Forensic Science International 206 (2011) 178–184

Contents lists available at ScienceDirect

Forensic Science International

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

Improved DCT-based detection of copy-move forgery in images

Yanping Huang [a], Wei Lu [a,*], Wei Sun [b], Dongyang Long [a]

[a] School of Information Science and Technology, Guangdong Key Laboratory of Information Security Technology, Sun Yat-sen University, Guangzhou 510275, China
[b] School of Software, Sun Yat-sen University, Guangzhou 510275, China

ARTICLE INFO

Article history:
Received 17 December 2009
Received in revised form 1 August 2010
Accepted 4 August 2010
Available online 15 September 2010

Keywords:
Copy-move forgery
Digital image forensics
Duplicated region detection

ABSTRACT

Techniques for digital image tampering are becoming more and more sophisticated and widespread. Copy-move forgery is one of the tampering techniques that are frequently used. In this paper, an improved DCT-based method is developed to detect this specific artifact. Firstly, the image is divided into fixed-size overlapping blocks and, DCT is applied to each block to represent its features. Truncating is employed to reduce the dimension of the features. Then the feature vectors are lexicographically sorted and, duplicated image blocks will be neighboring in the sorted list. Thus duplicated image blocks will be compared in the matching step. To make the method more robust, a scheme to judge whether two feature vectors are similar is imported. Experiment results demonstrated that the proposed method can detect the duplicated regions even when an image was distorted by JPEG compression, blurring or additive white Gaussian noise.

© 2010 Elsevier Ireland Ltd. All rights reserved.

1. Introduction

With the development of the sophisticated techniques for digital image tampering and the low cost to get a high quality digital image, anyone can doctor an image easily without leaving visible clues. As a result, digital images can no longer be trusted and they will not hold the unique stature as a definitive record of an event. In order to recover people's confidence towards the authenticity of the digital image, it is necessary to develop a set of methods to authenticate digital images.

Digital watermarking has already been used in this area and a lot of researches have been done (for a general survey, see [1]). Although there are several kinds of authentication watermarking schemes, such as erasable fragile watermarking [2], semi-fragile watermarking [3] and robust tell-tale watermarking [4], all of these schemes assume that the digital watermark embedded in digital images would be altered after digital tampering, thus this requires that watermarks are inserted into digital images when they are taken. This requirement is the main drawback of these schemes, since only specialized expensive digital cameras embed the watermark in the image when a photo is taken.

On the contrary, there are some other methods which are called blind detection techniques that do not need the presence of digital watermarks. These approaches work by assuming that any form of tampering would alter the underlying statistics of an image. There are some classic processes used in the tampering and one of them is the copy-move process. A copy-move process is usually used to conceal a special object in the original image. To do so, one region in an image is copied and pasted onto another region in the same image. Although some post process, like edge smoothing, blurring and noise adding, would be taken to remove the visible clues, the process has created two almost identical parts in the tampered image. This is not common in natural images and thus can be used to detect these kinds of images.

Some copy-move detecting methods have been developed. In [5], Fridrich first analyzed the exhaustive search and then proposed a block matching detecting method based on discrete cosine transform (DCT) which is more effective than exhaustive search. Popescu proposed a similar method [6], which used principal component analysis (PCA) instead of DCT. Utilizing one of the characteristics of PCA, Popescu used about half of numbers of the features used by Fridrich and, by doing so, his method is demonstrated to be more effective. But the method also has its drawbacks. One of them is that it does not work when the copy region is slightly rotated before it is pasted. In contrast, Fridrich's method can detect a copy-move tampering with rotation up to 5° [7]. Recently, some methods were presented that can detect more sophisticated post-processing, like Mahdian's work [8] can detect the duplicated region even the copy-move region been blurred or added noise.

In this paper, we propose a detection algorithm based on DCT. Because the ideas which appear in Popescu's technique [6], such as dimension reduction and matching control, are introduced into the algorithm, the algorithm's framework is like Popescu's. With DCT coefficients as features, the algorithm can do a bit better in the case

* Corresponding author. Tel.: +86 20 39332640.
E-mail addresses: hypxp0083@yahoo.com (Y. Huang), luwei3@mail.sysu.edu.cn (W. Lu), sunwei@mail.sysu.edu.cn (W. Sun), issldy@mail.sysu.edu.cn (D. Long).

0379-0738/$ – see front matter © 2010 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.forsciint.2010.08.001

of detecting copy-move tampering with rotation compared to Popescu's technique. In comparison with Fridrich's technique [5] our algorithm will be quicker due to dimension reduction, while the detecting ability is not reduced. The robustness of JPEG compression with different quality factors, Gaussian blurring and additive white Gaussian noise is demonstrated by a series of experiments, and these experiments are absent in [5]. The rest of the paper is organized as follows: in Section 2 the proposed forgery detection method is described. The experiment results are shown in Section 3. The conclusion is drawn in Section 4.

2. The proposed algorithm

Due to the nature of region-duplication, there must be at least two similar regions in the tampered image. A natural image, on the contrary, is very unlikely to have two large similar regions (except for the images that have two large smooth regions). Thus an image can be considered as copy-move tampered if two large similar regions are detected. Here we assume that the duplicated regions are non-overlapping. The task of the detecting method is to determine whether an image contains duplicated regions. Since the shape and size of the regions are unknown, it is definitely computationally impossible to try to examine every possible pairs of region with different shape and size. It is more effective to divide an image into fixed-sized overlapping blocks and test whether pairs of blocks are duplicated.

Before the test, another important step, to represent the blocks by features, must be done. A good feature extraction method can make the detecting algorithm effective and robust. After all blocks are represented properly by some features, the algorithm continues into the matching step. As the similar regions would have similar features, if the features are sorted in a lexicographically order before matching, it can make the similar regions consecutive and make matching more effective. Each pairs of features is then tested whether they are matching. In the case of a pair of features are matching, the corresponding blocks are assumed to be similar. If many of those blocks are adjacent, a copy-move tampering may have occurred in the input image. The detected duplicated regions can be marked in a map image. The operator can judge whether an image has been copy-move tampered according to the map image.

2.1. Algorithm framework

The discussions above lead to the framework of the copy-move detecting algorithm, which is also shown in Fig. 1.

Step 1—*Division*: Dividing the input image into sub-blocks.
Step 2—*Representation*: Representing every sub-block using proper features.
Step 3—*Matching*: Testing each pair of sub-block whether they are similar.
Step 4—*Output*: Outputing the duplicated regions (if exist) visually.

The following is how the algorithm is implemented in the framework. A square with $B \times B$ pixels slides once along the image from the upper left corner right down to the lower right corner. Suppose the image has $M \times N$ pixels, the sliding will generate $(M-B+1)(N-B+1)$ blocks.

DCT is applied to every blocks to extract the features. The DCT coefficients is reshaped into a row vector in the zigzag order thus similar frequencies are grouped together and, $(M-B+1)(N-B+1)$ row vectors are generated when features of all blocks are extracted. Then the vectors are sorted in a lexicographically order. Let the matrix A denote the sorted vectors, the size of A will be $(M-B+1)(N-B+1) \times B^2$.

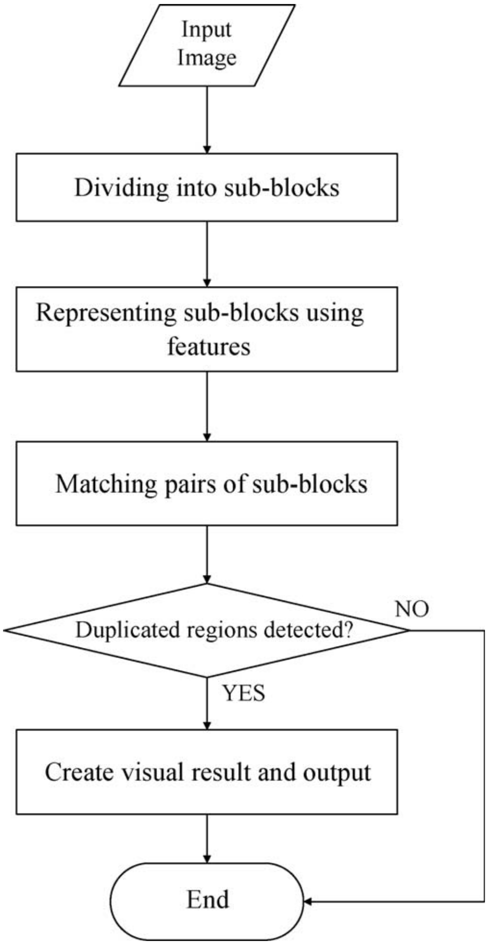

Input Image
Dividing into sub-blocks
Representing sub-blocks using features
Matching pairs of sub-blocks
Duplicated regions detected?
NO
YES
Create visual result and output
End

Fig. 1. Shown is the algorithm framework of copy-move forgery detection.

The ith row of A is noted as $\vec{a}_i$, and the top-left corner's coordinate of the corresponding block is noted as (x_i, y_i). In the matching step, each pairs of consecutive vector $\vec{a}_i, \vec{a}_j$ is tested whether they are similar. If they are, the shift vector s between the two corresponding blocks is calculated as:

$$s = (s_1, s_2) = (x_i - x_j, y_i - y_j) \tag{1}$$

Note that s and $-s$ correspond to the same shift, so multiplying s by -1 may be necessary to make sure that $s_1 \geq 0$. It can be called a normalization. Then the algorithm scans all the shift vector s, and looks for vectors $s_{(1)}, s_{(2)}, \ldots, s_{(n)}$ whose occurrence exceeds a predefined threshold T.

For every $s_{(i)}$, all the matching blocks that contributed to this shift vectors are found and colored. This algorithm outputs a black map image, regions which are considered to be duplicated are marked with a special color other than black. With the map image and human visual judgement, it can be quickly determined whether the input image is original or tampered.

2.2. Feature dimension reduction

In the feature representing step, DCT is applied to the $B \times B$ block and generate a feature vector with dimension of B^2. It is the nature of DCT that the energy of transformed coefficients will be focused on the first several values (the lower frequency coefficients). Thus, those higher frequency coefficients can be truncated. Since the DCT coefficients have been reshaped to a row vector in zigzag order, truncating can easily be done by saving only a part of vector components. Specifically, a factor $p(0 < p \leq 1)$ is defined and only first $\lceil p \times B^2 \rceil$ DCT coefficients are saved for further processing. The dimension reduction can make the sorting and matching faster. To make our algorithm more robust, a quantization factor q is also employed. Each DCT coefficients will be quantized by q and then rounded to the nearest integer.

2.3. Matching

In the matching step, for each row $\vec{a}_i$ in the matrix A, every $\vec{a}_j$ that satisfies $j - i < N_f$ is tested whether it is similar with $\vec{a}_i$, where N_f is a parameter that controls the amount of neighboring rows that tested to find the similarity. Though the DCT coefficients have been quantized and truncated, a normal method that consider two vectors equal by each corresponding component equal may still lead to robustness degradation. So the following method is employed to judge similar vectors. The method is based on a fact that if two vectors are correlative, the ratio of each corresponding components of two vectors is equal. It can, therefore, be inferred that if two vectors are similar each other, then the ratio of each corresponding components of these two vectors will be very close. So to judge whether two vectors are similar, the ratio of each corresponding components of two vectors should be calculated respectively and then, tested to see whether these ratios are close enough. In the case that the vector component is 0 and cannot be the divider, the test is switched to checking whether the pair of corresponding component is close enough.

Here is an example to judge whether $\vec{a}_i = (a_i^1, a_i^2, \ldots, a_i^k)$ is similar with $\vec{a}_j = (a_j^1, a_j^2, \ldots, a_j^k)$. Two parameters, *s _ threshold* and *t _ threshold* are defined and two variables, *maxr* and *minr* which would record the minimum and the maximum ratio, are initialize to a very small number and a very large number, respectively. For each $l(1 \leq l \leq k)$, if $a_j^l = 0$, check whether $abs(a_j^l - a_i^l) < s\_threshold$. If the constraint not holds, $\vec{a}_i$ and $\vec{a}_j$ are not similar. In the case that $a_j^l \neq 0$, calculate the ratio $r_l = a_i^l / a_j^l$ and update *minr*, *maxr* as such if $minr > r_l$, $minr = r_l$; if $maxr < r_l$, $maxr = r_l$. Then check whether $maxr - minr > t\_threshold \times l$. If it is, $\vec{a}_i$ and $\vec{a}_j$ are not similar. If all the constraints hold for every l, $\vec{a}_i$ and $\vec{a}_j$ are considered to be similar.

Since the duplicated regions are assumed to be not overlapping and the divided blocks may be overlapping, an additional judgement have to be made when a matching is detected, that is, only the shift vectors generated by those blocks which are not too close are counted. Thus, another parameter N_d is also imported in this step. Shift vector is counted if and only if it is generated by two similar feature vectors whose distance is larger than N_d.

2.4. Post-processing

When there are shift vectors $s_{(i)}$ whose appear frequency exceeds threshold T, a map image is initialized with the same size as the input image and all the pixels are 0. For each $s_{(i)}$, the matching blocks that contributed to this shift vector are marked in two special colors in the map, one color at the source, and the other at the destination. There are maybe some false match because we truncate the quantized DCT coefficient for dimension reduction, so a morphology operation, the opening operation [9] is taken to remove the isolated blocks.

2.5. Algorithm flow

The details of the copy-move detection algorithm are described as follows.

(1) The input image is a gray image I of the size $M \times N$. If it is a color image, it can be converted to a grayscale image using the standard formula $I = 0.299R + 0.587G + 0.114B$.

(2) A fixed-sized $B \times B$ window is slide one pixel along from the upper left corner to the button right, divide I into $(M - B + 1)(N - B + 1)$ blocks.

(3) For each blocks, applied the DCT and reshape the $B \times B$ quantized coefficient matrix with a zigzag scan to a row vector. The vector is then truncated to only $\lceil p \times B^2 \rceil$ elements.

(4) All vectors are sorted lexicographically and form a $(M - B + 1)(N - B + 1) \times \lceil pB^2 \rceil$ matrix A.

(5) For each row a_i in A, test its neighboring rows a_j which satisfy $j - i < N_f$ that whether they are similar.

(6) If a_i and a_j come out to be similar, the distance between two corresponding blocks $d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is calculated.

(7) If $d > N_d$, the shift vector s is calculated and normalized. Then the shift vector's existing frequency plus 1.

(8) If there are $s_{(i)}$ which existing frequency over threshold T, go to the next step, otherwise the algorithm exit.

(9) For each $s_{(i)} > T$ mark the regions in the map image, then use the opening operation to remove isolated regions. The algorithm output the map image together with the input image.

3. Experiment results

In this section, we first introduce the experiment procedure and parameters and then, discuss the results to prove the robustness and sensitivity of our algorithm.

3.1. Experiment methods and procedure

The experiments were carried out on the Matlab R2007a. All images used in the experiment were in the data set [10]. All the images were 128 × 128 pixels gray image saved in BMP format. A 2G duo core processor was adopted and it took about 4 s to check one image. In our experiments, without specific specification, all the parameter in the experiment were set as: $B = 8$, $T = 35$, $N_f = 3$, $N_d = 16$, $p = 0.25$, $q = 4$, *s _ threshold* = 4, *t _ threshold* = 0.0625 by default. In the case that image had been heavily distorted and the default parameters not well worked, the specific parameters we used would be given. A 10 × 10 square was used in the opening operation.

If an input image was natural, while our method had detected some duplicated regions (i.e., false alarm), or the input image was tampered, while our method failed to detect any duplicated regions (i.e., miss alarm), W would be set to 1, otherwise $W = 0$. Two counters, *au _ cm* and *cm _ au* which were initialized to 0, were also plus 1 when wrong detection or failure in detecting forgery happened, respectively. After the whole testing data set had been processed, false alarm rate *far*, miss alarm rate *mar* and classification accuracy rate *car* were calculated as the following formulas:

$$car = 1 - \frac{W}{pics}, \qquad far = \frac{au\_cm}{pics}, \qquad mar = \frac{cm\_au}{pics} \tag{2}$$

where *pics* was the amount of images we used. These parameters could be used to evaluate sensitivity of this algorithm.

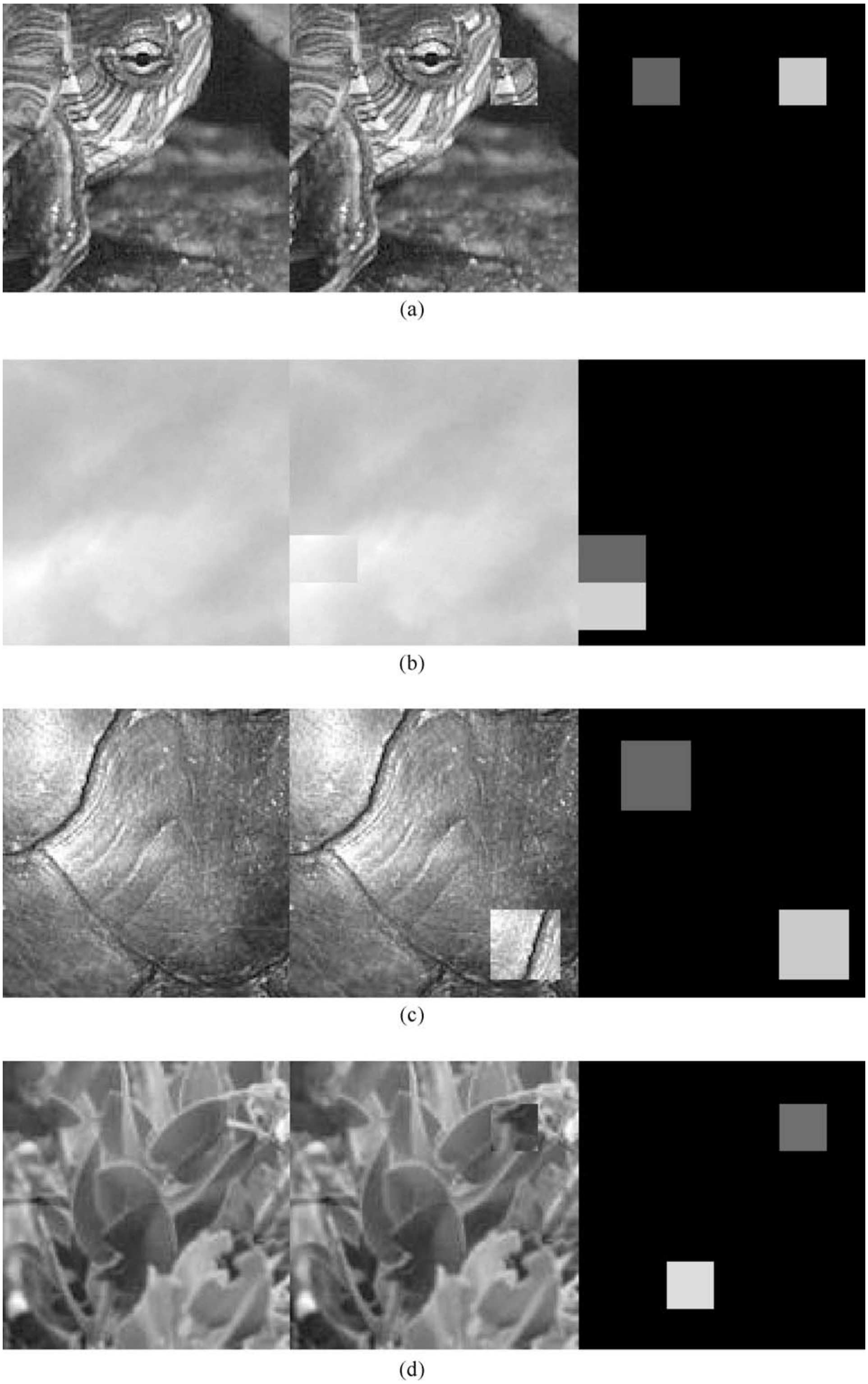
The non-overlapping duplicated regions in a tampered image were noted as D_1 and D_2, while the duplicated regions of an image detected by using our method were noted as R_1 and R_2. If the input image was tampered and our method had detected some duplicated regions, the accuracy p and false negative f would be calculated as the following formulas:

$$p = \frac{|D_1 \cap R_1| + |D_2 \cap R_2|}{|D_1| + |D_2|}, \qquad f = \frac{|D_1 \cup R_1| + |D_2 \cup R_2|}{|D_1| + |D_2|} - p \tag{3}$$

where $|\cdot|$ means the area of region, $\cap$ meant the intersection of two regions and $\cup$ meant the union of two regions. In the formula, $D_1 \cap R_1$ and $D_2 \cap R_2$ were the overlapping region of the method's detecting regions and the regions where copy-move tampering was really made, and $D_1 \cup R_1$ and $D_2 \cup R_2$ were the regions that included the method's detecting regions and the region where copy-move tampering is really made. These two parameters indicated how precisely our method could locate the duplicated regions. The more p was close to 1 and f was close to 0 the more precise the method would be.

3.2. Visual results

The images shown in Fig. 2 were the results of detecting tampered images without any distort operations. Each image was composed of three images: original image, tampered image and map image from left to right. Four images indicated four possible positions of duplicated regions: horizon, vertical, diagonal and


(a)

(b)

(c)

(d)

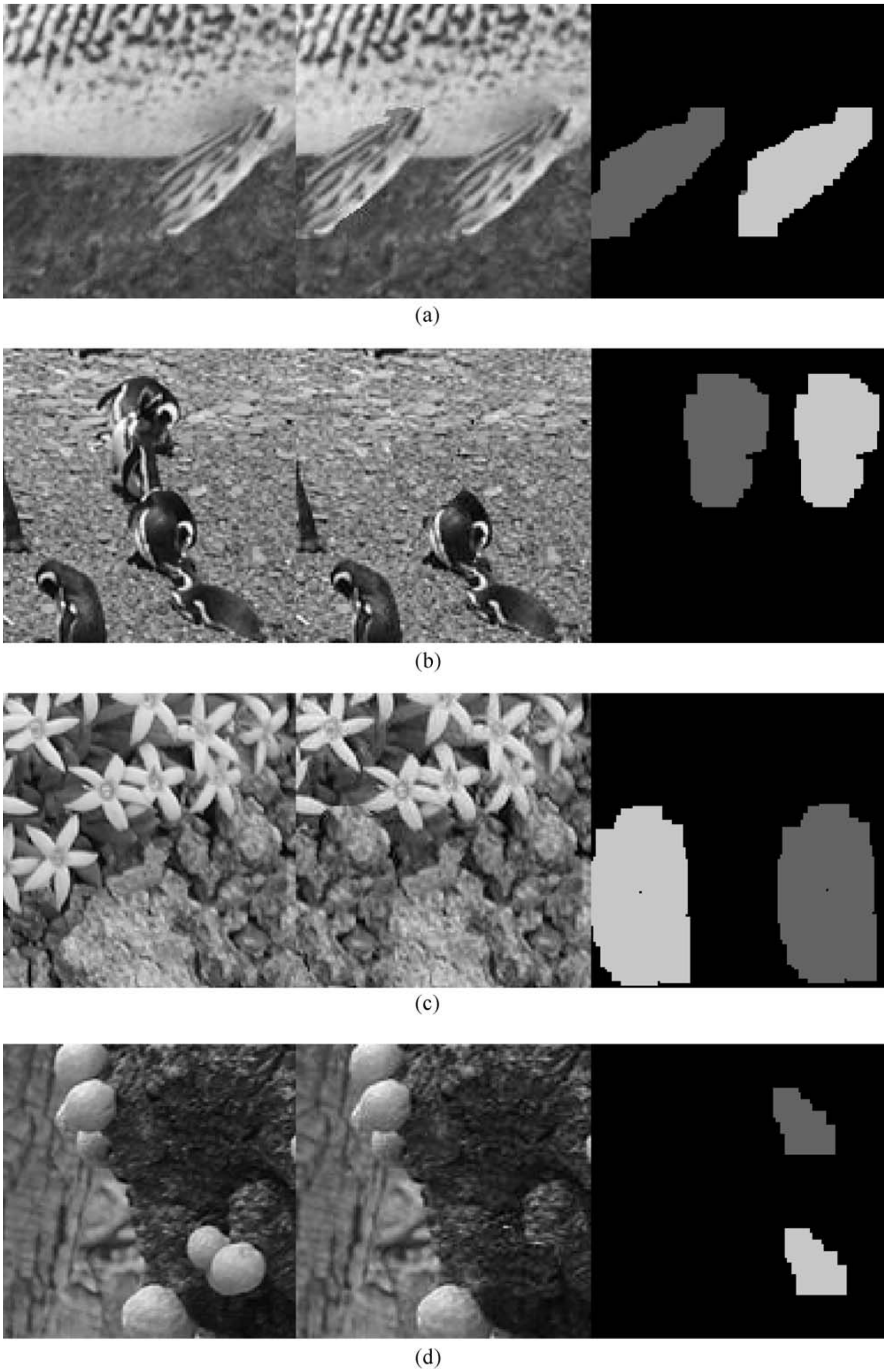
Fig. 2. Shown are the different possible positions of duplicated regions in (a) horizontal, (b) vertical, (c) diagonal, and (d) antidiagonal directions.

antidiagonal. Our algorithm could detect all the cases precisely. It was noted that Fig. 1 also indicated that the algorithm could process some images having large and similar regions in visual sense.

Some more tampered images with non-regular profile duplicated regions, which often happened in the copy-move forgery, were tested. Tampered images were saved in JPEG format with quality factor 70. In these examples, all parameters were set as: $B = 8$, $T = 35$, $N_f = 3$, $N_d = 16$, $p = 0.25$, $q = 4$, $s\_threshold = 8$, $t\_threshold = 0.3$. Images shown in Fig. 3 were the testing results which indicated that our algorithm worked well even the tampered images had been JPEG compressed.

3.3. Sensitivity and precision test

We used some data sets to test the sensitivity and precision of the algorithm. The purpose of the algorithm sensitivity test was to determine whether the algorithm could detect the copy-move


(a)

(b)

(c)

(d)

Fig. 3. Shown are the detecting results for JPEG compressed tampered images ($Q = 70$).

Table 1
Detection results of the proposed method, where *far* is the false alarm rate, *mar* is the miss alarm rate, and *car* is the classification accuracy rate, which are defined in Eq. (2).

	BMP	JPEG (Q = 90)	JPEG (Q = 80)	JPEG (Q = 70)	Blur	AWGN
far	0.045	0.18	0.18	0.195	0.07	0.07
mar	0	0	0.03	0.07	0.005	0.065
car	0.978	0.91	0.895	0.868	0.963	0.933

tampering even when an image was distorted. Also, it would not have any detecting result if an image had not been tampered. Precision of the algorithm meant that if a tampered image was detected, how precisely the algorithm could locate the tampering regions.

We used 200 natural images to make 200 tampered images by copying a square region at a random location and pasting onto a non-overlapping region. The square region's size was also randomly pickup from the set R = {16 × 16, 24 × 24, 32 × 32, 40 × 40, 48 × 48}. The tampered images together with their original version were then distorted by different processing operations: JPEG compression with different quality, Gaussian blurring (window size was 3 × 3 and $\sigma = 1$), additive white Gaussian noise (AWGN) with power = 4 dWB (in blurring and noise test, square region's size is fixed to 48). In experiments of JPEG compression and AWGN distortions, all parameters were set as: $B = 8$, $T = 35$, $N_f = 3$, $N_d = 16$, $p = 0.25$, $q = 4$, $s\_threshold = 8$ and $t\_threshold = 0.3$. These images were used to test the sensitivity of the method. The results were shown in Table 1. The classification accuracy rate(*car*) was higher then 85% in any cases, which demonstrated that the method was sensitive to detect copy-move tampering even the tampered image is distorted by some post-processing.

To test the precision of our method, 150 other natural images in the data set were used. For each image, a fixed-size region was copied and then pasted onto a random non-overlapping position. The size was set as 16 × 16, 24 × 24, 32 × 32, 40 × 40 and 48 × 48, respectively. The results were shown in Table 2, where $\bar{f}$ was the average rate of f, $\bar{p}$ was the average rate of p (f and p were defined in Eq. (3)) and $\bar{W}$ is the wrong classification rate. Except the case of 16 × 16 where the algorithm failed to detect two tampering images, but it detected all other tampering images. In all cases the $\bar{p}$ was very close to 1 and the $\bar{f}$ was very close to 0. This demonstrated that our method was sensitive and precise in the case of detecting normal copy-move forgery subject not to post-processing.

To test the precision of our method in the case of detecting JPEG compress image, tampered images were distorted by JPEG compression with different quality factor (90, 80 and 50). In this part, the square region's size started from 24 × 24, and the parameters were set as: $B = 8$, $T = 35$, $N_f = 3$, $N_d = 16$, $p = 0.25$, $q = 4$, $s\_threshold = 8$, $t\_threshold = 0.3$. The results shown in Table 3 indicated that the method did work well when the images were distorted by slight compression. When the image was highly compressed the precision declined as $\bar{p}$ decreased, while $\bar{f}$ increased. In such case the method was still sensitive to detect copy-move tampering when the duplicated regions were not too small.

Table 2
Detection results of the forgery images without any post-processing, where f and p are defined in Eq. (3), and $\bar{W}$ is the wrong classification rate.

Image size	16 × 16	24 × 24	32 × 32	40 × 40	48 × 48
$\bar{p}$	1	0.998	0.994	0.996	0.999
$\bar{f}$	0.027	0.012	0.011	0.008	0.004
$\bar{W}$	0.013	0	0	0	0

Table 3
Detection results of the forgery images distorted by JPEG compression.

	Image size	24 × 24	32 × 32	40 × 40	48 × 48
Q = 90	$\bar{p}$	0.980	0.988	0.991	0.996
	$\bar{f}$	0.034	0.029	0.017	0.009
	$\bar{W}$	0.006	0	0	0
Q = 80	$\bar{p}$	0.942	0.961	0.972	0.98
	$\bar{f}$	0.091	0.055	0.04	0.027
	$\bar{W}$	0	0.006	0.006	0
Q = 50	$\bar{p}$	0.760	0.716	0.714	0.729
	$\bar{f}$	0.345	0.307	0.298	0.277
	$\bar{W}$	0.613	0.16	0.003	0.003

Table 4
Detection results of the forgery images distorted by Gaussian blurring.

	$w = 3, \sigma = 0.5$	$w = 3, \sigma = 1$	$w = 5, \sigma = 0.5$	$w = 5, \sigma = 1$
$\bar{p}$	0.922	0.915	0.888	0.86
$\bar{f}$	0.091	0.089	0.162	0.162
$\bar{W}$	0	0.013	0	0

Table 5
Detection results of the forgery images distorted by additive white Gaussian noise.

	Power = 1 dWB	Power = 2 dWB	Power = 4 dWB
$\bar{p}$	0.946	0.924	0.864
$\bar{f}$	0.057	0.077	0.137
$\bar{W}$	0	0	0

Experiments on images that distorted by blur and additive white Gaussian noise were also performed to test how precise our method was in these two cases. The square region's size was fixed as 48 × 48 in this part and the parameters were set as: $B = 8$, $T = 35$, $N_f = 3$, $N_d = 16$, $p = 0.25$, $q = 4$, $s\_threshold = 8$, $t\_threshold = 0.3$ for AWGN distortion, while default parameters were used for blurred images. Results of tampered images distorted with different Gaussian blurring were shown in Table 4. The method failed to detect only three images in a total of 600 testing images. The values of $\bar{p}$ and $\bar{f}$ had, therefore, indicated that the algorithm had the ability to locate tampering regions in the case of blurring. Table 5 shows the results of tampered images distorted by AWGN with different power. These results had indicated that the method performed well also in the case of processing AWGN distorted image. The algorithm had detected all tampering images and the detecting precision was good.

4. Conclusions

We have presented an automatic duplication image region detection algorithm based on DCT. It works in the absence of digital watermarking and does not need any prior information about the tested image. Compare with previous works, our algorithm used less features to represent each blocks, and was more effective. The experiment results show that the proposed method's ability to detect copy-move forgery in an image is quite robust to JPEG compression, blurring or AWGN distortion. Thus, we believe our method could be useful in some areas of forensic science.

Acknowledgements

This work was supported by NSFC (no. 60803136), Guangzhou Science and Technology Program (no. 2009J1-C541-2).

References

[1] S. Katzenbeisser, F. Petitcolas, Information Techniques for Steganography and Digital Watermarking, Artec House, 2000.

[2] J. Fridrich, M. Goljan, M. Du, Invertible authentication, in: Proc. SPIE, Security and Watermarking of Multimedia Contents, vol. 4313, 2001, 197–208.

[3] E. Lin, C. Podilchuk, E. Delp, Detection of image alterations using semi-fragile watermarks, in: Proc. SPIE, Security and Watermarking of Multimedia Contens II, vol. 3971, 2000, 52–163.

[4] D. Kundur, D. Hatzinakos, Digital watermarking for tell-tale tamper proofing and authentication, Proc. IEEE 87 (7) (1999) 1167–1180.

[5] J. Fridrich, D. Soukalm, J. Lukáš, Detection of copy-move forgery in digital images, in: Digital Forensic Research Workshop, Cleveland, OH, (2003), pp. 19–23.

[6] A.C. Popescu, H. Farid, Exposing Digital Forgeries By Detecting Duplicated Image Regions, Tech. Rep. TR2004-515, Dartmouth College, 2004.

[7] S. Bayram, H.T. Sencar, N. Memon, An efficient and robust method for detecting copy-move forgery, in: IEEE Int. Conf. Acoustics, Speech, and Signal Processing, 2009, 1053–1056.

[8] B. Mahdian, S. Saic, Detection of copy-move forgery using a method based on blur moment invariants, For. Sci. Int. 107 (2007) 180–189.

[9] R.C. Gonzalez, R.E. Woods, Digital Image Processing, 2nd edition, Addison-Wesley, 1992.

[10] T.-T. Ng, J. Hsu, S.-F. Chang, Columbia Image Splicing Detection Evaluation Dataset, http://www.ee.columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet/AuthSplicedDataSet.htm/.