ON SECURE IMAGE HASHING BY HIGHER-ORDER STATISTICS

Li Weng, Bart Preneel
Katholieke Universiteit Leuven
Department of Electrical Engineering
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium

ABSTRACT
Due to the fact that the same multimedia content can have different digital representations, content-based hashing is important for multimedia identification and authentication. In this paper we discuss some issues on image hashing by higher-order statistics. We show by experiments that fourth-order cumulants can be used for image hashing and they have better overall performance than fourth-order moments which are used in an existing algorithm. We further extend the algorithm to a more secure version by incorporating a secret key. It is achieved by choosing different statistic patterns from image blocks according to a secure pseudo-random number generator. Simulation shows satisfactory results.

Index Terms— Perceptual hashing, image hashing, robust hashing, higher-order statistics, cumulants

1. INTRODUCTION
The success of digital technologies such as JPEG, MPEG, MP3, and the Internet has brought the world into a multimedia era, in which an enormous amount of multimedia data is being produced and manipulated every second. Consequently, two challenging issues have arisen: 1) it is difficult to identify multimedia content in a large database; 2) it is difficult to authenticate multimedia content due to the advance of editing software. These two issues both suffer from the fact, that the same multimedia content may have different digital representations. On one hand, the same content can be represented by various file formats, qualities, etc.; on the other hand, it might undergo signal processing such as smoothing, denoising, etc. – they all result in almost the same content but the underlying binary files can be quite different. In order to cope with this problem, content-based hashing, also known as perceptual or robust hashing, was proposed as a promising solution.

Hashing means to generate fingerprints (digests) from data. In real life, human fingerprints can be compared to tell whether they correspond to the same identity; in the multimedia domain, fingerprints generated by content-based hash algorithms work in a similar way: the same or similar content should result in the same or similar hash value, regardless of the digital representation. Therefore, multimedia identification and authentication can be carried out by hash comparison.

Content-based hash algorithms extract robust features from the content to form the hash value. Besides this feature, ideally they should also be one-way and collision-resistant. The former means, that given a hash output, it is difficult to find out the input; the latter means, that it is difficult to find two inputs which result in the same hash, and given a pair of hash input and output, it is difficult to find another input which results in the same output. Sometimes, it is also desired to make the hash dependent on a secret key, in scenarios such as authentication [1]. Once the key is changed, the hash should drastically change.

It was motivated that higher-order statistics might be suitable features for image hashing [2]. In this paper, the performance of fourth-order cumulants for image hashing is evaluated through an algorithm proposed in [2]. Their robustness against distortion and discrimination in different content are compared with results obtained by using fourth-order moments. The algorithm is further extended by incorporating a secret key.

The rest of the work is organized as follows: in Section 2, the background of higher-order cumulants and the rationale to use them as image features are introduced; then an algorithm based on fourth-order cumulants is given. Experiments are performed to evaluate its robustness and discrimination ability; in Section 3, the algorithm is extended to incorporate a secret key; Section 4 concludes the work.

2. CUMULANTS FOR IMAGE HASHING
The main goal of a content-based hash algorithm is to produce hash output which is tolerant to content-preserving distortion but sensitive to content modification. This somehow vague criterion can be interpreted in various ways.

Considering images, one possible approach is to assume that the relative relationship between pixels in a neighborhood remains approximately the same after authentic dis-
torment, otherwise not. This motivates using joint statistics as image features, for if the pixel sequence in a neighborhood is assumed to be a random process, its auto-statistics characterize the relative relationship between pixels.

In literature, Yu et al. [2] first proposed to use higher-order auto-cumulants as robust image features. The cumulant is a quantity in statistics for measuring deviation from Gaussian [2]. Given a set of variables \( \{x_1, x_2, \cdots, x_n\} \), their joint cumulants of order \( r = k_1 + k_2 + \cdots + k_n \) are defined as [3]

\[
C_{k_1 \cdots k_n} = (-j)^r \left. \frac{\partial^r \ln \Phi(\omega_1, \omega_2, \cdots, \omega_n)}{\partial \omega_1^{k_1} \partial \omega_2^{k_2} \cdots \partial \omega_n^{k_n}} \right|_{\omega_1 = \cdots = \omega_n = 0}
\]

where

\[
\Phi(\omega_1, \omega_2, \cdots, \omega_n) = E\{\exp j(\omega_1 x_1 + \cdots + \omega_n x_n)\}
\]  

is their joint characteristic function and \( E \) means expectation. The first- and second-order cumulants are also known as the mean and the autocorrelation. Orders higher than two are called higher orders. Assuming a zero-mean stationary process \( X(n) \), its second-, third-, and fourth-order cumulants are defined by:

\[
C_{2X}(k) = E\{x(n)x(n + k)\} \quad (3)
\]

\[
C_{3X}(k, l) = E\{x(n)x(n + k)x(n + l)\} \quad (4)
\]

\[
C_{4X}(k, l, m) = E\{x(n)x(n + k)x(n + l)x(n + m)\} - C_{2X}(k)C_{2X}(l - m) - C_{2X}(l)C_{2X}(k - m) - C_{2X}(m)C_{2X}(k - l)
\]  

(5)

There are two important reasons to use higher-order statistics for image hashing: 1) because higher-order statistics vanish for a Gaussian process, they show better resistance to Gaussian noise than lower order statistics; 2) because natural images are known to be non-Gaussian [4], non-Gaussianity might be used to characterize image content, i.e., content-preserving distortion should preserve the non-Gaussianity, while content modification tends to drastically change it. Assuming zero-mean random variables \( X \) and \( Y \) are independent and \( Y \) is Gaussian, the particular advantage of higher-order cumulants is that

\[
C_{n, X + Y} = C_{n, X} + C_{n, Y} = C_{n, X} , \ n \geq 3
\]  

(6)

Therefore, if content-preserving distortion can be modeled as Gaussian, they can be separated by higher-order cumulants. Another advantage of using higher-order cumulants is that the one-way and collision-resistant properties can be well satisfied. In general, due to the high complexity, it is difficult to reconstruct meaningful images from given higher-order statistics.

In [2], an image hash algorithm based on higher-order cumulants was proposed. It extracts image features by dividing an image into blocks, and computing fourth-order cumulants for each block. It works on gray-scale or luminance images, and consists of the following steps:

1. Resize the image to a canonical size of \( 256 \times 256 \);
2. Divide the image into \( 64 \times 64 \) blocks (no overlap);
3. Re-order the pixels into a vector by a raster scan;
4. Compute fourth-order cumulants for the vector;
5. Apply the discrete cosine transform (DCT) to the fourth-order cumulants;
6. The first 32 DCT coefficients are kept and quantized as hash output.

Since fourth-order cumulants are three-dimensional, in order to have a compact hash, they are only computed along a line by setting \( l \) and \( m \) to zero in Equation 5, i.e., the following is supposed to be estimated

\[
C_{4X}(k, 0, 0)
\]  

(7)

The DCT is used to concentrate the energy on the first few coefficients for further compression.

One can see that the essential part of this algorithm is computing cumulants. However, we note that in [2, Equation 5] the cumulants were defined as the first item on the right side of Equation 5, which is actually the definition of fourth-order moments, i.e.,

\[
M_{4X} = E\{x(n)x(n + k)x(n + l)x(n + m)\}
\]  

(8)

Therefore, the results of [2] are actually based on moments, not cumulants! Although they are related concepts, they have different physical meaning, so replacing one with the other might have significant influence on the performance of the algorithm. Interestingly, promising results were shown in [2], in spite that moments were used without adequate motivation. On the other hand, we also note that no other results have been shown in literature for using cumulants in such applications. Therefore, a series of questions arise. For example, why would moments work? Would cumulants work as well? If so, moments and cumulants, which are better for image hashing?

In order to gain insights into above questions, we first investigate whether cumulants are suitable for image hashing. In the following, we perform some experiments using the algorithm in [2] with both fourth-order cumulants (Equation 5) and moments (Equation 8). Robustness against distortion and discrimination in different content will be evaluated. The results of the two algorithms will be compared and discussed. For clarity, we call them the cumulant algorithm and the moment algorithm, respectively.

First, we apply the following manipulation to some typical images including Lena and Boat, and compare the resultant hash with the original one to test the robustness.

1. Rotation
2. Cropping
3. Gaussian noise
4. Salt & pepper
5. Shearing
6. Scaling
7. JPEG compression
8. Sharpening
9. Gaussian filtering
10. Median filtering

Since images distorted by these operations are normally still considered as authentic, their hashes should be similar.
to original ones. In our experiments, the same algorithm structure is used as listed before, with slight modification: 1) a canonical image size of $512 \times 512$ is used to have a better resolution; 2) the mean of a block is subtracted before cumulant or moment computation to meet the zero-mean assumption and improve the robustness; 3) quantization is skipped to have more accurate feature vectors. Hash comparison is done by correlation. Assuming $H_1$ and $H_2$ are two hash vectors, the normalized correlation is used to evaluate their similarity, which is defined as

$$\frac{|H_1 \cdot H_2|}{\|H_1\|_2 \cdot \|H_2\|_2}. \quad (9)$$

Recall that similar content should result in similar hashes. If two images contain similar content, the normalized correlation between their hash vectors should approach 1, otherwise approach 0.

The results of normalized correlation by the cumulant algorithm are listed in Table 1. As a comparison, the results by the moment algorithm are listed in Table 2. Most results in the two tables are quite similar, especially for non-geometric distortion. If quantization is further applied, the difference might be even smaller. For some geometric distortion (rotation, cropping, shearing), the results are less stable, due to the fact that there is no particular design for the algorithm to resist such distortion (both algorithms fail for large geometric distortion). Since most results are close to unity, it is difficult to tell at this point which algorithm possesses stronger robustness. It seems that cumulants work as well as moments.

Besides robustness to content-preserving distortion, a proper hash algorithm should also be able to distinguish different content. It is still unknown whether the cumulant algorithm produces hash vectors that are almost unique for different content. Therefore, we compare hash vectors between four different images by normalized correlation. Ideally, the correlation between hashes from different images should be as low as possible. The results are listed in Table 3. For comparison, the results by the moment algorithm are listed in Table 4. It is clear that the cumulant algorithm results in lower hash correlation between different images. Therefore, the cumulant algorithm may have better discrimination ability.

From above observation, we conjecture that the moment algorithm might have stronger robustness than the cumulant algorithm, since robustness and discrimination usually reflect each other. If one is strong, the other might...
be weak, and vice versa. This is consistent with the definition of moments and cumulants (Equation 8 and Equation 5). Since moments “contain” cumulants, they contain more energy, which might be exploited to achieve robustness. However, stronger robustness is not obvious in our experiments. This might be due to the fact that Equation 6 does not hold for moments. Although moments contain more energy, they are also more vulnerable to noise. When the noise becomes significant, the results might be worsened and become unstable. Therefore, we conclude that the cumulant algorithm is better than the moment algorithm in discrimination and has similar performance in robustness.

3. SECURITY EXTENSION

For some scenarios such as authentication, it is desired to make the hash dependent on a secret key [1]. Once the key is changed by even a bit, the hash output should drastically change. However, neither the cumulant algorithm nor the moment algorithm supports this. In the following, we show how to extend them to incorporate a secret key.

Recall that Equation 7 is estimated for all blocks. That means, only part of the non-Gaussianity is exploited. A good way to introduce security is to choose \( l \) and \( m \) randomly in Equation 5 (for the cumulant algorithm) or 8 (for the moment algorithm) for each block, i.e., instead of estimating cumulants or moments along a fixed line in a three-dimensional space, one could estimate along arbitrary lines. Specifically, we can use a secure pseudo-random number generator (PRNG) and make \( l \) and \( m \) dependent on the pseudo-random output. The secure PRNG should accept a secret key, thus the output is sensitive to any key bit change. We simulate this with the cumulant algorithm by choosing \( l \) and \( m \) randomly in the range 0-1000 for each block, and compare the resultant hashes with the one obtained by setting \( l = m = 0 \). The results of normalized correlation are listed in Table 5 for ten rounds. In most cases, the correlations are quite low.

In practice, if the change in \( l \) or \( m \) is too small, the hash output might be similar. Empirically we find that a minimum change of about 10 in \( l \) or \( m \) for each block is necessary to make a distinct hash output. Assuming each of the 64 blocks can have \( n \) distinct statistic patterns by choosing different \( l \) and \( m \), the size of search space for brute force attack is on the order of \( n^{64} \). For even higher security, the computation of cumulants or moments does not need to follow a straight line, i.e., we can take any curve in the three-dimensional space for more choices of statistic patterns. However, the implementation is less efficient. Trade-offs have to be made between security and efficiency.

4. CONCLUSIONS AND DISCUSSIONS

We have applied fourth-order cumulants in an image hash algorithm. Our experiments show that fourth-order cumulants can be used for image hashing and they have better overall performance than fourth-order moments. Since moments contain cumulants, this might explain why moments can work as in [2]. We have extended the algorithm by incorporating a secret key. It is achieved by choosing different statistic patterns for different image blocks according to the output of a secure pseudo-random number generator. We have performed simulation to show that a different key can result in a different hash. These conclusions will be verified by more extensive experiments in the future.

Some topics might be interesting for future research: 1) our approach of image hashing can be extended to even higher orders. However, it is unknown whether it is necessary to use orders higher than four, since the computation cost will increase; 2) the influence of key-based randomization on robustness and discrimination should be investigated; 3) it is possible to choose the most representative cumulants or moments within the three-dimensional space. However, the computation is high.

5. REFERENCES


Table 5. Normalized correlation between hashes by the cumulant algorithm with different keys.

<table>
<thead>
<tr>
<th></th>
<th>Lena</th>
<th>Boat</th>
<th>Pepper</th>
<th>Elaine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1</td>
<td>.398</td>
<td>.609</td>
<td>.800</td>
<td>.435</td>
</tr>
<tr>
<td>Round 2</td>
<td>.612</td>
<td>.670</td>
<td>.565</td>
<td>.438</td>
</tr>
<tr>
<td>Round 3</td>
<td>.537</td>
<td>.334</td>
<td>.653</td>
<td>.354</td>
</tr>
<tr>
<td>Round 4</td>
<td>.422</td>
<td>.433</td>
<td>.601</td>
<td>.605</td>
</tr>
<tr>
<td>Round 5</td>
<td>.557</td>
<td>.050</td>
<td>.848</td>
<td>.480</td>
</tr>
<tr>
<td>Round 6</td>
<td>.586</td>
<td>.010</td>
<td>.695</td>
<td>.431</td>
</tr>
<tr>
<td>Round 7</td>
<td>.648</td>
<td>.528</td>
<td>.789</td>
<td>.561</td>
</tr>
<tr>
<td>Round 8</td>
<td>.411</td>
<td>.126</td>
<td>.667</td>
<td>.738</td>
</tr>
<tr>
<td>Round 9</td>
<td>.745</td>
<td>.744</td>
<td>.838</td>
<td>.429</td>
</tr>
<tr>
<td>Round 10</td>
<td>.467</td>
<td>.525</td>
<td>.617</td>
<td>.449</td>
</tr>
</tbody>
</table>