,}000$ pairs of different images. We calculate hash similarity of each pair of images and then obtain the distribution of similarity results as shown in Fig. 9, where the x-axis is the correlation coefficient and the y-axis represents its frequency. It is found that, the minimum and maximum values of these results are −0.86918 and 0.97059, respectively. And their mean and standard deviation are 0.17532 and 0.35747, respectively. Consequently, if $T = 0.95$, there are only 0.04% different images wrongly considered as similar images.

To determine the required bits for hash storage, we exploit all hash elements of the above 200 image hashes (i.e., $200 \times 40 = 8000$ hash elements) to calculate value distribution of hash elements. The result is illustrated in Fig. 10, where the x-axis is the value of hash element and the y-axis represents its frequency. From Fig. 10, we observe that element value ranges from 0 to 900, the maximum value is 890, and most values are smaller than 400. Therefore, for

Fig. 8. Typical images in the test database for discriminative experiment.

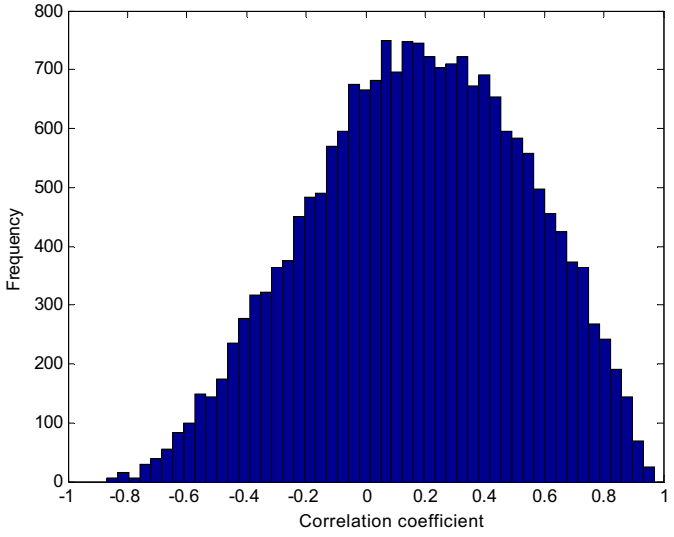


Fig. 9. Correlation coefficient distribution between hashes of different images.

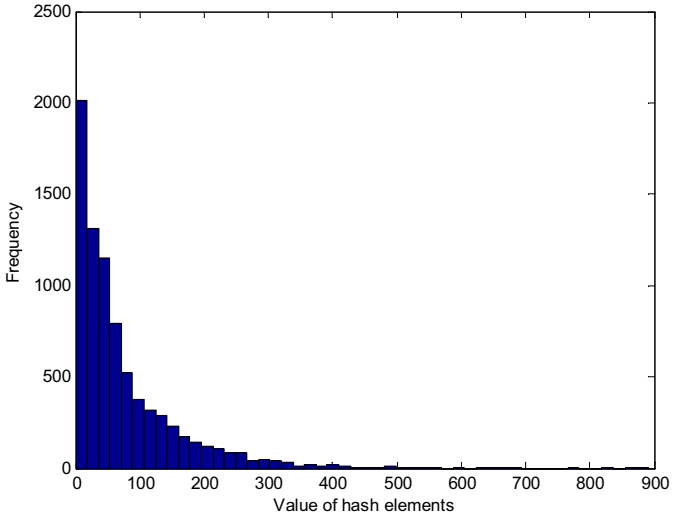


Fig. 10. Value distribution of hash elements.

storing a hash element, only 10 bits are needed which can represent those numbers ranging from 0 to $2^{10} - 1 = 1023$. Thus, our hash length is $10K$ bits. Here, $K = 40$ and thus our hash length is 400 bits. This length is short enough. As a reference, the lengths of the non-uniform sampling based hashing [2], the SVD–SVD hashing [12], the ZM-based hashing [19] and the moment invariants based hashing [20] are 444 bits, 1600 decimals, 560 bits and 960 bits, respectively.

3.3. Effect of circle number on hash performances

To view effect of the circle number on hash performances, we only vary the circle number K and keep other parameters used in the above sections unchanged. To analyze classification performances between perceptual robustness and discriminative capability under different K values, we choose the well-known receiver operating characteristics (ROC) graph [28] as the visualization tool, where true positive rate (P_{TPR}) and false positive rate (P_{FPR}) are two important indices defined as follows.

$$P_{\mathrm{TPR}} = \frac{n_{\mathrm{TPR}}}{N_{\mathrm{TPR}}} \tag{19}$$

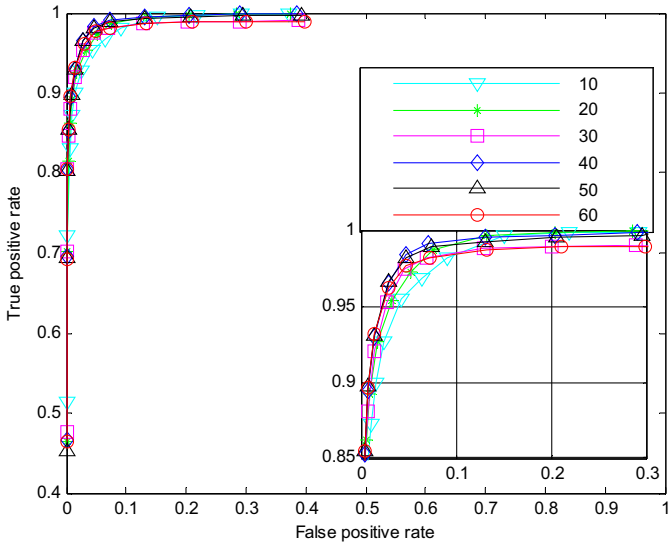


Fig. 11. ROC curve comparisons among different K.

$$P_{\mathrm{FPR}} = \frac{n_{\mathrm{FPR}}}{N_{\mathrm{FPR}}} \tag{20}$$

where n_{TPR} is the number of the pairs of visually identical images considered as similar images, N_{TPR} is the total pairs of visually identical images, n_{FPR} is the number of the pairs of different images considered as similar images, and N_{FPR} is the total pairs of different images. Actually, P_{TPR} and P_{FPR} indicate the robustness and the discriminative capability, respectively. Generally, in the ROC graph, the x-axis is P_{FPR}, the y-axis is P_{TPR}, and the curve close to the top-left corner has good classification performances.

In the experiments, the used K values are: 10, 20, 30, 40, 50 and 60. We calculate the ROC curves under different K values, and obtain the results as shown in Fig. 11. It is observed that our all ROC curves are close to the top-left corner, illustrating good classification performances. To view differences, these ROC curves near the top-left corner are enlarged as shown in the right-bottom part of Fig. 11. We find that the ROC curve of $K = 40$ is a little more closer to the top-left corner than those of other K values. In other words, classification performances of $K = 40$ is slightly better than those of other values. This can be understood as follows. When K is small, e.g., $K = 10$ and $K = 20$, there are few features in the hash and thus discrimination is hurt. As K increases, classification performances can be gradually improved. However, if K is too big, e.g., $K = 50$ and $K = 60$, the whole hash performances slightly decrease. The reasons are as follows. During the extraction of edge pixels on the circle, a threshold Δd is used for error control. Here $\Delta d = 3$. This means that those edge pixels near the circle (distance is not bigger than $\Delta d = 3$) are all used in feature extraction. In other words, the selected circle is a thin region with $\Delta d \times 2$ width. If K is too big, the selected regions of the adjacent circles will overlap, leading to redundant features in the hash. Clearly, the redundant features cannot improve discrimination. On the contrary, they are disturbed by normal digital processing and thus slightly decrease perceptual robustness.

In fact, in the experiment, the normalized image is 512×512 and thus the radius of inscribed circle is 256. When $K = 40$, $K \times \Delta d \times 2 = 40 \times 3 \times 2 = 240$ which is close to the radius of the inscribed circle. In other words, the regions of the selected circles can almost cover the inscribed circle area of the whole image, meaning that all image features are almost included in the hash. Therefore, when $2K\Delta d$ is close to the radius of the inscribed circle of the image, our hashing can reach a good classification performance. Moreover, as K increases, the hash length also increases. This will affect the time of hash generation. We record the total consumed

Table 3
Hash length and time comparisons under different K values (Time unit: seconds).

K	Hash length	Total time	Average time
10	10 integers (100 bits)	78.48	0.39
20	20 integers (200 bits)	83.13	0.42
30	30 integers (300 bits)	93.98	0.47
40	40 integers (400 bits)	97.63	0.49
50	50 integers (500 bits)	122.23	0.61
60	60 integers (600 bits)	136.93	0.68

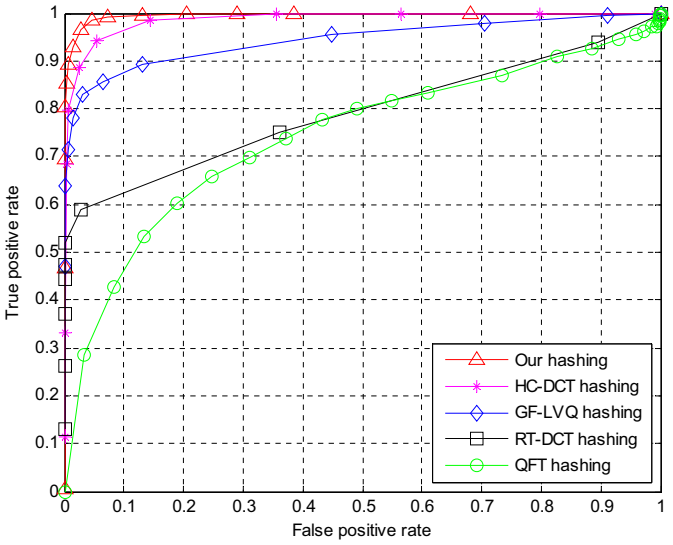


Fig. 12. ROC curve comparisons among different hashing algorithms.

time of generating the hashes of 200 images in the discrimination test under different K values and calculate the average time of generating a hash. Our algorithm is implemented with MATLAB 2012 and run on a personal computer with 3.20 GHz Intel Pentium Dual Core E5800 CPU and 2 GB RAM. Table 3 summarizes hash length and time comparisons under different K values. It is observed that when K increases, hash length becomes long and average time slightly increases.

4. Performance comparisons

To show advantages, we compare our hashing with some state-of-the-art algorithms: the RT-DCT hashing [11], the GF-LVQ hashing [15], the HC-DCT hashing [16], and the QFT hashing [18]. To make fair comparisons, the same images used in Section 3 are also taken to validate their classification performances. The ROC graph is used again for visualizing performance comparisons. For each algorithm, we choose some thresholds, calculate their corresponding P_{TPR} and P_{FPR}, and thus obtain its ROC curve. Fig. 12 presents the ROC curve comparisons among different hashing algorithms. Note that, in the ROC graph, the curve close to the top-left corner has better classification performances than that far away from the top-left corner. Clearly, the ROC curve of our hashing is closer to the top-left corner than those of other algorithms. Therefore, we can intuitively conclude that our hashing outperforms the compared algorithms in classification performances.

To make theoretical analysis of these results, we take the AUC (Area under the ROC curve) [28] as the metric. Note that the AUC is a good measure for evaluating classification performance. The AUC value is between 0 and 1. The bigger the AUC, the better the classification performance is. We calculate the AUC of every compared algorithm and obtain the results as follows. The AUCs of our hashing, the HC-DCT hashing, the GF-LVQ hashing, the RT-DCT hashing and QFT hashing are 0.9955, 0.9786, 0.9434, 0.7921 and 0.7424, respectively. Obviously, the AUC of our hashing is bigger than those of other compared algorithms. This means that our hashing is better than other algorithms in classification between robustness and discriminative capability.

Table 4
Time comparisons among different algorithms.

Algorithm	Average time (s)
Our hashing	0.49
HC-DCT hashing	0.14
GF-LVQ hashing	0.83
RT-DCT hashing	3.68
QFT hashing	0.51

Moreover, we compare the run time of the assessed algorithms. To do so, we record the consumed time of generating 200 different image hashes in the respective discrimination test, and calculate the average time of producing an image hash. Table 4 lists the average time of each algorithm. It is observed that, our hashing is slower than the HC-DCT hashing, but faster than all other algorithms. Among these algorithms, the HC-DCT hashing has the fastest speed and the RT-DCT hashing has the slowest speed. This is because the histogram extraction and compression in the HC-DCT hashing is simple, and the RT used in the RT-DCT hashing has a large computational cost.

5. Conclusions

In this work, we have proposed a robust image hashing based on color vector angle and Canny operator. Our image features are extracted from color vector angles of edge pixels on the circles. Since edge pixels on the circles are kept unchanged after rotation, our extracted features are invariant to image rotation and therefore make our hashing good rotation robustness. As color vector angle calculation fully exploits all RGB color components, our hashing reaches desirable discriminative capability. Experimental results have shown that our hashing is robust against normal content-preserving manipulations, such as image rotation with arbitrary angle, brightness adjustment, contrast adjustment, JPEG compression, gamma correction, 3×3 Gaussian low-pass filtering, watermark embedding and image scaling. ROC curve comparisons with some state-of-the-art algorithms have indicated that our hashing outperforms the compared algorithms in classification performances between robustness and discriminative capability.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (61300109, 61363034,61562007), the Guangxi Natural Science Foundation (2015GXNSFDA139040), Guangxi "Bagui Scholar" Teams for Innovation and Research, the Scientific and Technological Research Projects in Guangxi Higher Education Institutions (YB2014048), the Project of the Guangxi Key Lab of Multi-source Information Mining & Security (15-A-02-02, 14-A-02-02, 13-A-03-01), the Scientific Research and Technological Development Program of Guilin (20140103-17), and Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing.

References

[1] Tang Z, Zhang X, Dai X, Yang J, Wu T. Robust image hash function using local color features. AEÜ – Int J Electron Commun 2013;67(8):717–22.
[2] Qin C, Chang CC, Tsou PL. Robust image hashing using non-uniform sampling in discrete Fourier domain. Digital Signal Process 2013;23(2):578–85.
[3] Tang Z, Zhang X, Zhang S. Robust perceptual image hashing based on ring partition and NMF. IEEE Trans Knowl Data Eng 2014;26(3):711–24.

[4] Schneider M, Chang S-F. A robust content based digital signature for image authentication. In: Proceedings of the IEEE international conference on image processing (ICIP 1996), vol. 3. 1996. p. 227–30.
[5] Venkatesan R, Koon S-M, Jakubowski MH, Moulin P. Robust image hashing. In: Proceedings of the IEEE international conference on image processing. 2000. p. 664–6.
[6] Fridrich J, Goljan M. Robust hash functions for digital watermarking. In: Proceedings of the IEEE international conference on information technology: coding and computing. 2000. p. 178–83.
[7] Swaminathan A, Mao Y, Wu M. Robust and secure image hashing. IEEE Trans Inform Forensics Secur 2006;1(2):215–30.
[8] Monga V, Evans BL. Perceptual image hashing via feature points: performance evaluation and tradeoffs. IEEE Trans Image Process 2006;15(11):3452–65.
[9] Lefèbvre F, Macq B, Legat J-D. RASH: Radon soft hash algorithm. In: Proceedings of European signal processing conference. 2002. p. 299–302.
[10] Roover CD, Vleeschouwer CD, Lefèbvre F, Macq B. Robust video hashing based on radial projections of key frames. IEEE Trans Signal Process 2005;53(10):4020–36.
[11] Ou Y, Rhee KH. A key-dependent secure image hashing scheme by using Radon transform. In: Proceedings of the IEEE international symposium on intelligent signal processing and communication systems. 2009. p. 595–8.
[12] Kozat SS, Venkatesan R, Mihçak MK. Robust perceptual image hashing via matrix invariants. In: Proceedings of the IEEE international conference on image processing. 2004. p. 3443–6.
[13] Monga V, Mihcak MK. Robust and secure image hashing via non-negative matrix factorizations. IEEE Trans Inform Forensics Secur 2007;2(3):376–90.
[14] Tang Z, Wang S, Zhang X, Wei W, Zhao Y. Lexicographical framework for image hashing with implementation based on DCT and NMF. Multim Tools Appl 2011;52(2–3):325–45.
[15] Li Y, Lu Z, Zhu C, Niu X. Robust image hashing based on random Gabor filtering and dithered lattice vector quantization. IEEE Trans Image Process 2012;21(4):1963–80.
[16] Tang Z, Dai Y, Zhang X, Zhang S. Perceptual image hashing with histogram of color vector angles. In: The 8th international conference on active media technology (AMT 2012). 2012. p. 237–46.
[17] Tang Z, Zhang X, Dai Y, Lan W. Perceptual image hashing using local entropies and DWT. Imaging Sci J 2013;61(2):241–51.
[18] Laradji IH, Ghouti L, Khiari E-H. Perceptual hashing of color images using hypercomplex representations. In: Proceedings of the IEEE international conference on image processing (ICIP 2013). 2013. p. 4402–6.
[19] Zhao Y, Wang S, Zhang X, Yao H. Robust hashing for image authentication using Zernike moments and local features. IEEE Trans Inform Forensics Secur 2013;8(1):55–63.
[20] Tang Z, Yu J, Zhang X, Zhang S. Discovery of tampered image with robust hashing. In: The 10th international conference on advanced data mining and applications (ADMA 2014). 2014. p. 112–22.
[21] Kim NW, Kim TY, Choi JS. Edge-based spatial descriptor for content-based image retrieval. Lect Notes Comput Sci 2005;3568:454–64.
[22] Wang D, Zhou S. Color image recognition method based on the Prewitt operator. In: Proceedings of 2008 international conference on computer science and software engineering. 2008. p. 170–3.
[23] Zhang J, Change W, Wu L. Edge detection based on general grey correlation and LoG operator. In: Proceedings of 2010 international conference on artificial intelligence and computational intelligence. 2010. p. 480–3.
[24] Canny J. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 1986;8(6):679–98.
[25] USC-SIPI Image Database. Available from: http://sipi.usc.edu/database/.
[26] Petitcolas FAP. Watermarking schemes evaluation. IEEE Signal Process Mag 2000;17(5):58–64.
[27] Ground Truth Database, http://www.cs.washington.edu/research/imagedatabase/groundtruth/, 2008.
[28] Fawcett T. An introduction to ROC analysis. Pattern Recog Lett 2006;27(8):861–74.

Zhenjun Tang received the B.S. and M.Eng. degrees from Guangxi Normal University, Guilin, P.R. China, in 2003 and 2006, respectively, and the PhD degree from Shanghai University, Shanghai, P.R. China, in 2010. He is now a professor with the Department of Computer Science, Guangxi Normal University. His research interests include image processing and multimedia security. He has contributed more than 30 papers in international journals such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Information Forensics and Security*, *IET Image Processing*, *Signal Processing*, *Digital Signal Processing*, *Applied Mathematics and Computation*, *Fundamenta Informaticae*, *Multimedia Tools and Applications*, *Imaging Science Journal*, *Applied Mathematics & Information Sciences*, *AEÜ-International Journal of Electronics and Communications*, and *Optik-International Journal for Light and Electron Optics*. He holds five China patents. He is a reviewer of many reputable journals such as *IEEE Transactions on Image Processing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Circuits and Systems for Video Technology*, *Signal Processing*, *Digital Signal Processing*, *IET Image Processing*, *IET Computer Vision*, *Journal of Visual Communication and Image Representation*, *Neurocomputing*, *Multimedia Tools and Applications*, and *Imaging Science Journal*.

Liyan Huang received the B.S. and M.Eng. degrees from Guangxi Normal University, Guilin, P.R. China, in 2011 and 2014, respectively. Her research interests include image processing and multimedia security.

Xianquan Zhang received the M.Eng. degree from Chongqing University, Chongqing, P.R. China, in 1996. He is currently a Professor with the Department of Computer Science, Guangxi Normal University. His research interests include image processing and computer graphics. He has contributed more than 100 papers.

Huan Lao received the B.S. degree from Guangxi Normal University, Guilin, China, in 2014. She is now pursuing the M.Eng. degree in computer science and technology. Her research interests include image processing and multimedia security.