Int. J. Electron. Commun. (AEÜ) 67 (2013) 717–722

Contents lists available at SciVerse ScienceDirect

International Journal of Electronics and Communications (AEÜ)

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

Robust image hash function using local color features

Zhenjun Tang*, Xianquan Zhang, Xuan Dai, Jianzhong Yang, Tianxiu Wu

Department of Computer Science, Guangxi Normal University, No. 15 Yucai Road, Guilin 541004, PR China

ARTICLE INFO

Article history:
Received 28 September 2012
Accepted 26 February 2013

Keywords:
Hash function
Image hashing
Color space
Digital watermarking
Image retrieval

ABSTRACT

Conventional image hash functions only exploit luminance components of color images to generate robust hashes and then lead to limited discriminative capacities. In this paper, we propose a robust image hash function for color images, which takes all components of color images into account and achieves good discrimination. Firstly, the proposed hash function re-scales the input image to a fixed size. Secondly, it extracts local color features by converting the RGB color image into HSI and YCbCr color spaces and calculating the block mean and variance from each component of the HSI and YCbCr representations. Finally, it takes the Euclidian distances between the block features and a reference feature as hash values. Experiments are conducted to validate the efficiency of our hash function. Receiver operating characteristics (ROC) curve comparisons with two existing algorithms demonstrate that our hash function outperforms the assessed algorithms in classification performances between perceptual robustness and discriminative capability.

1. Introduction

Image hash function, also called image hashing, maps an input image to a short string called image hash, which has been applied to image authentication, tamper detection [1], digital watermarking [2], image indexing, image copy detection, image retrieval [3], digital forensics, and reduced-reference image quality assessment. For example, people often use image editing tools, such as Photoshop and ACDSee, to process photographs and save them in JPEG format with different filenames. Consequently, there may be several copies of an image in the computer. These copies have the same visual contents with the original image, but their digital representations are different from that of the original one. In this case, people can exploit image hash function to efficiently search all similar versions (including the original one and its copies) of the image from large-scale image database. Unlike cryptographic hash functions such as SHA-1 and MD5, the image hash function is not sensitive to digital representation of an image. It produces the same or very similar hash values for visually identical images no matter whether their representations are the same or not. In general, an image hash function must have two properties. (1) *Perceptual robustness*: the hash function should be robust against content-preserving operations, such as JPEG compression and de-noising. In other words, the hashes of an image and its attacked versions generated by content-preserving operations should be the same. (2) *Discriminative capability*: different images should have different hashes. For special applications, image hash function should have other properties. For example, it must be controlled by one or several keys when it is applied to image authentication or digital forensics.

Many researchers have devoted themselves to developing high performance image hash functions in the last decade. For example, Venkatesan et al. [4] used the statistics of wavelet coefficients to construct image hashes. This method is not robust enough against contrast adjustment, gamma correction, and cannot detect malicious object insertion. Fridrich and Goljan [5] found that discrete cosine transform (DCT) coefficients can indicate the visual content of images, and exploited them to produce image hashes. Lin and Chang [6] presented a technique based on the observation that relations between DCT coefficients at the same position in separate blocks are preserved after JPEG compression. This method can distinguish JPEG compression from malicious attacks. In another study [7], a RAdial Variance (RAV) vector is extracted using the radial projections of image pixels. The low-frequent DCT coefficients of the RAV vector are then quantized to form the image hash. This scheme is resilient to image rotation and re-scaling, but insensitive to local change. Swaminathan et al. [8] used the Fourier–Mellin coefficients to generate image hashes. This hash function is resilient to several content-preserving modifications such as moderate geometric transforms and filtering. In [9], Kozat et al. viewed image and attacks as a sequence of linear operators, and proposed to calculate hashes using singular value decompositions (SVDs). The SVD–SVD hashing is resilient to geometric attacks, e.g., rotation, at the cost of significantly decreasing discriminative capability. Monga and Mihcak [10] first used non-negative matrix factorization (NMF) to derive image hashing, and obtained a high performance algorithm.

* Corresponding author. Tel.: +86 15295990968.
E-mail addresses: tangzj230@163.com, zjtang@gxnu.edu.cn (Z. Tang).

1434-8411/$ – see front matter © 2013 Elsevier GmbH. All rights reserved.
http://dx.doi.org/10.1016/j.aeue.2013.02.009

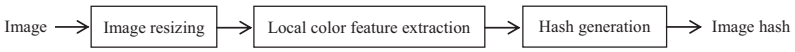

Fig. 1. Block diagram of our image hash function.

In [11], Tang et al. found the invariant relation existing in the NMF coefficient matrix and used it to construct robust image hashes. Ou and Rhee [12] applied Radon transform (RT) to the input image, randomly selected 40 projections to perform 1-D DCT, and took the first AC coefficient of each projection to produce hash. The RT–DCT hashing is resistant to rotation within 5°. In another work [13], Tang et al. proposed a lexicographical framework for robust image hashing and gave an implementation based on DCT and NMF. The lexicographical hashing consists of two parts: dictionary construction and maintenance, and hash generation. This method is more secure than traditional methods since it is virtually impossible to duplicate an identical dictionary.

A common of the above-mentioned algorithms is that they just consider gray images. For a color image, they use its luminance component for representation, and then calculate image hash based on the luminance component. Since other color features such as hue and saturation are discarded, their discriminative capacities are limited. In this paper, we propose a robust hash function for color images, which takes all components of a color image into account and then achieves a good discrimination. We conduct a lot of experiments to validate the efficiency of our hash function and the results demonstrate the our hash function have a desirable classification performances between perceptual robustness and discriminative capability.

The rest of the paper is organized as follows: Section 2 will describe the proposed image hash function. Section 3 will present the experimental results and performance comparisons. Conclusions are finally made in Section 4.

2. Proposed image hash function

As shown in Fig. 1, our image hash function consists of three steps, i.e., image resizing, local color feature extraction and hash generation. In the first step, we convert the input image into a normalized image with a standard size $k \times k$ by the bilinear interpolation. This is based on the consideration that real images have different sizes and their hashes should have the same length. In the second step, we extract local color feature from the normalized image. In the final step, we compress the extracted color features to make a short image hash. Details of the local color feature extraction and hash generation are described in the following two subsections, respectively.

2.1. Local color feature extraction

In general, the characteristic of a color image can be depicted by its hue, saturation, and luminance. The hue represents the color appearance, and can be used for color classification, e.g., red, green, and blue. The saturation, also called chroma, is the property of a color which indicates the purity or amount of white contained in the color. For example, it can distinguish the pale red from the deep red. The luminance, also called intensity, is an indicator of how bright an image area will appear and thus used to characterize its brightness. Obviously, if only one of these color properties is used, the extracted features cannot fully indicate the real characteristic of a color image and therefore limit the discriminative capability of hash system. To overcome this problem, we convert an RGB color image into two color spaces, i.e., HSI and YCbCr, and then extract features from their components, respectively. Let R, G, and B be the red, green and blue component of a pixel, where the ranges of R, G, and B are [0,1]. Suppose that H, S, and I are the hue, saturation, and intensity of the pixel color. Thus, we can obtain the values of H, S, and I by using the following rules.

$$I = \frac{R+G+B}{3} \tag{1}$$

$$S = 1 - \left[\frac{3}{R+G+B}\right] \times \min(R, G, B) \tag{2}$$

$$H = \cos^{-1}\left(\frac{(1/2)[(R-G)+(R-B)]}{\sqrt{(R-G)^2-(R-B)(G-B)}}\right)$$
$$\text{If } S = 0 \text{ then } H = 0$$
$$\text{If } \left(\frac{B}{I}\right) > \left(\frac{G}{I}\right) \text{ then } H = 360 - H \tag{3}$$

Suppose that Y, C_b, and C_r are the luminance, blue-difference chroma and red-difference chroma, respectively. Thus, the conversion from RGB color space to YCbCr color space is as follows.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{4}$$

The selection of YCbCr color space is to make the extracted features robust against JPEG compression. This is because the color images in JPEG format use the YCbCr color space.

For each component of HSI and YCbCr representations, we divide it into non-overlapping block sized $b \times b$. Suppose that N is the block number and B_i is the ith block ($i = 1, 2, \ldots, N$). Thus, we calculate the mean m and the variance u to represent the block as follows.

$$m = \frac{1}{b^2}\sum_{j=1}^{b^2} B_i(j) \tag{5}$$

$$u = \frac{1}{b^2-1}\sum_{j=1}^{b^2} [B_i(j) - m]^2 \tag{6}$$

where $B_i(j)$ is the value of the jth element of the ith block. The selection of the above statistics is based on the consideration that the mean and variance can indicate the average energy and the fluctuation of the block. We apply the formulas (5) and (6) to each component of the HSI and YCbCr color spaces, and thus obtain the color feature vector $\mathbf{v}_i$ of the ith block.

$$\mathbf{v}_i = [m_H, u_H, m_S, u_S, m_I, u_I, m_Y, u_Y, m_{C_b}, u_{C_b}, m_{C_r}, u_{C_r}]^T \tag{7}$$

where m_H and u_H are the mean and variance of the H component of HSI, m_S and u_S are the mean and variance of the S component, m_I and u_I are the mean and variance of the I component, m_Y and u_Y are the mean and variance of the Y component of YCbCr, m_{Cb} and u_{Cb} are the mean and variance of the C_b component, and m_{C_r} and u_{C_r} are the mean and variance of the C_r component. By arranging these vectors $\mathbf{v}_i$, we obtain a feature matrix $\mathbf{V}$.

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N] \tag{8}$$

2.2. Hash generation

To make the final hash as short as possible, we compress the feature matrix $\mathbf{V}$ as follows. Data normalization is first conducted.