Multimed Tools Appl
DOI 10.1007/s11042-009-0437-y

Lexicographical framework for image hashing with implementation based on DCT and NMF

Zhenjun Tang · Shuozhong Wang · Xinpeng Zhang · Weimin Wei · Yan Zhao

Abstract Image hash is a content-based compact representation of an image for applications such as image copy detection, digital watermarking, and image authentication. This paper proposes a lexicographical-structured framework to generate image hashes. The system consists of two parts: dictionary construction and maintenance, and hash generation. The dictionary is a large collection of feature vectors called words, representing characteristics of various image blocks. It is composed of a number of sub-dictionaries, and each sub-dictionary contains many features, the number of which grows as the number of training images increase. The dictionary is used to provide basic building blocks, namely, the words, to form the hash. In the hash generation, blocks of the input image are represented by features associated to the sub-dictionaries. This is achieved by using a similarity metric to find the most similar feature among the selective features of each sub-dictionary. The corresponding features are combined to produce an intermediate hash. The final hash is obtained by encoding the intermediate hash. Under the proposed framework, we have implemented a hashing scheme using discrete cosine transform (DCT) and non-negative matrix factorization (NMF). Experimental results show that the proposed scheme is resistant to normal content-preserving manipulations, and has a very low collision probability.

Keywords Image hashing · Lexicographic framework · Image retrieval · Digital rights management (DRM) · Non-negative matrix factorization (NMF)

Z. Tang · S. Wang (✉) · X. Zhang · W. Wei · Y. Zhao
School of Communication and Information Engineering, Shanghai University, 149 Yanchang Road, Shanghai 200072, People's Republic of China
e-mail: shuowang@shu.edu.cn

Z. Tang
e-mail: tangzj230@163.com

X. Zhang
e-mail: xzhang@shu.edu.cn

W. Wei
e-mail: wwm@shu.edu.cn

Y. Zhao
e-mail: yanzhao@shu.edu.cn

Published online: 13 January 2010

1 Introduction

Digital images are easily manipulated by using powerful image processing tools, leading to various problems such as copyright infringement and hostile tampering to the image contents. Perceptual hashing is a useful means for digital rights management (DRM). An image hash is a short string extracted from the image to represent the contents, which can be used in copy detection, digital watermarking, indexing, authentication, tamper detection, and content-based image retrieval (CBIR). In general, an image hash should be perceptually robust, unique, and secure. Perceptual robustness means that visually identical images should have almost the same hash with high probability even if their digital representations are not identical. Uniqueness is also called anti-collision capability, which implies that probability of two different images having an identical hash value, or very close hash values, should tend to zero. Security ensures that prediction of hash is impossible without the knowledge of the keys. Classical cryptographic hash functions such as SHA-1 and MD5 are not directly applicable to image hashing as they are too sensitive to slight changes of the input data therefore not robust to normal content-preserving image processing.

Various image hashing schemes and performance evaluation metrics have been proposed. Feature extraction is an important step in image hashing. The methods of feature extraction may be based on various transformations such as discrete wavelet transform (DWT) [8, 16], discrete cosine transform (DCT) [1, 6, 11], Radon transform [4] and Fourier-Mellin transform [12], and on standard-rank reduction techniques such as singular value decomposition (SVD) [3] and non-negative matrix factorization (NMF) [9, 13].

An early method proposed by Venkatesan et al. [16] uses quantized means or variances, depending on the sub-band, of wavelet coefficients as image features to construct the hash. The drawback of this method is that the known wavelet statistics can be used to generate a visually different image with the same hash. Monga and Evans [8] extract the corner-like stimuli and high curvature points to represent the visual appearance of the image. This is achieved by calculating a wavelet transform based on an end-stopped wavelet obtained by applying the first derivative of Gaussian operator to the Morlet wavelet. This method is robust against JPEG compression, scaling and small-angle rotation.

Fridrich and Goljan [1] observe that the magnitude of a low-frequency DCT coefficient cannot be changed easily without causing visible changes to the image. They construct a robust hash using the block DCT coefficients and a set of DC-free random smooth patterns. This method withstands a set of common processing operations, but is not sensitive enough to small-area tampering. In [6], Lin and Chang propose a technique for image authentication based on the observation that relationships between DCT coefficients at the same position in separate blocks is preserved before and after JPEG compression. The method can also handle distortions introduced by various acceptable manipulations such as integer rounding, image filtering, and image enhancement. In another work [11], a RAdial Variance (RAV) vector is extracted using the radial projections of image pixels. The low-frequency DCT coefficients of the RAV vector are quantized to form the image hash. This scheme is resilient to image rotation and re-scaling, but its collision risk is not low enough.

Lefebvre et al. [4] first use the Radon transform to obtain image features invariant against rotation and scaling. It is also robust against basic image processing operations and StirMark attacks. Swaminathan et al. [12] use the rotation-invariant Fourier-Mellin coefficients to generate image hashes. This hash function is resilient to several content-preserving modifications such as moderate geometric transforms and filtering. Since the hash is only dependent on the magnitudes of Fourier coefficients but independent of the

phases, attackers can easily create a similar image corresponding to a totally different hash or a synthesized image corresponding to the same hash [17].

In [3], Kozat et al. view attacks on the image as a sequence of linear operators, and propose a hashing algorithm using low-rank matrix approximations obtained by SVD. It is robust against geometric attacks at the expense of significantly increasing misclassification. Because of the non-negativity of images, NMF is distinguished from traditional matrix approximation methods such as SVD. Monga and Mihcak [9] first use NMF to derive image hashing. They apply NMF to some sub-images, use factorization factors to construct a secondary image, and obtain a low-rank matrix approximation of the secondary image with NMF again. The matrix entries are concatenated to form an NMF-NMF vector. To get a short vector, the inner product between the NMF-NMF vectors and a set of weight vectors which have i.i.d. Gaussian components of zero mean and unit variance is calculated. The NMF-NMF-SQ hashing has been shown to have good performances. In another study [13], the invariant relation existing in the coefficient matrix of NMF is used for constructing robust image hashes. This method is also robust against StirMark attacks such as JPEG compression, additive noise, and watermarking embedding. It shows better capabilities than NMF-NMF-SQ hashing in anti-collision and tamper detection, but is fragile to image rotation.

Most of the above methods can be divided into three steps: preprocessing, feature extraction, and post-processing. In the first step, the input image undergoes a set of operations such as filtering and resizing to produce a normalized image. The most important step for hash generation is feature extraction, which will directly affect the hash performance. Common features include DWT or DCT coefficients, Radon feature, Fourier-Mellin coefficients, and factorization factors. Feature compression or quantization is often performed in the last step to produce a hash of a reasonable length. To make a secure hash, randomization in the feature extraction is generally necessary, often achieved by randomizing the block selection or using random encryption matrices. However, [7] shows that randomized block selection and random encryption matrices can be accurately estimated if the secret key is reused several times. Thus, there is a risk that the attacker may exploit the security loophole to forge the image hash.

In this paper, we introduce a new image hashing framework, which constructs a hash of the entire image using basic elements representing small blocks of the image. This is like composing an article with words chosen from a dictionary. The system consists of two parts: dictionary construction and maintenance, and hash generation. The dictionary is a large collection of feature vectors called words, representing characteristics of various image blocks. The extracted hash is dependent on the input image and secret keys, and also closely related to the dictionary, ensuring robustness and security. Section 2 describes the lexicographical hashing framework. Section 3 presents an implementation using DCT and NMF. Sections 4 and 5 give a performance analysis of the implemented scheme, and an application to image retrieval, respectively. Section 6 concludes the paper.

2 Proposed hashing framework

2.1 Basic thoughts and the system diagram

An image can be viewed as a collection of small building blocks, which in the simplest form are non-overlapping rectangles. Suppose we have a database containing a very large

number of image blocks of the same size taken from various images. If the source images cover a sufficiently wide variety of natural images, we can use a set of blocks in the database to approximate any given image.

For image hashing, we extract features of the blocks, and form short binary sequences called *words* to represent the features. The entire collection of the words is called a *dictionary*. This way, we can find words from the dictionary that best represent the block contents, and construct the image hash just like composing an article using words available in the dictionary. A proper coding scheme is needed to make the final hash reasonably short, and satisfy the fundamental requirements of robustness, uniqueness, and security.

It is clear that the larger the number of blocks in the database, the better the quality of the approximate images. Therefore, it is desirable to make the dictionary expandable as adding new words originating from more source images can improve quality of the dictionary hence performance of the image hash. In the meantime, the system should be backward compatible, i.e., hashes generated before the dictionary expansion should remain valid. In other words, using any updated version of the dictionary, the same hash should be obtained for the same original image if the control parameters of hash function are unchanged.

Figure 1 is a block diagram of the proposed two-part lexicographical hashing system. In the sub-system of dictionary construction and maintenance, words are derived from the feature vectors of image blocks extracted from the source image database. The dictionary is expanded by supplying more source images. A secret key is used to control the process for security. In the hash generation sub-system, the dictionary is looked up to give words corresponding to the image block features. The words are then sent to an encoder to produce the hash. Keys in the diagram are indicative, which may be designed differently in specific implementations.

2.2 Dictionary construction

A dictionary should contain a very large number of words. In the described system, a *word* reflects features of an image block such as structure, color and texture. A simple method for constructing a dictionary is to concatenate feature vectors. This method has weakness though. As the number of features increases, more time is needed to find a

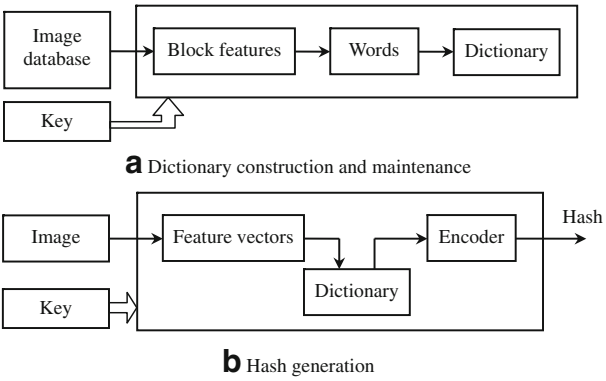


Fig. 1 Block diagram of the lexicographical image hashing system. **a** Dictionary construction and maintenance, **b** Hash generation

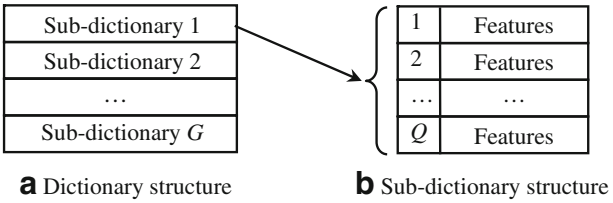


Fig. 2 Structure of the dictionary. **a** Dictionary structure, , **b** Sub-dictionary structure

matched feature. For example, if there are 100,000 words in the dictionary, we need to carry out 6,400,000 computations to generate an image hash if the image has 64 blocks. To bring the computation cost to a reasonable level, we divide the dictionary into a set of sub-dictionaries, as shown in Fig. 2(a). The dictionary contains G sub-dictionaries where G is a fixed system parameter. In each sub-dictionary, there are Q feature vectors where Q equals the number of images in the database for dictionary generation. Feature vectors in the sub-dictionary are stored in a certain order controlled by a key. The order should be invariant after dictionary expansion. For simplicity in the present work, the feature vectors are arranged in a natural order as shown in Fig. 2(b). Storing feature vectors in a given order can ensure that, using the same parameters, the original image hash can be correctly extracted after the dictionary is expanded.

To build the dictionary, each training source-image is made to contribute one feature vector (word) to a sub-dictionary, and in total, G feature vectors to the dictionary. Given an image, a series of blocks are randomly selected to form a block set. The goal is to form G feature-vectors from the set, and add them to the sub-dictionaries. Figure 3 shows the updating procedure. For the i-th sub-dictionary, we select a block from the block set, and extract its features. To avoid duplicate feature vectors in a sub-dictionary, we search all vectors in the sub-dictionary. The feature vector is appended to the sub-dictionary if no similar feature is found. Otherwise, the block is discarded, and another block is tested. This process is repeated for every block in the set. To determine whether two feature vectors are matched, a similarity metric is needed. The candidate metrics may be the well-known Euclidian distance, L1 norm, L2 norm, PSNR, SSIM [18], etc.

Note that the updating is performed in each sub-dictionary, not in the entire dictionary. It only avoids duplicate feature vectors in any individual sub-dictionary. However, duplicate

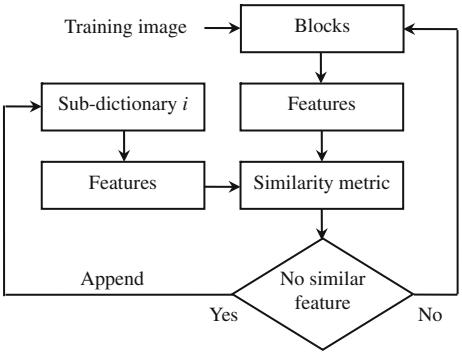


Fig. 3 Sub-dictionary updating

vectors may exist in other sub-dictionaries. This means that dividing the dictionary into a set of sub-dictionaries reduces computation cost at the expense of storing duplicate features in the whole dictionary.

2.3 Hash generation

Let $H(\mathbf{I}, g, q, p, k_1, k_2, k_3)$ be an image hash function, where $\mathbf{I}$ is the input image, g, q, p, k_1, k_2 and k_3 are control parameters. We randomly select g sub-dictionaries from G sub-dictionaries, and establish the mapping relation between the sub-dictionary and image block by a secret key k_1. We let g equal the block number to make a one-to-one mapping between the sub-dictionaries and the blocks in the following three steps:

1) Use a random generator controlled by k_1 to produce G pseudo-random numbers.
2) Sort the G numbers to produce an ordered sequence.
3) Record the original positions of these ordered numbers in an array A. Thus, the j-th block corresponds to the sub-dictionary $A[j]$.

Having obtained the mapping, we can find a feature of sub-dictionary $A[j]$ to represent the j-th block. As shown in Fig. 4, only the top q features are used as candidates, from which we pseudo-randomly select p features by using another secret key k_2 to produce a mapping array as described in the above. Thus the most similar feature representing the image block is determined among the p features. In general, the same similarity metric used in the dictionary construction is selected. The final argument k_3 is an encryption key to permute the extracted features by generating a mapping array as described in the above.

Here k_1, k_2, and k_3 may be simply the seeds used for pseudo-random number generation. Keeping them different and hard to guess is desirable. Since the features are orderly stored, and the top q features are not changed, the correct image hash can always be extracted with the same parameters even if the dictionary has been updated.

Figure 5 shows a block diagram of hash generation. The image $\mathbf{I}$ is first preprocessed to produce a normalized image, and divided into non-overlapping blocks. According to the mapping relation, find the most similar features to represent these blocks. Concatenate the matched features to obtain an intermediate hash. To improve security, the intermediate hash is pseudo-randomly scrambled. The hash sequence is encoded/compressed to give the final image hash.

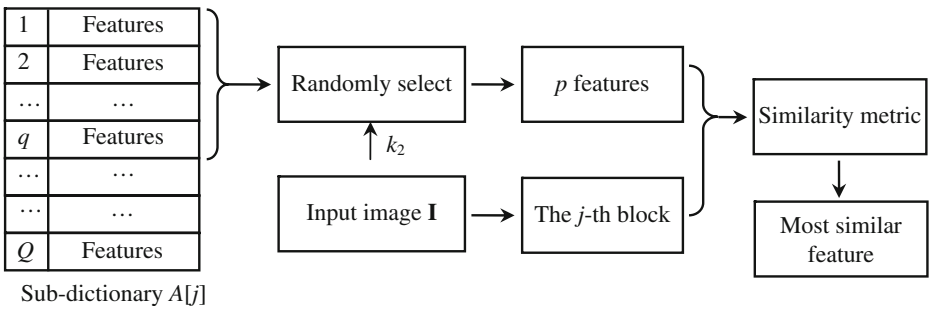


Fig. 4 Diagram showing the match process between image blocks and features

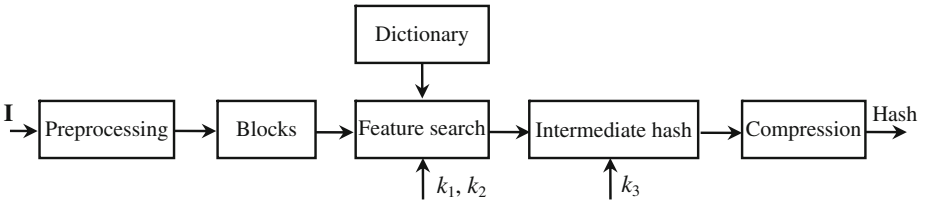


Fig. 5 Block diagram of hash generation

3 Implementation of a hashing system

3.1 Feature vector and similarity metric

Based on the framework introduced in the above, we propose an implementation of the lexicographic-structured scheme using discrete cosine transform (DCT) and non-negative matrix factorization (NMF). DCT is calculated to capture the block structure, and a structural similarity metric is then used to find a matched block in the dictionary. NMF is performed to generate words, which is used to form the intermediate hash.

To build the dictionary, we need to collect a large quantity of image blocks. For this purpose, select blocks of size $2b \times 2b$ containing significant contents from the source images. Blocks having little details such as those in a large white wall, or taken from a patch of blue sky, are discarded. This may be done by using an SVD-based approach previously described in [19]. Consider the singular values of a block, and use the ratio of the first two singular values, S_1/S_2, to measure its "roughness". A flat block with little information has a large ratio, while a busy block has a small ratio. In extreme cases, a block of constant values has $S_1/S_2=\infty$, and a block containing white noise has $S_1/S_2=1$. We can set a threshold to ignore blocks with large S_1/S_2 ratios.

For each block, two-dimensional DCT coefficients are obtained. After zigzag scanning, the first K components are used since they represent the skeleton of the block therefore may be used to examine structural similarity of the blocks. Generally, the larger the K value, the better the quality of block skeleton. This value is also related to the block size. A large block requires a large K to get a good quality of the block skeleton. Note that as the K value increases, more space is required to store the DCT coefficient vector. A tradeoff is thus needed in choosing the K value. In the experiment, we choose $b=32$ and $K=50$ for a 512×512 normalized image.

Following [18] and [14], we use the correlation coefficient as a metric to measure structural similarity between two image blocks:

$$\rho = \frac{\sigma_{\mathbf{XY}}}{\sigma_{\mathbf{X}}\,\sigma_{\mathbf{Y}}} \tag{1}$$

where $\sigma_{\mathbf{X}}$ and $\sigma_{\mathbf{Y}}$ are standard deviations of blocks $\mathbf{X}$ and $\mathbf{Y}$, respectively, and $\sigma_{\mathbf{XY}}$ the covariance:

$$\sigma_{\mathbf{XY}} = \frac{1}{4b^2 - 1}\sum_{j=1}^{4b^2}\left[X(j) - \mu_{\mathbf{X}}\right]\left[Y(j) - \mu_{\mathbf{Y}}\right] \tag{2}$$

In (2), $4b^2$ is the total pixel number in the block, $X(j)$ and $Y(j)$ are the j-th pixel values, and $\mu_{\mathbf{X}}$ and $\mu_{\mathbf{Y}}$ their mean values. If both standard deviations are zero, set $\rho=1$. If only one

of them is zero, let $\rho=0$. $\mathbf{X}$ and $\mathbf{Y}$ are considered as similar blocks if ρ is greater than a predefined threshold T_ρ. In evaluating similarity between two blocks, we must convert the DCT coefficient vector back into the image domain since the inputs of the similarity metric are pixel values. To do so, form a vector of size $4b^2\times1$ from the DCT coefficient vector and append zeros to the missing elements, convert it to a matrix of size $2b\times2b$ using inverse zigzag scanning, and then calculate inverse 2-D DCT.

The NMF coefficient vector is obtained by using the method as described in [13]. In general, NMF can be used as a technique of dimensionality reduction. Let $\mathbf{V}$ be a non-negative matrix of size $M\times N$. NMF can be achieved by using the following updating rules.

$$\begin{cases} B_{m,r} \leftarrow B_{m,r}\dfrac{\sum_{n=1}^{N} C_{r,n}V_{m,n}/(\mathbf{BC})_{m,n}}{\sum_{n=1}^{N} C_{r,n}} \\ C_{r,n} \leftarrow C_{r,n}\dfrac{\sum_{m=1}^{M} B_{m,r}V_{m,n}/(\mathbf{BC})_{m,n}}{\sum_{m=1}^{M} B_{m,r}} \end{cases} \tag{3}$$

$$m = 1, \ldots, M; n = 1, \ldots, N; r = 1, \ldots, R$$

where $\mathbf{B}$ of size $M\times R$ and $\mathbf{C}$ of size $R\times N$ are called the base matrix and the coefficient matrix (or encoding matrix), respectively, and R the rank for NMF. In this work, an image block is a non-negative matrix of size $2b\times2b$, which is divided into four sub-blocks sized $b\times b$. Stack the columns of each sub-block to form a $b^2\times1$ vector $\mathbf{v}_i=[v_{i,1}, v_{i,2}, \ldots, v_{i,M}]^{\mathrm{T}}$ where $M=b^2$, and $i=1, 2, 3, 4$. These vectors are used as columns to produce an $M\times4$ matrix $\mathbf{V}$. Apply NMF to $\mathbf{V}$, and obtain a coefficient matrix $\mathbf{C}$. The NMF coefficient vector of size $4R\times1$ is obtained by concatenating the columns of $\mathbf{C}$.

3.2 Proposed hashing scheme

In the preprocessing, a series of operations including image resizing, color space conversion, and filtering are carried out to produce a normalized image of size $U\times U$. Image resizing achieved by bilinear interpolation is done to make an equal sized hash resistant to scaling. For a color image, only the luminance component in the YCbCr space is considered. The purpose of low-pass filtering is to reduce high frequency components and alleviate influences of minor image modifications, e.g., noise contamination and filtering, on the final hash value.

The normalized image is first divided into non-overlapping blocks of size $2b\times2b$. For convenience, we let U be a multiple of $2b$. Thus the total number of blocks is $J=[U/(2b)]^2$, which equals the parameter g defined in Subsection 2.3. Let $\mathbf{X}_j$ denote the j-th block where $j=1, 2, \ldots, J$. For security, a secret key k_1 is used to establish the mapping relation between $\mathbf{X}_j$ and sub-dictionary. In each sub-dictionary, we pseudo-randomly select p vectors from the top q DCT coefficient vectors by k_2, convert them into image blocks by inverse DCT, and calculate similarity values between $\mathbf{X}_j$ and these blocks from Eq. 1. The block with maximum ρ is considered as the most similar one. Thus, we use its corresponding NMF coefficient vector to represent $\mathbf{X}_j$, denoted as $\mathbf{c}_j$. After similarity calculations, we obtain J NMF coefficient vectors. By randomly arranging these vectors, an intermediate hash is obtained:

$$\mathbf{C}_{\mathrm{U}} = [\ \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \ldots, \mathbf{c}_J\ ] \tag{4}$$

Finally, we quantize $\mathbf{C}_U$ by using the rule introduced in [13]:

$$c_{r,n}^{(b)} = \begin{cases} 0 & c_{r,n} \le c_{r,n+1} \\ 1 & c_{r,n} > c_{r,n+1} \end{cases} \qquad n = 1, 2, \ldots, J;\ r = 1, 2, \ldots, 4R \tag{5}$$

where $c_{r,n}$ denotes the entry of $\mathbf{C}_U$ in the r-th row and the n-th column, and $c_{r,\,J+1}=c_{r,1}$. Concatenate the quantized values, and then obtain a binary string, which is the final hash. So the hash length L is equal to the entry number of $\mathbf{C}_U$, i.e., $L=4RJ$.

4 Experiments and performance analysis

To test the hash performance, we collect 2,000 color images, including 12 standard images sized 256×256 or 512×512, 100 images captured with digital cameras ranging from 1,280×960 to 2,592×1,944, 100 images downloaded from the Internet ranging from 415×260 to 853×530, and 1,688 images from the image database of Washington University [2] ranging from 480×722 to 883×589. The 2,000 images are divided into two sets, each containing 1,000 images, used to build the dictionary and study the system performance.

The parameters used in dictionary construction are: number of sub-dictionaries $G=100$, each containing $Q=1{,}000$ words, block size $2b\times 2b=64\times 64$, number of DCT coefficients taken from each block $K=50$, rank of NMF $R=2$, and the threshold for similarity comparison $T_\rho=0.7$. The choice of threshold is based on the following observations: Correlation coefficients between the 64×64 blocks in standard test images such as Airplane, Baboon and Lena, and the corresponding modified versions are greater than 0.8 with only a few exceptions that are slightly lower. The modifications tested include JEPG compression, watermark embedding, and low-pass filtering.

In hash generation, the input image is resized to 512×512, low-pass filtered using a 3×3 Gaussian mask with a unit standard deviation, and divided into non-overlapping blocks of size 64×64, i.e., $U=512$. Thus, the block number J is $[U/(2b)]^2=[512/64]^2=64$, and the hash length L is $4RJ=4\times 2\times 64=512$ bits.

Clearly, the hash length is directly related to the block size. The smaller the block size, the more the number of blocks, and thus the longer the hash length will be produced. However a large block size means poor sensitivity to local area image tampering, and a small number of blocks hurts the hash uniqueness. We find a block size of 32×32 or 64×64 to be an acceptable compromise. The block number is also related to the computation cost which will be discussed in Subsection 4.3.

To find the most similar features, the 50-element DCT coefficient vectors are randomly fetched from the top 1,000 features, i.e., $p=50$, and $q=1{,}000$. Normalized Hamming distance is calculated to measure similarity between hashes:

$$d\left(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}\right) = \frac{1}{L}\sum_{l=1}^{L}\left| h_l^{(1)} - h_l^{(2)} \right| \tag{6}$$

We compare the method proposed in the present work with the RASH method [11], the NMF-NMF-SQ scheme [9], and a previously introduced NMF method [13]. All images are normalized to a fixed size 512×512 before hash extraction as described in [9] and [13]. To measure similarity, the RASH method exploits peak of cross correlation (PCC), and the

NMF-NMF-SQ hashing uses L2 norm. Both the lexicographic-structured DCT/NMF scheme and the NMF method of [13] use the normalized Hamming distance as a similarity metric.

In [13], the number of blocks are 8×8=64, and the rank of the NMF is 5. In [9], number of sub-images is 80, height and width of sub-images is 64, rank of the first NMF is 2, rank of the second NMF is 1, and the hash length is 64 decimal digits. The same parameters are used in the present comparison except the sub-image number and size because the normalized image size in [9] is 256×256, while that of the present work is 512×512.

4.1 Perceptual robustness

StirMark 4.0 [10] is applied to attack five standard color images sized 512×512: Airplane, Baboon, House, Lena and Peppers. Content-preserving manipulations in the experiment include JPEG coding, watermark embedding, additive noise contamination, and image scaling. In addition, brightness adjustment and contrast adjustment using Adobe Photoshop, and gamma correction and 3×3 Gaussian filtering using MATLAB are also tested. Parameters used in these operations are listed in Table 1. Effect of image rotation will be discussed in the next subsection.

Calculate the hash distances between the original images and their attacked versions for the proposed method, the RASH method, the NMF-NMF-SQ scheme, and the NMF method of [13], respectively. The results are presented in Fig. 6(a)–(d), respectively. Indices of the abscissa indicate various attacks on the image as listed in Table 1. The ordinates of Fig. 6(a) and (d) are the normalized Hamming distances between the original and attacked images, while that of (b) and (c) are the PCC values and L2 norms respectively.

In Fig. 6(a), the maximum normalized distance is 0.065, implying that the proposed scheme is robust against the listed content-preserving operations. Thus, we can safely set a threshold at 0.08. If the normalized Hamming distance of two images is smaller than 0.08, they are considered visually identical. Similarly, to make the previous method in [13] resistant to all listed attacks, we choose a threshold 0.12. The authors of [11] take 0.87 as the threshold. All PCC values in Fig. 6(b) are above this value, indicating the RASH method is robust against content-preserving changes. In Fig. 6(c), all L2 norms are below 2800 except a few cases of strong watermarking attacks, brightness adjustments and gamma corrections not treated in [9]. So 2800 can be selected as a threshold. Comparing Fig. 6(a) with (d), we observe that most of the normalized Hamming distances in Fig. 6(a) are

Table 1 Image manipulations and the corresponding parameters

Type no.	Manipulation	Description	Parameter value
I	JPEG compression	Quality factor	20, 30, … , 100
II	Additive noise	Level	1, 2
III	Watermark embedding	Strengthen	10, 20, … , 100
IV	Scaling	Ratio	0.5, 0.75, 0.9, 1.1, 1.5, 2.0
V	Brightness adjustment	Photoshop's brightness scale	10, 20, −10, −20
VI	Contrast adjustment	Photoshop's contrast scale	10, 20, −10, −20
VII	Gamma correction	γ	0.75, 0.9, 1.1, 1.25
VIII	3×3 Gaussian filtering	Standard deviation	0.1, 0.2, ... , 1.0

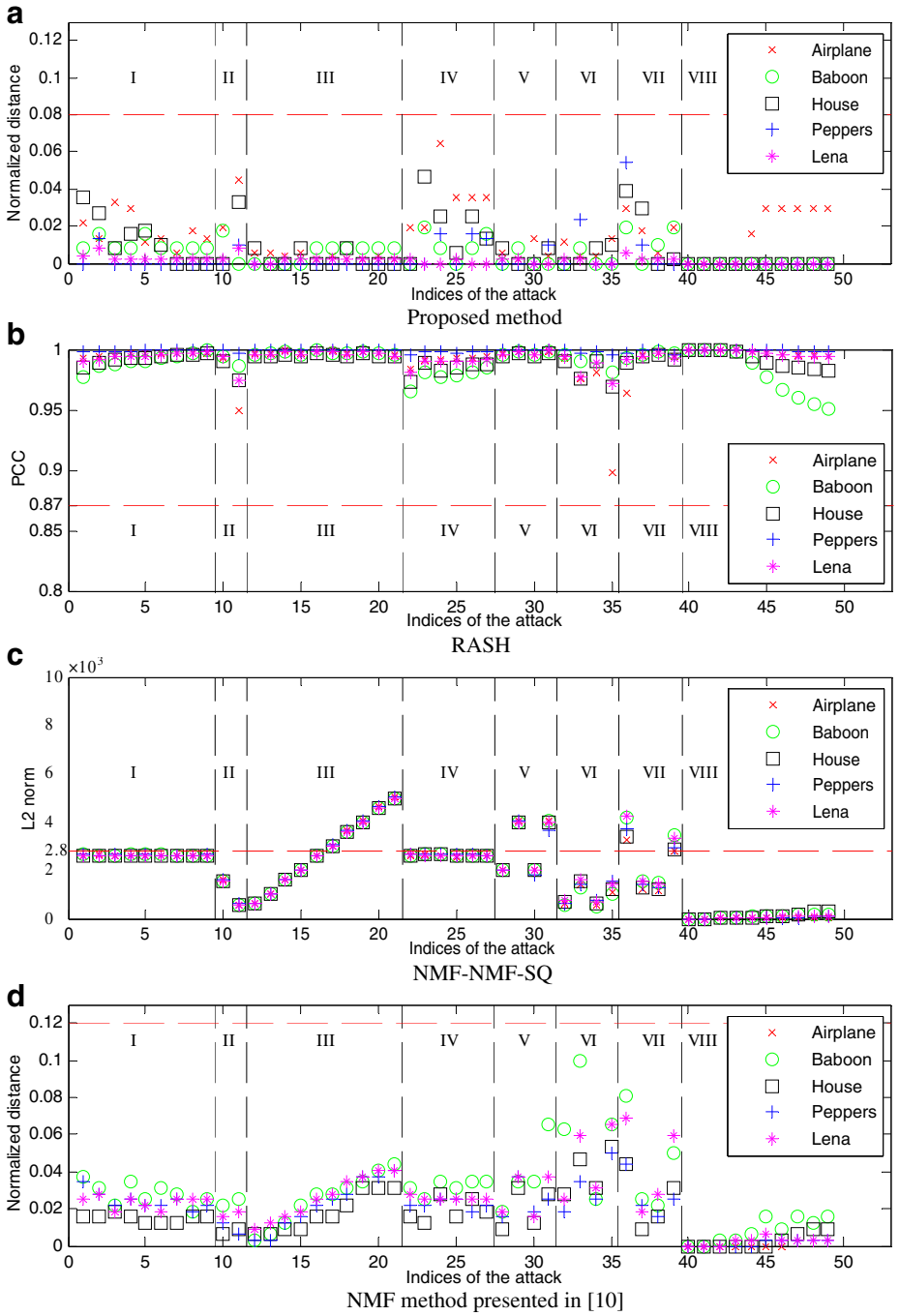


Fig. 6 Robustness performances. Indices in abscissa indicate different attacks listed in Table 1. **a** Proposed method, **b** RASH, **c** NMF-NMF-SQ, **d** NMF method presented in [13]

smaller than those of (d). It means that the method proposed in the present paper is more robust than that of [13] when using the same threshold.

4.2 Anti-collision performance

Uniqueness means low collision probability. Collision happens if Hamming distance d between two different images is less than a given threshold. To find the probability of collision between different images, we generate hashes of 1,000 different images, and calculate the Hamming distance between each pair of hashes to obtain 499,500 results. In order to determine the most probable distribution of the Hamming distance among Poisson, Rayleigh, Weibull, lognormal, normal, and Gamma distributions, chi-square tests [5] are applied to these results. Parameters of these distributions are first obtained from the maximum likelihood estimation. Then the statistics χ^2 is calculated:

$$\chi^2 = \sum_{i=0}^{L} \frac{(n_i - n_{\text{total}} P_i)^2}{n_{\text{total}} P_i} \tag{7}$$

where n_{total} is the number of trials, n_i the occurrence frequency of d being i (i=0, 1, 2, …, L), L the hash length, and P_i the probability at i obtained by using the probability density function (PDF). The results of chi-square tests are presented in Table 2. Since χ^2 of normal distribution is the smallest, we identify the distribution of Hamming distances between different images as a normal distribution with its mean and standard deviation being μ=217 and σ=22.87, respectively. Given a threshold T, the collision probability can be calculated as follows:

$$\begin{aligned} \Pr(d \le T) &= \frac{1}{\sqrt{2\pi}\,\sigma} \int_{-\infty}^{T} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right] dx \\ &= \frac{1}{2}\operatorname{erfc}\left(-\frac{T-\mu}{\sqrt{2}\,\sigma} \right) \end{aligned} \tag{8}$$

where erfc(·) is the complementary error function. As T increases, the calculated collision probability will increase, while the probability of false judgment for content-preserving modification will decrease. Table 3 lists collision probabilities when using different normalized thresholds. We can choose proper threshold values to suit specific applications.

It has been shown that the Hamming distance in the previous method [13] follows a normal distribution with μ=147 and σ=14.2, and the metric of the NMF-NMF-SQ scheme

Table 2 Results of chi-square test for hash uniqueness

Distribution type	Estimated parameters	χ^2
Poisson	λ=217	9.6×10^{9}
Rayleigh	β=154.50	1.33×10^{6}
Weibull	β=10.04, η=227.60	1.83×10^{11}
Lognormal	μ=5.38, σ=0.11	59246
Normal	μ=217, σ=22.87	350
Gamma	α=89.11, β=2.44	4825

Table 3 Collision probabilities with different thresholds

Normalized threshold	Collision probability
0.08	6.94×10^{-15}
0.10	2.08×10^{-13}
0.12	5.16×10^{-12}
0.14	1.05×10^{-10}
0.16	1.75×10^{-9}
0.18	2.40×10^{-8}
0.20	2.71×10^{-7}

satisfies a Gamma distribution with α=9.5 and β=1,564. We calculate the collision probabilities for hashing methods under the respective safe thresholds mentioned in Subsection 4.1. The results of the method proposed here, the previous method [13], and the NMF-NMF-SQ method are 6.94×10^{-15}, 1.02×10^{-14}, and 4.3×10^{-5} respectively. Collision probability reported in [11] is 5.86×10^{-6}, which is much higher than the proposed method and the previous method [13], but better than the NMF-NMF-SQ method [9]. Therefore, the anti-collision property of the proposed method is equivalent to that of the previous method, both being much better than RASH and NMF-NMF-SQ.

The present method also has a very low collision probability for hashes of the same image generated with different keys. To show this, we extract 2,000 hashes from the test image Man, only changing k_1 while keeping the other parameters unchanged. Compute the Hamming distances between each pair out of the 2,000 hashes, and calculate the collision probability. The chi-square test results give a normal distribution of the Hamming distance with μ=247 and σ=22.01, indicating very low collision probability of 3.94×10^{-21} when using a normalized threshold of 0.08.

As to security, since the RASH method does not take any measure to prevent forgery in hash generation, it is easy to coin a fake hash. On the other hand, because hash values generated both in the NMF-NMF-SQ scheme and in the previous work depend on the input image and on the selection of randomized block, reuse of a key can lead to accurate estimation of the block selection. Therefore these three methods all have the risk that an attacker can generate a totally different hash by using a similar image, or synthesize visually different images from the same hash.

By using NMF coefficients to construct the dictionary, the proposed method has inherited good robustness and uniqueness from other NMF-based methods. Moreover, the aforementioned security risk is not a problem for the lexicographical technique. In the lexicographical framework, the hash is no longer dependent solely on the block selection. It is determined by the dictionary, the input image, and the secret keys. Without the knowledge of all these, it is extremely hard to forge a correct hash. In particular, the dictionary is constructed using a very large quantity of source images so that it is virtually impossible to duplicate. Meanwhile, the number of mapping relations between the image blocks and the sub-dictionaries are very large. So it is still very difficult to forge a correct hash even if the dictionary is available. Therefore, in addition to the good anti-collision performance, the proposed hash is also very hard to forge by any adversary.

Block-based methods are inherently weak in resisting image rotation because rotation destroys the block structure. To reach a compromise, one can raise the threshold to make the hash tolerate small angle rotation. For example, by rotating the aforementioned test

images by ±0.25°, ±0.5°, ±0.75° and ±1.0°, the average normalized Hamming distance is 0.143. Thus we can set T=0.16 to trade the anti-collision property for resistance of small angle rotation. Collision probability in this case becomes 1.75×10^{-9}, still several orders of magnitude lower than RASH and NMF-NMF-SQ.

Consideration of small angle rotation has practical significance since it is often performed in the post-processing of digital photographs to correct imperfect leveling of the picture. Using a larger block size can help enhance the rotation-resistant capability, but this reduces the block number so that the performance of anti-collision and local forgery detection may be sacrificed. Further improvement of the anti-rotation performance would require some modifications to the system design, for example, using overlapped blocks and taking a central circle in the block for feature extraction, scanning blocks in a spiral manner starting from the center of the image, and assigning smaller weights to the blocks toward the border. Nevertheless, arbitrarily large angle rotation is considered as not content-preserving for the targeted applications.

4.3 Effect of dictionary size and number of words used in feature extraction

In the above experiment, the dictionary has a total of $D=G\cdot Q$=100,000 words. As the number of sub-dictionary G is fixed (G=100 here), expansion of the dictionary means increase of Q. A larger sub-dictionary provides more choices in finding the best match of the word with a given image block, hence better performance of the final hash.

To show that the hash performance is improved with the number of words increased in the sub-dictionaries, we limit the value q to 10, 50 and 80 respectively, and let $p=q$ to generate hashes. This is equivalent to a test of the influence of different dictionary sizes D= 1,000, 5,000, and 8,000. We use the same images as in the previous subsection for this experiment. Normalized Hamming distances between the original and modified images are calculated and shown in Fig. 7 for different q values.

It is observed that, as the usable words increase, most of the large distances are decreased with a few exceptions, while many small distances show a moderate increase. Such increase is a result of more choices of words that are similar in “meaning” but slightly different. In other words, a larger dictionary is more sensitive to subtle changes in the block contents. Nonetheless, the slightly increased small distances are not harmful because they are still much lower than the threshold and, further, the distance is not the only specification of the hash’s overall performance. If there were only a few choices of words, some hashes of different images would be likely to collide. As explained in the next paragraph, availability of more usable words makes substantial enhancement of the anti-collision capability, thus effectively improving the hash performance as a whole.

To check collision probability, chi-square tests are carried out under different dictionary sizes, and collision probability is then calculated. The results are given in Table 4. It is clear that collision probability is high when the available words in the dictionary are too few. As the training images increase, more words are appended to the dictionary, making collision probability lower. This trend appears to slow down when p is above 50. However, if the size of sub-dictionaries Q is increased therefore q can be larger to make more words available in matching block features even with a small p, the collision probability decreases significantly. For example, if we randomly select 10 words from 1,000 words in hash generation, i.e., q=1,000 and p=10, the collision probability drops to 2.32×10^{-14}.

A larger dictionary means more words in each sub-dictionary. Larger sub-dictionaries provide possibility of selecting more words to represent an image. Most computation in the hash generation is done in finding the best match between words and image blocks. The

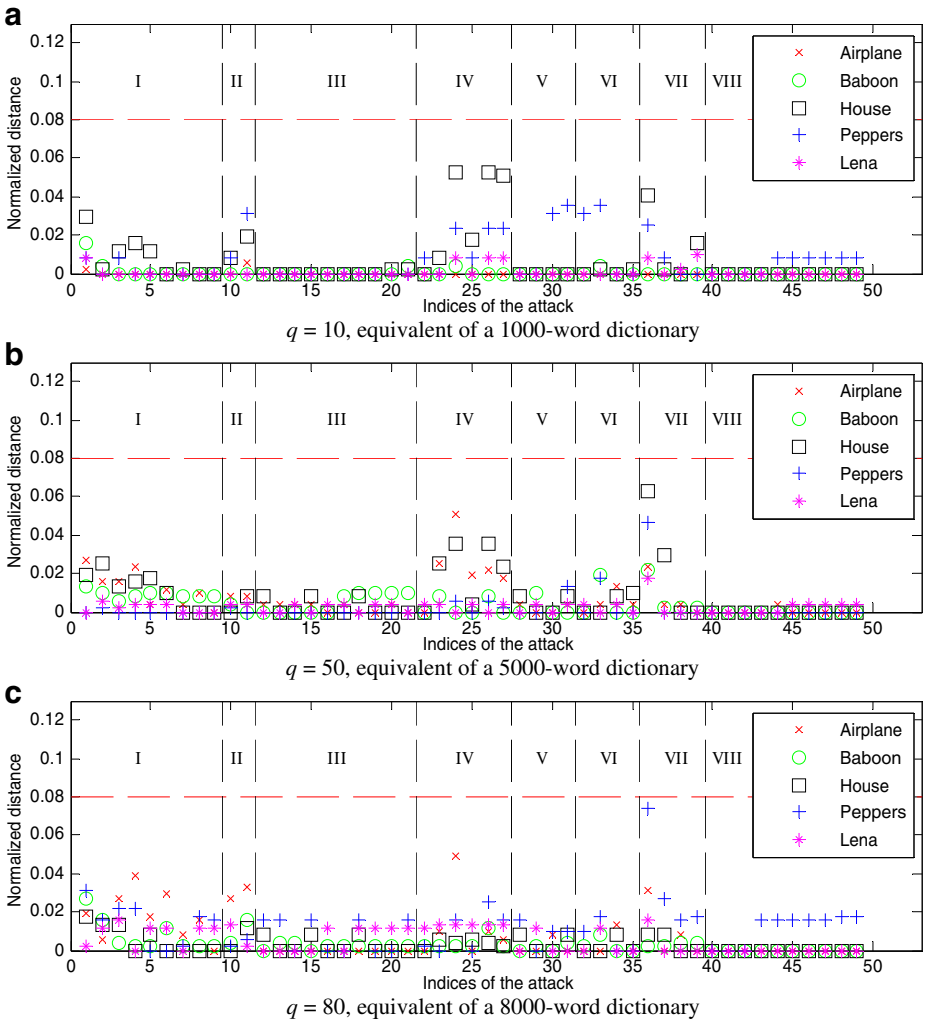


Fig. 7 Robustness with different sizes of sub-dictionaries. Indices in the abscissa indicate the attacks as listed in Table 1. **a** q=10, equivalent of a 1000-word dictionary, **b** q=50, equivalent of a 5000-word dictionary, **c** q=80, equivalent of a 8000-word dictionary

computation time for generating an image hash, $t=Jpt_0$, is proportional to J and p, where t_0 is the time used to evaluate Eq. 1, and J the number of image blocks. In other words, the smaller the block size and the more features in a block are used, the more the computation time is required.

To measure the time consumed in hash generation under different dictionary sizes and different numbers of words used in feature extraction, i.e., different q and p values, we calculate 100 hashes of different images and record the consumed time, and find the average time needed to produce an image hash. In the experiment, only q and p are varied while the other parameters are fixed. A desktop computer with a 3.0 GHz Pentium 4 CPU and 512 MB RAM was used, running MATLAB. The results are listed in

Table 4 Collision probabilities under different dictionary sizes

p	q	Equivalent dictionary size	Collision probability
10	10	1000	1.09×10^{-5}
50	50	5000	1.88×10^{-10}
80	80	8000	0.61×10^{-10}
10	50	5000	6.93×10^{-9}
10	150	15000	8.21×10^{-11}
10	500	50000	4.72×10^{-12}
10	1000	100000	2.32×10^{-14}

Table 5. We observe from the table that the dictionary size, $G{\cdot}q$ where G is fixed in the experiment, has little influence on the computation load, and it is the number of words, p, taken from each sub-dictionary that matters. When p is 10 and q is 1,000, the computation takes 4.1 s. The consumed time is increased approximately linearly with p if q is unchanged. This is consistent with the theoretical analysis. As a reference, the RASH and NMF-NMF-SQ methods take 16.9 and 2.4 s respectively, and the method introduced in [13] uses 6.3 s, under the same conditions.

5 Applications

Since the lexicographic-based hash is robust against normal digital processing, it can be used in content-based watermarking generation and image copy detection. For copy detection, we can set a threshold leading to a low collision probability, such as 0.08, or 0.10. In this section, we show an application of the method in searching frames with similar scenes in a video sequence. This can be used, for example, by an advertiser to check the time and number of repetitions of a particular commercial clip being broadcast during a TV program, or for the police to find a suspect or a trouble-making vehicle from video records.

In this experiment, the video sequence is taken from the image database of USC-SIPI [15]. As shown in Fig. 8, the top-left image is a query image. Calculate the normalized Hamming distances between the query image and images in the database to obtain the results. Images in the first and second rows give the top nine most similar ones with the query image. To provide a comparison, five images with different scenes are given in

Table 5 Average time consumed in hash generation under different p and q values

p	q	Equivalent dictionary size	Average time (seconds)
10	600	60000	4.0
10	800	80000	4.0
10	1000	100000	4.1
20	1000	100000	6.3
30	1000	100000	8.4
40	1000	100000	10.4
50	1000	100000	12.6

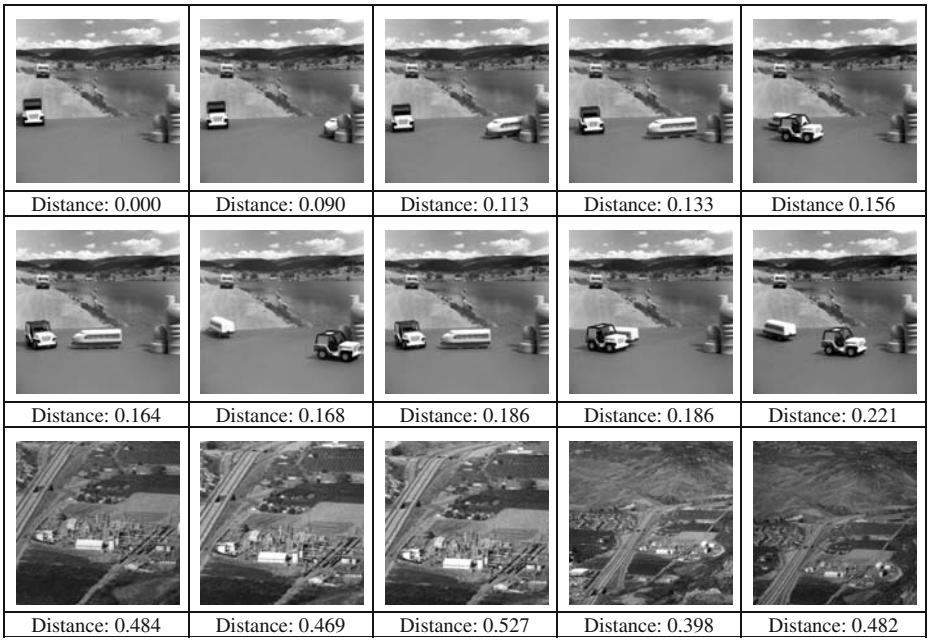


Fig. 8 Results of retrieving images with similar scenes

the bottom row. Images containing similar scenes with the query image have a small distance, while those with different scenes have a large distance. We use a threshold 0.2 in this case to successfully retrieve most of the similar images in the database, with a low false positive probability of 2.71×10^{-7}.

6 Conclusions

In this paper, we have developed a lexicographical framework to generate secure image hashes, and implemented a scheme using DCT and NMF. The proposed scheme is robust against normal content-preserving manipulations such as JPEG compression, watermarking attacks, scaling, adjustment of brightness and contrast, and low-pass filtering. Hashes of different images, and of the same image generated with different keys have very low collision probability. The dictionary is a key component of the proposed scheme, which is constructed by taking features from a very large quantity of image blocks. Without knowledge of the dictionary, it is practically impossible to forge the hash for any given image. The update procedure guarantees that there are no redundant features in sub-dictionary. The control parameters of hash function ensure a backward-compatible framework, and make the image hashes secure.

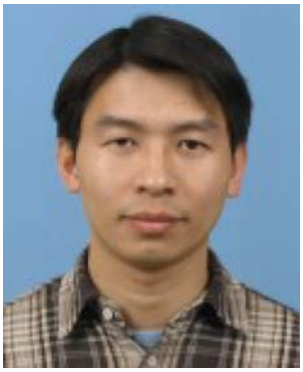
It has been shown that the proposed framework has good robustness and uniqueness. As expected, a large dictionary, and taking more words from the sub-dictionaries for feature matching can lead to better performance. However, using too many words in the sub-dictionaries does not provide the performance advantage in a linear fashion, but only increases the computation burden linearly. Therefore a proper choice of the system parameters is important.

The technique is applicable to digital rights management (DRM) including detection of illegal image manipulations, CBIR, watermarking, etc. Further research on the lexicographical structured image hashing system is under way, mainly concentrated on the capability of image tampering detection. Issues being considered include the introduction of color features, image-rotation resistance, more sophisticated dictionary generation and maintenance mechanisms, etc.

Acknowledgements This work was supported by the Natural Science Foundation of China (60773079, 60872116, and 60832010), the High-Tech Research and Development Program of China (2007AA01Z477), and the Innovative Research Foundation of Shanghai University for Ph.D. Programs (shucx080148). The authors would like to thank the anonymous referees for their valuable comments and suggestions.

References

1. Fridrich J, Goljan M (2000) Robust hash functions for digital watermarking. In: Proceedings of IEEE International Conference on Information Technology: Coding and Computing (ITCC'00), Las Vergas, USA, Mar. 27–29, 2000, pp 178–183
2. Ground Truth Database, http://www.cs.washington.edu/research/imagedatabase/groundtruth/. Accessed 8 May 2008
3. Kozat SS, Mihcak K, Venkatesan R (2004) Robust perceptual image hashing via matrix invariants. In: Proceedings of IEEE Conference on Image Processing (ICIP'04), Singapore, Oct. 24–27, 2004, pp 3443–3446
4. Lefebvre F, Macq B, Legat J-D (2002) RASH: Radon soft hash algorithm. In: Proceedings of European Signal Processing Conference (EUSIPCO'02), Toulouse, France, Sep. 3–6, 2002, pp 299–302
5. Lehmann EL, Romano JP (2005) Testing statistical hypotheses, 3rd edn. Springer, New York, pp 590–599
6. Lin Y, Chang SF (2001) A robust image authentication system distinguishing JPEG compression from malicious manipulation. IEEE Trans Circuits Syst Video Technol 11(2):153–168
7. Mao Y, Wu M (2007) Unicity distance of robust image hashing. IEEE Trans Inf Forensics Secur 2 (3):462–467
8. Monga V, Evans BL (2006) Perceptual image hashing via feature points: performance evaluation and trade-offs. IEEE Trans Image Process 15(11):3453–3466
9. Monga V, Mihcak MK (2007) Robust and secure image hashing via non-negative matrix factorizations. IEEE Trans Inf Forensics Secur 2(3):376–390
10. Petitcolas FAP (2000) Watermarking schemes evaluation. IEEE Signal Process Mag 17(5):58–64
11. Roover CD, Vleeschouwer CD, Lefebvre F, Macq B (2005) Robust video hashing based on radial projections of key frames. IEEE Trans Signal Process 53(10):4020–4036
12. Swaminathan A, Mao Y, Wu M (2006) Robust and secure image hashing. IEEE Trans Inf Forensics Secur 1(2):215–230
13. Tang Z, Wang S, Zhang X, Wei W, Su S (2008) Robust image hashing for tamper detection using non-negative matrix factorization. Journal of Ubiquitous Convergence and Technology 2(1):18–26
14. Tang Z, Wang S, Zhang X, Wei W (2009) Perceptual similarity metric resilient to rotation for application in robust image hashing. In: Proceedings of the 3rd International Conference on Multimedia and Ubiquitous Engineering (MUE'09), Qingdao, China, June 4–6, 2009, pp 183–188
15. USC-SIPI Image Database, http://sipi.usc.edu/services/database/database.html. Accessed 12 Jan. 2007
16. Venkatesan R, Koon S-M, Jakubowski MH, Moulin P (2000) Robust image hashing. In: Proceedings of IEEE International Conference on Image Processing (ICIP'00), Vancouver, Canada, Sep. 10–13, 2000, pp 664–666
17. Wang S, Zhang X (2007) Recent development of perceptual image hashing. Journal of Shanghai University 11(4):323–331
18. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
19. Wang S, Lu X, Su S, Zhang X (2007) Image block feature vectors based on a singular-value information metric and color-texture description. Journal of Shanghai University 11(3):205–209



Zhenjun Tang received the B.S. degree and the M.S. degree from Guangxi Normal University, Guilin, P.R. China, in 2003 and 2006, respectively. He is currently a Ph.D. student in Shanghai University, Shanghai, P.R. China. His research interests include image processing and information security of digital media.

Shuozhong Wang received the B.S. degree from Peking University, Beijing, P.R. China, in 1966 and the Ph. D. degree from University of Birmingham, Birmingham, England, in 1982. Currently, he is a Professor of Shanghai University. His research interests include image processing, audio processing, and information hiding.

Xinpeng Zhang received the B.S. degree in computation mathematics from Jilin University, Jilin, P.R. China, in 1995, and the M.E. and Ph.D. degrees in communication and information system from Shanghai University, Shanghai, China, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. His research interests include information hiding, image processing, and digital forensics.

Weimin Wei received the M.S. degree from Wuhan University, Wuhan, P.R. China, in 2004. He is currently a Ph.D. student in Shanghai University, Shanghai, P.R. China. His research interests include image processing, digital forensics and data mining.

Yan Zhao received M.S. degree from Shanghai Jiao Tong University, P.R. China, in 2005. She is currently a Ph.D. student in Shanghai University and a lecturer of Shanghai University of Electric Power. Her research interests include mobile communication and image processing.