Hindawi
Security and Communication Networks
Volume 2018, Article ID 1301290, 11 pages
https://doi.org/10.1155/2018/1301290

Research Article

Nonoverlapping Blocks Based Copy-Move Forgery Detection

Yu Sun, Rongrong Ni, and Yao Zhao

Institute of Information Science and Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing Jiaotong University, Beijing 100044, China

Correspondence should be addressed to Rongrong Ni; rrni@bjtu.edu.cn

Received 29 September 2017; Revised 12 December 2017; Accepted 18 December 2017; Published 22 January 2018

Academic Editor: Zhenxing Qian

In order to solve the problem of high computational complexity in block-based methods for copy-move forgery detection, we divide image into texture part and smooth part to deal with them separately. Keypoints are extracted and matched in texture regions. Instead of using all the overlapping blocks, we use nonoverlapping blocks as candidates in smooth regions. Clustering blocks with similar color into a group can be regarded as a preprocessing operation. To avoid mismatching due to misalignment, we update candidate blocks by registration before projecting them into hash space. In this way, we can reduce computational complexity and improve the accuracy of matching at the same time. Experimental results show that the proposed method achieves better performance via comparing with the state-of-the-art copy-move forgery detection algorithms and exhibits robustness against JPEG compression, rotation, and scaling.

1. Introduction

With the development of computer technique, there are more and more image editing tools, such as Photoshop and Fireworks. As a result, it becomes much easier for people even nonprofessionals to do some operations on digital images. However, it also brings convenience for people to tamper maliciously. Once these tampered images are applied to court forensics, newspaper, or academic research, the social credibility crisis will be aroused. Therefore, the image forgery detection becomes necessary. There are many types of tampering operations, and copy-move is one of the most common operations among them. The copy-move forgery is to copy a region and paste it into another place in the same image.

In the last few years, a large number of methods have been proposed to detect copy-move forgery that are mainly concentrated on two categories. One is based on block feature, and the other is based on keypoint feature. The block-based methods usually first divide the image into overlapping blocks, then extract each block's feature, and finally find the duplicate regions after matching them. Fridrich et al. [1] first proposed the copy-move forgery detection (CMFD) method by using the coefficients of Discrete Cosine Transform (DCT), and they applied dictionary sorting to the matching process. However, the computing complexity of this algorithm is very high. As a result, many dimensionality reduction-based algorithms have been proposed. Popescu and Farid [2] reduced the feature dimensions via principal component analysis (PCA). Bashar et al. [3] improved the performance further by Kernel-PCA (KPCA). Li et al. [4] combined discrete wavelet transform (DWT) and Singular Values Decomposition (SVD) to extract block feature. Kang and Wei [5] extracted the singular values of a reduced-rank approximation (SVD) as the block feature. Bayram et al. [6] computed the Fourier-Mellin Transform (FMT) for each block, and this method had high robustness in rotation and scaling. Mahdian and Saic [7] proposed a method based on blur moment invariants (Blur). Wang et al. [8] applied Hu moments (Hu) to each block to extract features. Ryu et al. [9, 10] used Zernike moments (Zernike) as block feature, which was only robust in rotation. Luo et al. [11] used the average of red, green, and blue components, respectively, and computed directional information for each block. Wang et al. [12] used the mean intensities of circles with different radii around the block center to obtain the robust feature. Lin et al. [13] got a 9-dimensional feature vector via calculating the gray average intensities of each block and its subblocks. Bravo-Solorio and Nandi [14] extracted the entropy as block feature. Though the block-based method can locate the tampered

region in pixel level accurately, the performance may decrease a lot when the suspicious image was attacked by some operations, such as noise, JPEG compression, and scaling. In addition, it is hard for these algorithms to estimate an accurate geometrical transform. Moreover, the computing complexity is very high in the matching process due to a large number of overlapping blocks, and the computation time would increase rapidly as the image size becomes larger.

The keypoint-based methods rely on the extraction of keypoints in the images. Huang et al. [15] proposed an algorithm based on Scale Invariant Feature Transform (SIFT), which is robust and sensitive to postimage processing. Amerini et al. [16] matched the SIFT descriptors via g2NN, resulting in managing multiple regions matching successfully. Speeded-Up Robust Feature (SURF) [17, 18] is used to improve matching efficiency, and the length of this feature is half of SIFT. SIFT and SURF are the most widely used keypoints for CMFD. Some other local features are also proposed to detect duplicate regions, like Local Binary Pattern (LBP) [19], Binary Robust Invariant Scalable Keypoints (BRISK) [20], and DAISY [21]. Though the computing complexity of these algorithms in matching process is much lower, they also have a main drawback: the performance will be very poor when the copy-move regions are smooth.

To overcome the shortcomings in block-based and keypoint-based approaches, a novel matching strategy is proposed to detect the copy-move forgery in digital images. We divide image into nonoverlapping blocks; then the image is segmented into smooth part and texture part according to the distribution of keypoints. In texture part, we use the extracted keypoints to find duplicate regions, while in smooth part, in order to reduce the computational complexity, blocks with similar color are clustered into a group, and searching duplicated blocks will also be carried out in the same group. Because we use nonoverlapping blocks, we cannot find the two same blocks in the content, which results in mismatching. So we need to update the candidate block by registration if the query block and the candidate block are partly consistent in content before projecting them to hash space. Finally, LSH algorithm is used to group the updated candidate blocks into several hash buckets. The strategy based on nonoverlapping can significantly reduce computational complexity, while maintaining the high accuracy levels for CMFD. The rest of the paper is organized as follows. In Section 2, we show the framework in our proposed method. The experimental results are presented in Section 3, and Section 4 draws conclusions.

2. Nonoverlapping Blocks Based CMFD Approach

The framework of our method is shown in Figure 1, which consists of three key steps: image division, feature matching, and postprocessing. In the first step, an image is divided into two parts: the yellow part is the texture region, while the blue part is the smooth region. In the second step, two types of features for the two parts are extracted to obtain the matched parts. In the last step, falsely matched pairs are removed; then we exploit morphology operations to generate a forgery detection map.

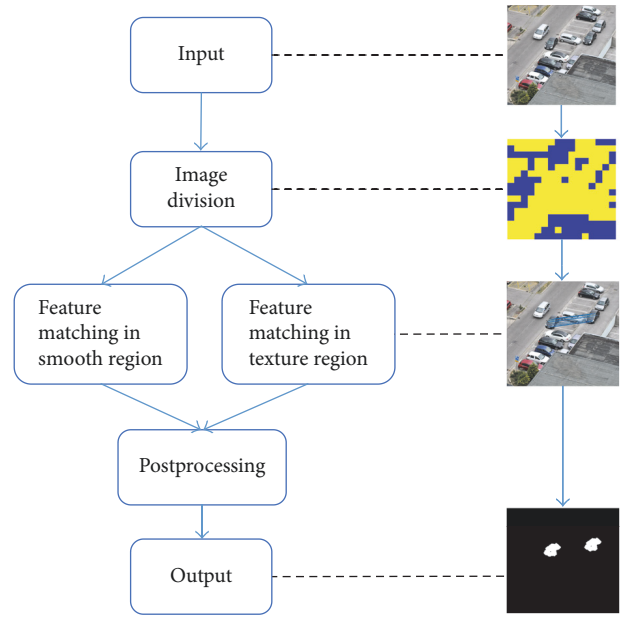

Figure 1: The framework of the proposed method.

2.1. Image Division. In this section, we will introduce the image division. When we get a suspicious image, we first divide the image into nonoverlapping blocks; then we extract the keypoints in each of them. There are many kinds of keypoints, and we use SIFT in this paper. The reason why we choose SIFT keypoint will be explained in Section 2.2. Finally, we count the number of keypoints N_i in each block. As we all know, the identification and selection of the keypoints rely on the high-entropy regions. As a result, most keypoints can only exist in texture regions. Consequently, we select a threshold τ_1 to decide whether the current block is a smooth block or a texture block.

$$\begin{aligned} N_i \geqslant \tau_1 \quad & \text{block}^i \in \text{texture block} \\ N_i < \tau_1 \quad & \text{block}^i \in \text{smooth block}. \end{aligned} \tag{1}$$

2.2. Feature Matching. According to the paper [23], we know that SIFT feature and Zernike moments are recommended for their excellent performance among the keypoint-based and block-based features, respectively. In our implementation, we employ vlFeat software to help us to detect and describe the keypoints. Once the parameters are set, the distribution and quantity of the keypoints are fixed. SIFT feature can detect forged regions even after some attacks but fail to detect flat area. Zernike moments are invariant to rotation and show effective for flat area. Therefore, in our implementation we choose SIFT and Zernike moments as features for detection in texture blocks and smooth blocks, respectively.

2.2.1. Feature Matching in Texture Region. After image division, we obtain a set of keypoints $X = \{x_1, x_2, \ldots, x_n\}$

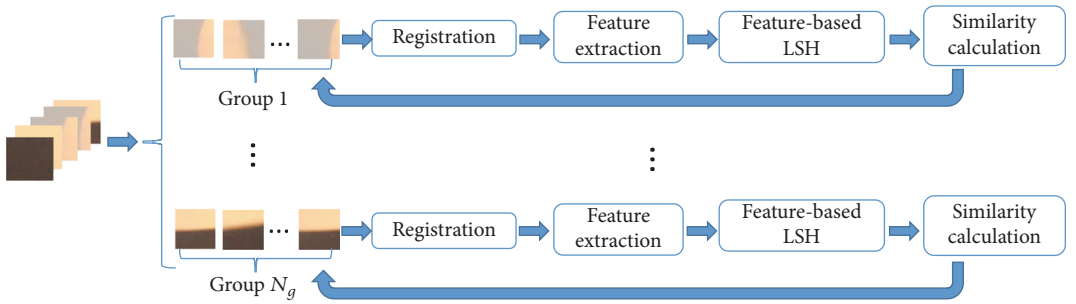

Figure 2: The framework of feature matching in smooth region.

with their corresponding descriptors $S = \{s_1, s_2, \ldots, s_n\}$. We use g2NN matching in this step, because this method can well solve multiple regions matching problem. The candidate matches for each keypoint x_i are found by calculating the Euclidean distance between all the other $(n-1)$ keypoints; then we obtain the vector $D = \{d_1, d_2, \ldots, d_{n-1}\}$ which represents these distances sorted from nearest to farthest. Ratios d_i/d_{i+1} are calculated in turn $T_i = d_i/d_{i+1}\,(i = 1, 2, 3, \ldots, n-2)$. The iteration stops in the value k when $T_k < \tau_2$ and $T_{k+1} > \tau_2$. Each keypoint in correspondence to a distance in $\{d_1, d_2, \ldots, d_k\}$ is considered as a matched keypoint.

2.2.2. Feature Matching in Smooth Region. Due to the fact that only a small number of keypoints can be extracted in the smooth region, we propose a new matching strategy. Firstly, blocks with similar color are clustered into a group. Secondly, blocks are updated by registration within a group. Feature-based LSH is used to accelerate finding correct matches in the last. Noting that after the first round of matching we need to do registration according to the next block and execute the same operation until all the blocks in this group are traversed. The framework of feature matching in smooth region is shown in Figure 2.

We assume that the source region and its corresponding target region in the image share the same color distribution. Hence, when we detect a pair of suspicious regions, we just need to search in blocks which have similar color. For example, if the content of the coping source region is ocean, its matching region must exist in the blue region in the image. In addition, in this way, we can reduce computational complexity. In order to remove the influence of brightness, we convert these selected smooth blocks from RGB color space to HSV color space to get new ones B_{hsv}. The H and S components are uniformly quantized to the k levels, where $H = 0, 1, \ldots, k-1$; $S = 0, 1, \ldots, k-1$. Then we compute the mean value of H and S in each block $B_{\text{hsv}}^{(i,j)}$ to obtain $h_{\text{mean}}^{(i,j)}$ and $s_{\text{mean}}^{(i,j)}$, where (i, j) represents the coordinates of the upper left corner of the block. We use $(h_{\text{mean}}^{(i,j)}, s_{\text{mean}}^{(i,j)})$ as block's color feature; then some of them will be divided into a group if their color features are very similar, since H and S are quantized to the k levels, respectively, and the total number of color groups is k^2. Finally, we count all the nonempty color groups; then the color group can be represented as $\{\text{group}_1, \text{group}_2, \ldots, \text{group}_{Ng}\}$, where N_g is the number of the nonempty color groups.

The traditional block-based methods usually use overlapping blocks to extract feature, which are very time consuming, so we use nonoverlapping blocks in this paper. As shown in Figure 3, the copying source region is marked by a green line and the pasting target region is marked by a blue line. The two red solid lines a and b are candidate blocks to be matched in the next step. If we use the LSH algorithm directly to find similar blocks, the mismatching is more likely to occur due to disalignment. To solve this problem, before hashing them, we need to calibrate the remaining blocks according to each block in turn within a color group.

Phase correlation is a common method of image registration [24]. Given an block $b_1(x, y)$, shift it by $(\Delta x, \Delta y)$, and we will get $b_2(x, y)$, where

$$b_2(x, y) = b_1(x - \Delta x, y - \Delta y). \tag{2}$$

Their Fourier transforms satisfy

$$B_2(u, v) = B_1(u, v) \cdot e^{j(u\Delta x + v\Delta y)}. \tag{3}$$

The cross power spectrum of them is

$$P(u, v) = \frac{B_1(u, v)\, B_2^*(u, v)}{\left\| B_1(u, v)\, B_2^*(u, v) \right\|}. \tag{4}$$

As shown in Figure 4, the inverse Fourier transformation of (4) is a two-dimensional pulse function, with a peak at $(-\Delta x, -\Delta y)$. We make block$^{(i,j)}$ and block$^{(u,v)}$ stand for two candidate matching blocks a and b in Figure 3. We first compute the inverse Fourier transform of their cross power spectrum. If the peak value is larger than τ_3 and the values of other positions are lower than τ_4, we consider that the two blocks satisfy the registration condition. Then we use the position of the peak $(\Delta x, \Delta y)$ as a displacement vector to change block$^{(u,v)}$ to block$^{(u-\Delta x, v-\Delta y)}$. In Figure 2, the updated block is marked by a red dashed line. On the contrary, if these two blocks cannot satisfy the registration condition, we will not update the current candidate block.

We use Zernike moments as block feature here. Zernike moments of order n and repetition m of a digital image $I(\rho, \theta)$ are defined as

$$Z_{n,m} = \frac{n+1}{\pi} \sum_{(\rho,\theta) \in \text{uint disk}} \sum I(\rho, \theta)\, V_{n,m}^*(\rho, \theta), \tag{5}$$

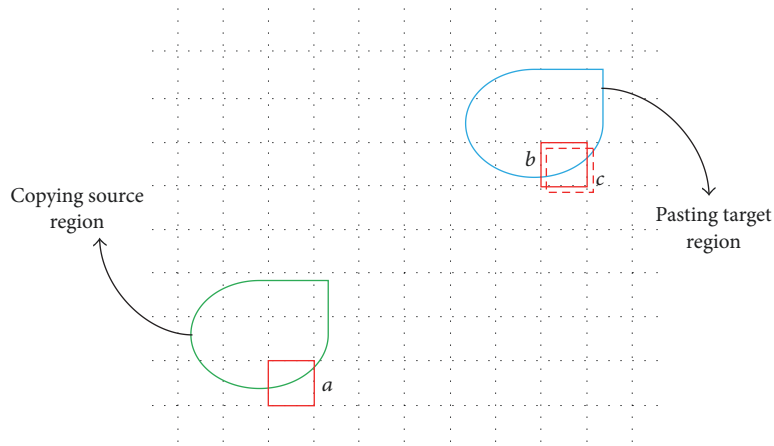

Figure 3: Alignment of candidate blocks.

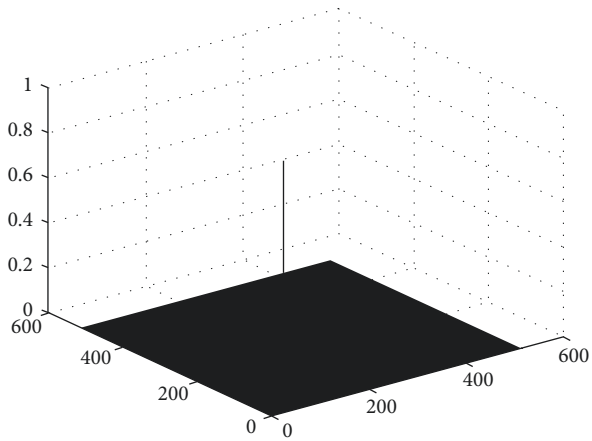

Figure 4: The inverse Fourier transformation of cross power spectrum.

where $V^*_{n,m}(\rho,\theta)$ is a Zernike polynomial of order n and repetition m.

$$V_{n,m}(\rho,\theta) = R_{n,m}(\rho)\, e^{jm\theta}, \tag{6}$$

where $n = 0, 1, \ldots, 0 \leqslant |m| \leqslant n$, $n - |m|$ is even, and $R_{n,m}(\rho)$ are real-valued radial polynomials. We choose the order of the Zernike moment, $n = 5$. Consequently, the Zernike moments extracted from all blocks can be grouped as follows:

$$Z_l = \left\{ z^{(i,j)}, z^{(p,q)}, \ldots, z^{(u,v)} \right\}, \quad l = 1, 2, \ldots, N_g. \tag{7}$$

We make the blocks corresponding to feature vectors in the set Z_l satisfying

$$\left(\text{block}^{(i,j)}, \text{block}^{(p,q)}, \ldots, \text{block}^{(u,v)} \right) \in \text{group}_l. \tag{8}$$

Feature vectors can be regarded as data points distributed in high dimensional character space. The closer the data points, the higher the similarity they have. LSH algorithm can project data points from original data space to hash space through hash functions, and the hash functions satisfy the intuitive notion that the probability of a hash collision for two points is related to the similarity between the points.

Each feature vector $z^{(i,j)}$ can be projected to hash space and get hash value. We choose a hash function based on p-stable distribution:

$$\text{hash}^{(a,b)}\left(z^{(i,j)}\right) = \left\lfloor \frac{a \cdot z^{(i,j)} + b}{w} \right\rfloor, \tag{9}$$

where $\lfloor \cdot \rfloor$ represents an operation of rounding down, w is quantization step, b is a random real number located in $[0, w]$, p is set to 2, and a is a random vector, $a = (a_0, a_1, \ldots, a_{N_{\text{moments}}-1})$. In order to increase the accuracy of collision between similar vectors, we use Q groups of hash functions $g_0, g_1, \ldots, g_{Q-1}$ to project the feature vectors, respectively, in which each group has P hash functions $g(z) = (\text{hash}^{(a_0,b_0)}(z), \text{hash}^{(a_1,b_1)}(z), \ldots, \text{hash}^{(a_{P-1},b_{P-1})}(z))$. Projecting each feature vector with a group of hash functions can obtain a set of hash values that will be used as their bucket numbers. Therefore, we can get Q different hash index tables for similarity vectors searching. The establishment of a hash tables is shown in Figure 5. After projecting feature vector $z^{(i,j)}$ by Q groups of hash functions, we get Q different bucket numbers. All the feature vectors of these buckets are taken out as a set to be screened. Then the similarity between $z^{(i,j)}$ and all of these vectors is calculated. It is worth noting that we do not use the first order $Z_{0,0}$ as it represents the average intensity and its value is much higher than that of others. In our implementation, L2-norm is used to represent similarity between feature vectors. The similarity is therefore

$$\text{similar}\left(z^{(i,j)}, z^{(k,l)}\right) = \left\| z^{(i,j)} - z^{(k,l)} \right\|. \tag{10}$$

If similar $(z^{(i,j)}, z^{(k,l)}) < D$, the two blocks are detected to be similar. Then, we calibrate the rest of the blocks according to another block in this color group until all the blocks are traversed.

A lot of measures are taken to try to reduce the computational complexity of our algorithm. We divide image into

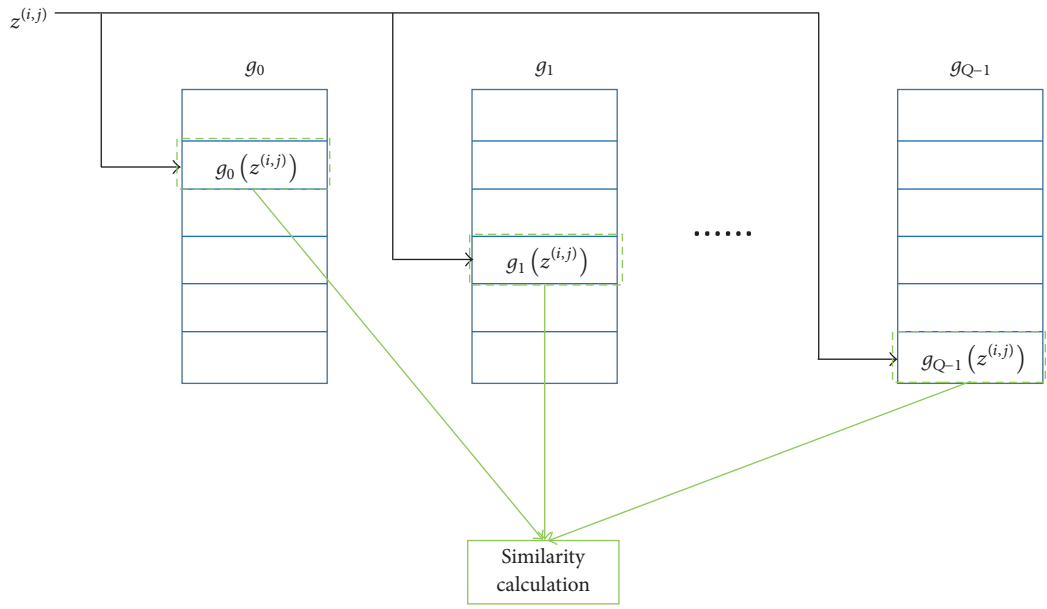

Figure 5: Establishment of hash tables. We use Q groups of hash functions to project the feature vectors, respectively. Projecting each feature vector with a group of hash functions can get a set of hash values as their bucket numbers. Finally, we take out all the feature vectors of these Q hash buckets and calculate the similarity.

texture part and smooth part; therefore we need to calculate their computational complexity separately. In texture part, we extract SIFT feature and use g2NN matching strategy, so the computational complexity of this part is $O(n^2)$, where n is the number of keypoints, while in smooth part blocks with similar color are clustered into a group. Since H and S are quantified to 10 levels, the number of color groups is 100. And in registration step, we update all the other blocks with reference to a query block each time. So the computational complexity of this part is $O(100*(m/100)^3)$ on average, where m is the number of smooth blocks. Special to note is that traditional block-based approaches use overlapping blocks to find duplicated regions, and the total number of blocks is $(M - B + 1) * (N - B + 1)$, where M and N are the width and height of the image, respectively, and B is the length of square block. It is obvious that its computational complexity is $O((M * N)^2)$. Because we use nonoverlapping blocks, $m = (M * N)/B^2$ at most, and $B = 16$ in most cases. Therefore, in the worst case the computational complexity of our method is $O(n^2 + 6 * 10^{-12} * (M * N)^3)$. The number of keypoints is usually not much, and for the most frequently used datasets, our method can be much faster than the traditional block-based methods.

2.3. Postprocessing. The goal of this last step is to present to us more accurate matches. In this paper, we consider three steps to remove falsely matched pairs, including distance, relative position, and affine transformation.

The adjacent keypoints or blocks might be misregarded as the matching pairs, due to their similar characteristics. So the matching pairs will be eliminated, if the distance between them is smaller than the threshold D_2.

The true matching pairs are often distributed densely, so a merging method named Broad First Search Neighbors (BFSN) clustering [25] is performed on spatial locations of the matching pairs to help delete the isolated keypoints or blocks. In this algorithm, if the distance between point a and point b is smaller than a threshold, the two points are defined as neighbors. For a cluster C with m elements, point p can be merged into C, while p is the neighbor of at least λm elements in C. The radio factor λ, which ranges from 0 to 1, controls size and shape of the cluster. First of all, we create an empty class A_1 and put the first row (x_1', y_1') of coordinate matrix C' into A_1. Secondly, we search for all neighbors of (x_1', y_1') in a breadth first order and determine whether the current vector can be incorporated into the class A_1 in terms of clustering conditions. Finally, we delete the vectors in C' which have been incorporated into A_1 and put the first row of the current matrix C' into A_2. Repeat the above three steps until $C' = \varnothing$. The cluster whose inner elements are less than 3 will be regarded as the isolated cluster and the elements in them will be deleted.

Truly matched pairs satisfy the same affine transformation, which means they should exhibit similar amounts of translation, scaling, and rotation. The relationship between these two matched blocks $\text{block}^{(i,j)}$ and $\text{block}^{(k,l)}$ can be represented as follows, where H is a 3×3 matrix:

$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} k \\ l \\ 1 \end{bmatrix}. \tag{11}$$

We assume that if the two matching pairs are from the same duplicated regions, the slopes of their matching lines are

Table 1: Detection result on Benchmark database.

Methods	Precision (%)	Recall (%)	$F1$ (%)
SIFT [16]	79.59	81.25	80.41
Segmentation [22]	70.18	83.33	76.19
Proposed method	90.91	83.33	86.96

consistent. BFSN clustering is performed on the slopes of matching lines to divide the matching pairs into different clusters. RANSAC algorithm [26] is then performed on each group, respectively, to eliminate false matches. This method can avoid misdeleting in the situation of managing the multiduplicated matching. In addition, the test image will be identified as the tampered one, if the number of remaining matches is more than a threshold.

3. Experimental Results

In this section, error measures are first introduced, and experiments are conducted on three databases that are available online, Benchmark presented by Christlein et al. [23], CoMoFoD [27], and GRIP [28]. The first one is composed of 48 images, which range from 800×533 to 3888×2592 in size, having about 10% of the whole image as tampered regions. The second one comprises 200 original images and their corresponding tampered images with size of 512×512. There are 7 groups of tampered images, one is without any attacks, and the rest are manufactured by adding various attacks, including JPEG compression, blurring, noise adding, and color reduction. The last one consists of 80 images, and these images have fixed size 1024×768 pixels; each of them has only one pair of duplicated regions. The hardware environment of the experiments is a 3.60 GHz Intel Core i7-4790 processor; the software used is Windows 7 MATLAB R2014b.

3.1. Error Measures. We often evaluate the performance of a CMFD method at two levels, namely, image level and pixel level. At image level, we are concerned about whether or not this image has been tampered, while in pixel level we pay more attention to the accuracy of location. Precision and recall are used here. The detection error at the image level is defined as

$$\begin{aligned} \text{precision} &= \frac{T_P}{T_P + F_P}, \\ \text{recall} &= \frac{T_P}{T_P + F_N}, \end{aligned} \tag{12}$$

where T_P is the number of images that have been correctly identified as forged, F_P represents the number of images that have been erroneously detected as forged, and F_N indicates the falsely missed forged images. The detection error at the pixel level is defined as

$$\begin{aligned} \text{precision} &= \frac{\left|\Phi \cap \Phi'\right|}{|\Phi|}, \\ \text{recall} &= \frac{\left|\Phi \cap \Phi'\right|}{\left|\Phi'\right|}, \end{aligned} \tag{13}$$

where Φ is the number of pixels detected by the CMFD method; Φ' is the number of all forgery pixels marked by ground truth.

We also use F_1 score as a comprehensive measure.

$$F_1 = \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{14}$$

3.2. Detection Results on the Benchmark Database. We first test the proposed method on Benchmark database. Several example test cases are shown in Figure 6. The first row of Figure 6 shows the original images, the second row shows the tampered images, the third row shows the ground truth maps, and the last row shows the detection results of the proposed method. In Table 1 we compare the proposed algorithm with two other methods, and the result indicates that our method can achieve 90.91% precision, 83.33% recall, and 86.96% F_1 score, which preforms better than SIFT [16] and Segmentation [22]. In the third column of Figure 6, though the tampered area is smooth, we can still detect it.

A practical CMFD algorithm should have a relatively low computational complexity in addition to maintaining a certain degree of accuracy. In order to measure the performance of our proposed method, we compare the time complexity of different algorithms on this dataset. The average execution times are illustrated in Figure 7. It should be noted that the implementation platforms of evaluated methods are different. For instance, Zernike is implemented in C++ for speed, and segmentation and the proposed method are implemented in MATLAB. Because of the high resolution, all methods demand more time. The proposed method is the fastest, except for SIFT [16].

3.3. Detection Results on the CoMoFoD Database. Next the experiments are carried out on CoMoFoD database. Illustration of four cases is shown in Figure 8. Each row from top to bottom represents original images, tampered images, ground truth, and detection results of the proposed method, respectively. Table 2 shows the detection results comparing with other methods. From this table, it can be clearly seen that the results of the proposed method outperform [16, 22] under ideal conditions.

In addition, we conduct experiments to evaluate the robustness of our method against various attacks. That will make the CMFD more difficult. The forgery images are

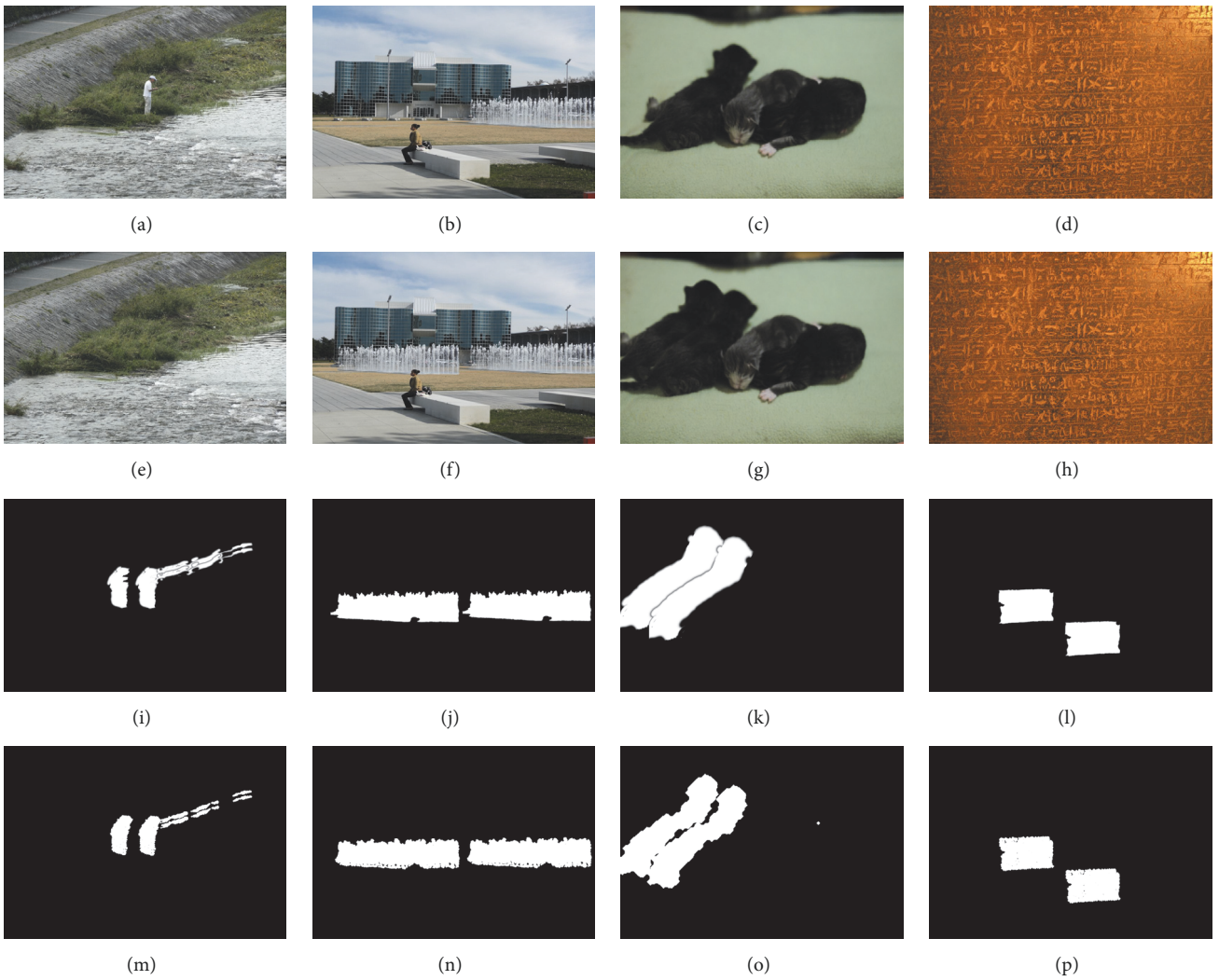

Figure 6: Example test cases. ((a), (b), (c), and (d)) Original. ((e), (f), (g), and (h)) Tampered. ((i), (j), (k), and (l)) Ground truth. ((m), (n), (o), and (p)) Proposed.

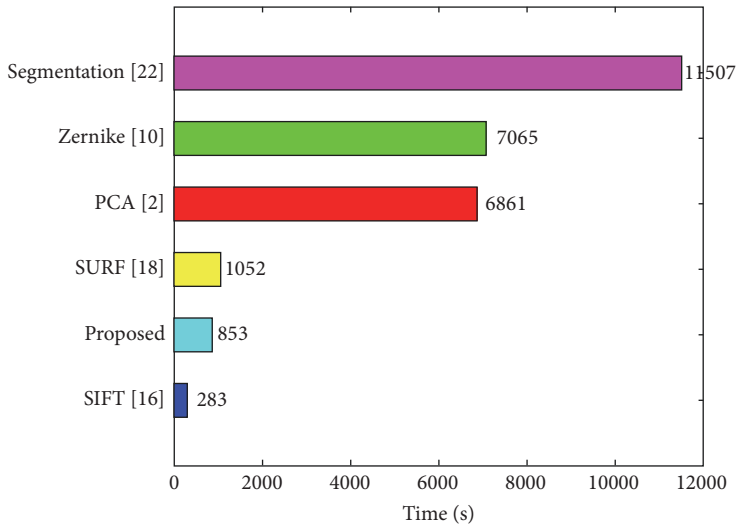

Figure 7: The average execution times [2, 10, 16, 18, 22].

Table 2: Detection result on CoMoFoD database.

Methods	Precision (%)	Recall (%)	$F1$ (%)
SIFT [16]	77.83	82.50	80.10
Segmentation [22]	77.19	66.00	71.16
Proposed method	89.30	83.50	87.55

Table 3: Detection result on GRIP database.

Methods	Precision (%)	Recall (%)	$F1$ (%)
SIFT [16]	78.87	70.00	74.17
Segmentation [22]	69.47	82.50	75.43
Proposed method	90.54	83.75	87.01

Table 4: Detection results on pixel level.

Serial number	Precision (%)	Recall (%)	$F1$ (%)
8	99.18	60.05	74.80
10	86.04	48.54	62.06
12	99.56	70.13	82.29
30	91.26	89.57	90.40
31	98.51	72.30	83.39
54	86.88	66.62	75.42
61	98.80	85.76	91.82

generated by making the copied snippets undergo 3 kinds of attacks, namely, JPEG compression, rotation, and scaling.

(1) JPEG compression: the image quality-factor varies from 20 to 100 with the step of 10. The results are shown in first row of Figure 9. For most of the quality factors, the precision, recall, and F_1 score of our method are higher than those of [16, 22]. The curve of red line has smaller slope than others; that means our method has better robustness against JPEG compression.

(2) Rotation: we rotate the copied snippets with the rotation angles varying from 2° to 10° with the step of 2°. The results are shown in second row of Figure 9. It can be clearly seen that our method is much better than [16, 22].

(3) Scaling: the copied snippets are scaled with the scale factors varying from 91% to 109% with the step of 2%. As shown in third row of Figure 9, our method outperforms [16, 22] in terms of precision and F_1 score, and recall is almost as same as that of [16].

3.4. Detection Results on the GRIP Database. We also test the performance of the proposed method on the GRIP database. The detection results are given in Table 3. It can be observed that all evaluation indicators of our method are well over those of [16, 22]. In particular, each evaluation indicator is more than 11% higher than that of [16]. This is because the database contains a lot of smooth images, and our method divides the smooth area of the image into nonoverlapping blocks to find correct duplicated regions.

We first classify the blocks into several groups according to their color distribution. In order to prove that this step can improve the efficiency of the algorithm, a baseline reference technique with registration and LSH is considered. From Figure 10, we are able to see that the advantage of grouping is obvious, and the computational complexity is decreased by 94.67%.

Moreover, we present some experiments to assess the impact of registration. Some smooth images from this dataset are picked out for this purpose. Results reported in Table 4 confirm the effectiveness of the registration step in matching.

4. Conclusion

In this paper, we propose a new CMFD method, which is able to solve the problem of high computational complexity in traditional block-based methods. First, image is divided into two different types of regions. Next, features are extracted and matched in texture region and smooth region using SIFT and Zernike moments, respectively. In particular, we use nonoverlapping blocks as candidates in smooth area and do the registration via the phase correlation algorithm before matching. Finally, the exact forgery regions will be generated after removing falsely matched pairs and exploiting morphology operations. There are three main contributions in our proposed method as follows. (1) Dividing image into texture region and smooth region according to the number of keypoints can not only avoid the poor performance using SIFT in smooth area but also reduce computational complexity. (2) We use nonoverlapping blocks as candidates in smooth area instead of using all overlapping blocks

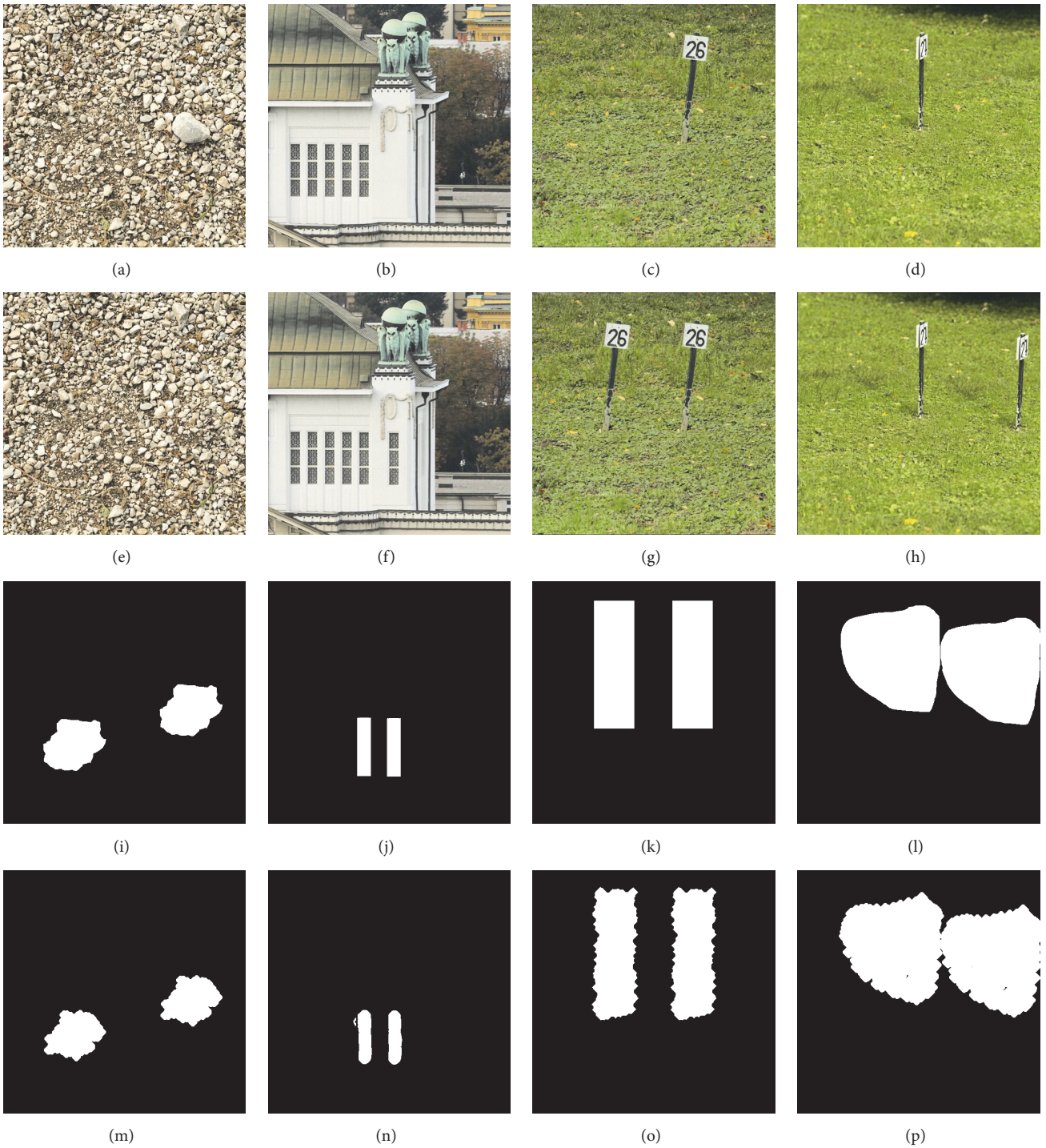

FIGURE 8: Illustration of four cases. ((a), (b), (c), and (d)) Original. ((e), (f), (g), and (h)) Tampered. ((i), (j), (k), and (l)) Ground truth. ((m), (n), (o), and (p)) Proposed.

and apply color based feature to decrease the number of blocks needed to be matched. (3) In order to prevent mismatching due to disalignment, we do registration before using LSH algorithm to obtain more accurate candidate blocks.

Experimental results show that the performance of the proposed algorithm is better than the state-of-the-art CMFD methods. In addition, our method can manage multiple copied regions. Meanwhile, it exhibits robustness to JPEG compression, rotation, and scaling. However, when dealing with very smooth blocks, such as sky and wall, the effect of registration is not particularly good. In our future work, we will try to improve the accuracy of location and the detection speed.

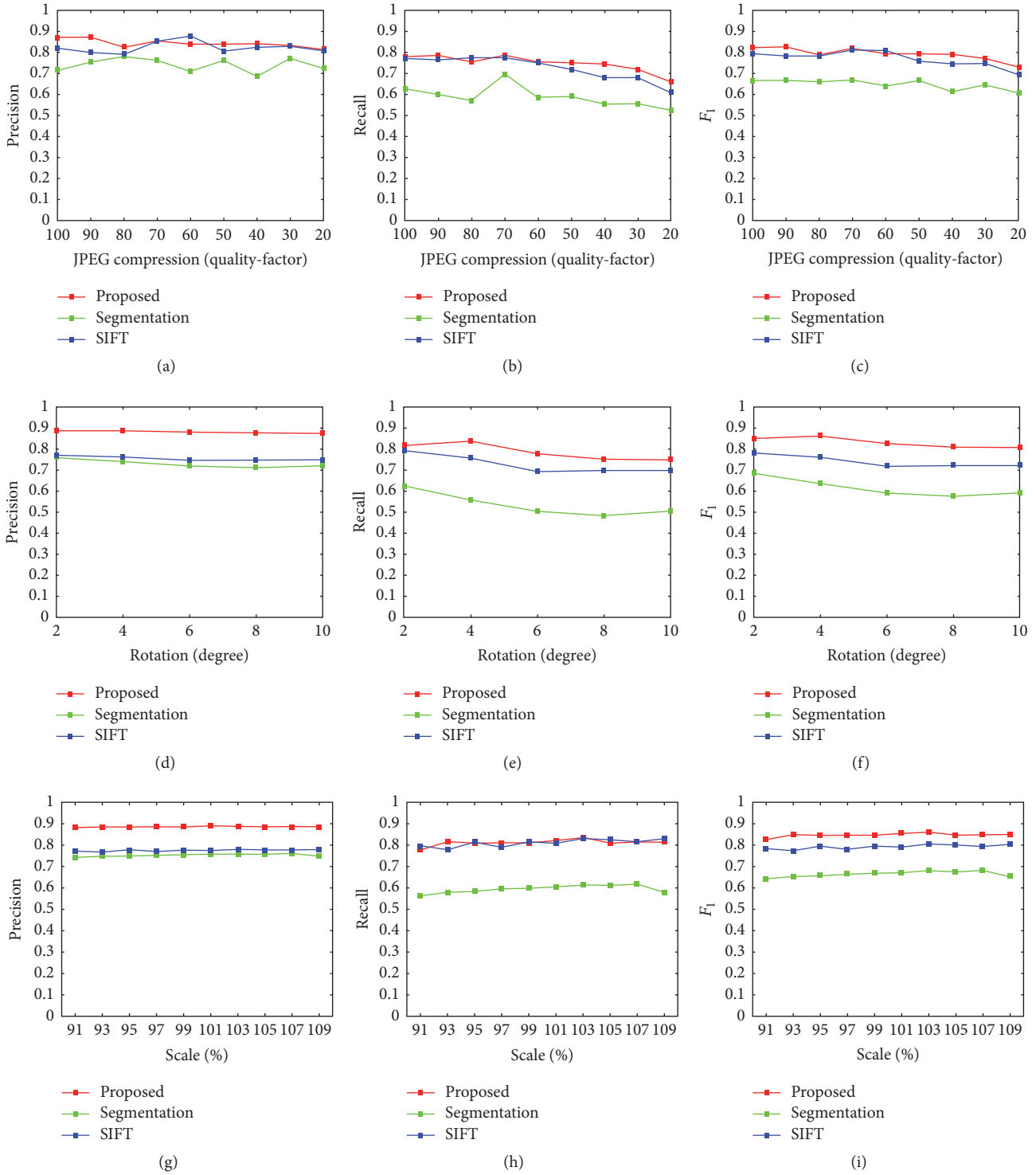

Figure 9: Detection results under 3 kinds of attacks. The three rows from (a–c), (d–f), and (g–i) are JPEG compression, rotation, and scaling, respectively. The three columns from (a–g), (b–h), and (c–i) give the precision, recall, and F_1 score, respectively.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by National Key Research and Development of China (2016YFB0800404), National

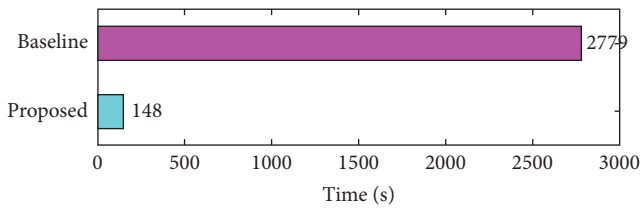

Figure 10: Average execution time per image for the proposed method and baseline reference.

NSF of China (61672090, 61332012), and Fundamental Research Funds for the Central Universities (2015JBZ002).

References

[1] J. Fridrich, D. Soukal, and J. Lukāiåi, "Detection of copy-move forgery in digital images," in *Proceedings of Digital Forensic Research Workshop*, Citeseer, 2003.

[2] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of resampling," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 758–767, 2005.

[3] M. Bashar, K. Noda, N. Ohnishi, and K. Mori, "Exploring duplicated regions in natural images," *IEEE Transactions on Image Processing*, no. 99, pp. 1–40, 2010.

[4] G. Li, Q. Wu, D. Tu, and S. Sun, "A sorted neighborhood approach for detecting duplicated regions in image forgeries based on DWT and SVD," in *Proceedings of the IEEE International Conference onMultimedia and Expo (ICME '07)*, pp. 1750–1753, Beijing, China, July 2007.

[5] X. B. Kang and S. M. Wei, "Identifying tampered regionsusing singular value decomposition in digital image forensics," in *Proceedings of the International Conference on Computer Science and Softare Engineering (CSSE '08)*, vol. 3, pp. 926–930, IEEE, December 2008.

[6] S. Bayram, H. T. Sencar, and N. D. Memon, "An efficient and robust method for detecting copy-move forgery," in *Proceedings of the Proceeding of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '09)*, pp. 1053–1056, Taipei, Taiwan, April 2009.

[7] B. Mahdian and S. Saic, "Detection of copy-move forgery using a method based on blur moment invariants," *Forensic Science International*, vol. 171, no. 2-3, pp. 180–189, 2007.

[8] J.-W. Wang, G.-J. Liu, Z. Zhang, Y.-W. Dai, and Z.-Q. Wang, "Fast and robust forensics for image region-duplication forgery," *Acta Automatica Sinica*, vol. 35, no. 12, pp. 1488–1495, 2009.

[9] S. Ryu, M. Kirchner, M. Lee, and H. Lee, "Rotation invariant localization of duplicated image regions based on zernike moments," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1355–1370, 2013.

[10] S.-J. Ryu, M.-J. Lee, and H.-K. Lee, "Detection of copy-rotate-move forgery using zernike moments," in *Information Hiding*, vol. 6387 of *Lecture Notes in Computer Science*, pp. 51–65, Springer, 2010.

[11] W. Luo, J. Huang, and G. Qiu, "Robust detection of region-duplication forgery in digital image," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, vol. 4, pp. 746–749, Hong Kong, August 2006.

[12] J. Wang, G. Liu, H. Li, Y. Dai, and Z. Wang, "Detection of image region duplication forgery using model with circle block," in *Proceedings of the 1st International Conference on Multimedia Information Networking and Security (MINES '09)*, vol. 1, pp. 25–29, IEEE, November 2009.

[13] H. J. Lin, C. W. Wang, and Y. T. Kao, "Fast copy-move forgery detection," *WSEAS Transactions on Signal Processing*, vol. 5, pp. 188–197, 2009.

[14] S. Bravo-Solorio and A. K. Nandi, "Exposing duplicated regions affected by reflection, rotation and scaling," in *Proceedings of the 36th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '11)*, pp. 1880–1883, Prague, Czech Republic, May 2011.

[15] H. Huang, W. Guo, and Y. Zhang, "Detection of copy-move forgery in digital images using Sift algorithm," in *Proceedings of the Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA '08)*, pp. 272–276, Computer Society, December 2008.

[16] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A SIFT-based forensic method for copy-move attack detection and transformation recovery," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, 2011.

[17] B. Xu, J. Wang, G. Liu, and Y. Dai, "Image copy-move forgery detection based on SURF," in *Proceedings of the 2nd International Conference on Multimedia Information Networking and Security (MINES '10)*, pp. 889–892, IEEE, Nanjing, China, November 2010.

[18] B. L. Shivakumar and S. Baboo, "Detection of region duplication forgery in digital images using SURF," *International Journal of Computer Science Issues*, vol. 8, no. 4, pp. 199–205, 2011.

[19] R. Davarzani, K. Yaghmaie, S. Mozaffari, and M. Tapak, "Copy-move forgery detection using multiresolution local binary patterns," *Forensic Science International*, vol. 231, no. 1–3, pp. 61–72, 2013.

[20] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 2548–2555, Barcelona, Spain, November 2011.

[21] R. Sekhar and R. S. Shaji, "A study on segmentation-based copy-move forgery detection using DAISY descriptor," *Advances in Intelligent Systems and Computing*, vol. 398, pp. 223–233, 2016.

[22] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507–518, 2015.

[23] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, 2012.

[24] A. Yousef, J. Li, and M. Karim, "High-speed image registration algorithm with subpixel accuracy," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1796–1800, 2015.

[25] J. Qian and Y. Dong, "Clustering algorithm based on broad first searching neighbors," *Journal of Southeast University (Natural Science Edition)*, vol. 34, no. 1, pp. 109–112, 2004.

[26] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[27] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic, "CoMoFoD - New database for copy-move forgery detection," in *Proceedings of the 55th International Symposium ELMAR 2013*, pp. 49–54, hrv, September 2013.

[28] D. Cozzolino, G. Poggi, and L. Verdoliva, "Copy-move forgery detection based on PatchMatch," in *IEEE International Conference on Image*, pp. 5312–5316, 2013.