Fast and Effective Image Copy-Move Forgery Detection via Hierarchical Feature Point Matching

Yuanman Li, *Student Member, IEEE*, and Jiantao Zhou, *Member, IEEE*

***Abstract*— Copy-move forgery is one of the most commonly used manipulations for tampering digital images. Keypoint-based detection methods have been reported to be very effective in revealing copy-move evidence due to their robustness against various attacks, such as large-scale geometric transformations. However, these methods fail to handle the cases when copy-move forgeries only involve small or smooth regions, where the number of keypoints is very limited. To tackle this challenge, we propose a fast and effective copy-move forgery detection algorithm through hierarchical feature point matching. We first show that it is possible to generate a sufficient number of keypoints that exist even in small or smooth regions by lowering the *contrast threshold* and rescaling the input image. We then develop a novel hierarchical matching strategy to solve the *keypoint matching problems* over a massive number of keypoints. To reduce the false alarm rate and accurately localize the tampered regions, we further propose a novel iterative localization technique by exploiting the robustness properties (including the dominant orientation and the scale information) and the color information of each keypoint. Extensive experimental results are provided to demonstrate the superior performance of our proposed scheme in terms of both efficiency and accuracy.**

***Index Terms*— Copy-move, forgery detection, hierarchical feature matching, iterative localization.**

I. Introduction

With the development of modern image editing softwares, such as Photoshop and Gimp, digital images can be forged at a very low cost. This brings a big threat for the reliability of digital images. Copy-move forgery is one common manipulation among various digital image forgeries, where one or several regions of an image are pasted elsewhere in the same image in order to hide or duplicate objects of interest [1]–[7]. Such process may be accompanied with rotation, resizing, compression and noise addition to make the final forgeries more convincing. Detecting them sometimes can be very challenging, especially when the copy-move forgery only involves small or smooth regions, or when the forged areas have been processed by some severe attacks, such as large-scale resizing and heavy noise addition. Two examples are shown in Fig. 1, where the copy-move forgeries are conducted over only smooth or small regions. In the recent years, many image copy-move forgery detection methods have been proposed, which can be roughly categorized into two groups: 1) dense-field (or block-based) approaches [1], [2], [6], [8]–[13] and 2) sparse-field (or keypoint-based) approaches [3], [5], [14]–[21].

Fig. 1. Two Examples of the copy-move forgery. Left to right: original images, forged images through copy-move operations, and copy-move regions.

For the dense-field copy-move forgery detection approaches, the input images are first divided into overlapped and regular blocks; then the forgery localization procedure is performed through block matching. To improve the robustness against some common distortions, such as geometric transformations, various techniques have been employed to design the block features, such as Discrete Cosine Transform (DCT) [1], Discrete Wavelet Transform (DWT) [2], Principal Component Analysis (PCA) [8], Singular Value Decomposition (SVD) [9], and other techniques [10], [11]. The dense-field approaches were shown to be more accurate than the keypoint-based ones at the cost of higher complexity [6]. More recently, Cozzolino *et al.* [12] proposed an efficient dense-field copy-move forgery detection method, where the processing time was highly reduced by resorting to the PatchMatch algorithm—a fast approximate nearest-neighbor search scheme [22]. Unfortunately, all the existing dense-field schemes suffer from some attacks, such as scaling, rotation and noise addition. This can

Manuscript received April 27, 2018; revised August 30, 2018 and October 4, 2018; accepted October 8, 2018. Date of publication October 22, 2018; date of current version January 30, 2019. This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/046/2014/A1 and Grant FDCT/022/2017/A1 and in part by the Research Committee, University of Macau, under Grant MYRG2016-00137-FST and Grant MYRG2018-00029-FST. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Luisa Verdoliva. *(Corresponding author: Jiantao Zhou.)*

The authors are with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau 999078, China (e-mail: yb57410@umac.mo; jtzhou@umac.mo).

Digital Object Identifier 10.1109/TIFS.2018.2876837

be fully validated by the experimental results to be presented in Section VI.

Pan and Lyu [14] pioneered the work on using keypoint matching for robust copy-move forgery detection. Assisted by the Scale Invariant Feature Transform (SIFT) feature [23], their method was shown to be very robust against geometric transformations, where the parameters were estimated by the RANdom SAmple Consensus (RANSAC) algorithm [24]. A somewhat similar scheme was proposed by Amerini *et al.* [3] to detect multiple duplicated regions, where the matched correspondences may follow different geometric transformations. In this case, a global the RANSAC estimation over all the matched pairs does not work any longer. To tackle this issue, [3] suggested to use the hierarchical agglomerative clustering algorithm [25] to group the matched keypoints into separated clusters based on their locations in the image plane, and then apply the RANSAC estimation over each two *matched* clusters. Rather than clustering the keypoints, [16] proposed to cluster the matched pairs in a *conceptual* space. For brevity, we call such keypoint-based techniques involving clustering procedures as *keypoint-clustering-based* algorithms. Instead of using the clustering algorithms to group the matched keypoints, some other researchers proposed to first segment the whole image into non-overlapped small patches; the matching process was then conducted between each two segmented regions [5]. In our work, we call those keypoint-based techniques involving segmentation procedures as *keypoint-segmentation-based* algorithms. Besides the SIFT descriptor, SURF [17], LBP [18] and some other local features [19], [20] were also considered in the recent literatures. Though the keypoint-based copy-move forgery detection methods have been studied from various aspects, they were unfortunately shown to be less accurate than the dense-field ones [6], [12], and the performance gap was quite large when the copy-move forgery only involves small or smooth regions as shown in Fig. 1. The main drawbacks of the existing keypoint-based copy-move forgery detection methods can be summarized as follows:

1) They fail to generate a sufficient number of keypoints (hence matched pairs) in those small or smooth copy-move regions, causing detection failure;
2) It is very difficult (even impossible) to find a *universally good* clustering/segmentation algorithm and associated parameters applicable for all images. This is because the copy-move regions can be of any sizes, and can be highly diverse from the textures. In addition, the number of copy-move regions is typically unknown; properly performing the clustering in this case is difficult;
3) The existing keypoint-based methods lack of reliable affine matrix validation and inliers selection, in the sense that some outliers could be treated as inliers by the existing homography estimation techniques (e.g., RANSAC), causing a high false alarm rate.

In this paper, we propose an efficient and accurate keypoint-based method for image copy-move forgery detection and localization, achieving consistently good performance even if the copy-move forgery only involves smooth or small regions, or the forged images have been processed by some severe attacks (e.g., large-scale resizing and heavy noise addition). Fig. 2 presents the framework of our proposed image forgery detection scheme, which follows the classic workflow, namely, 1) feature extraction; 2) feature matching; and 3) forgery localization. Our main contribution lies in designing novel and sophisticated solutions for *all* these three steps. At the first stage, we design a simple yet effective way to extract a sufficient number of SIFT keypoints, even in smooth and small regions, by lowering the *contrast threshold* and rescaling the input image. At the second stage, a novel hierarchical point matching strategy is proposed to solve the *keypoint matching problems* over a massive number of keypoints. At the third stage, a novel iterative homography estimation and a copy-move localization technique are suggested, without involving any clustering and segmentation procedures. By fully exploiting the robustness properties (including the dominant orientation and the scale information) and the color information of each keypoint, our proposed method achieves very accurate detection results at considerably lowered computational cost. Extensive experimental results demonstrate that our proposed scheme leads to a higher True Positive Rate (TPR) and a lower False Positive Rate (FPR) simultaneously in most of the cases, compared with both the existing dense-field and keypoint-based approaches.

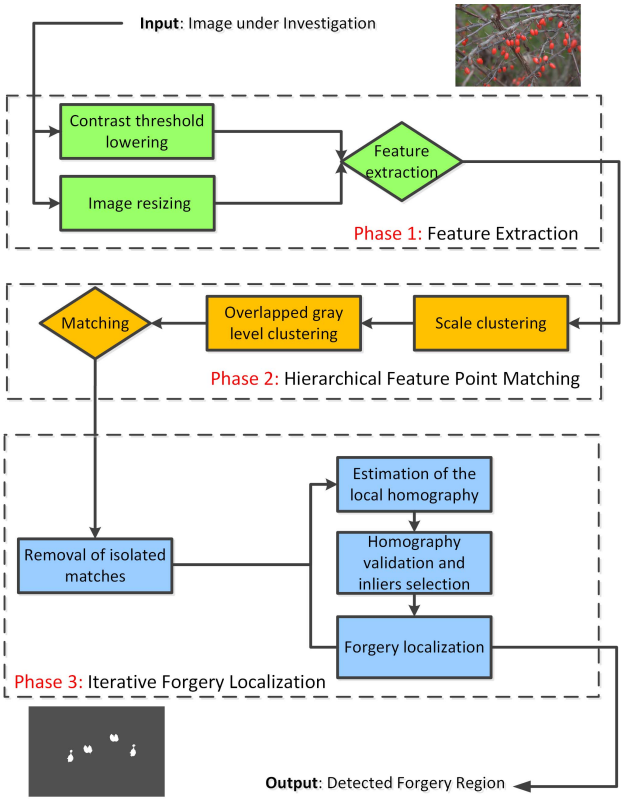


Fig. 2. Framework of the proposed algorithm.

Difference from the Conference Version: Portions of the work presented in this paper have previously appeared in [21] as a conference version. We have substantially refined the paper in terms of both technical and experimental parts. The primary improvements can be summarized as follows. First of all, we carefully present the strategy to select inliers

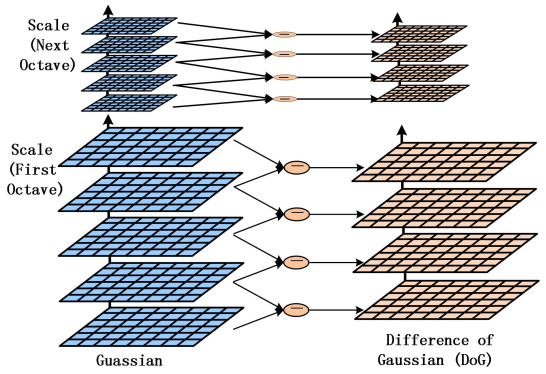


Fig. 3. The schematization of Gaussian-blurred images (left) and DoG images (right).

using the dominant information in Section V-C, and we insert a new Section V-D to present the forgery localization in dense fields by exploiting the scale information and the color information of each keypoint. Secondly, in Section VI, we compare our algorithm with many state-of-the-art schemes to thoroughly demonstrate the superior detection performance. Further, we conduct the computational complexity comparison to show the high efficiency of our scheme. Last but not least, a new Section VI-D is provided to evaluate the robustness against different transforms. We demonstrate that the proposed scheme achieves better detection performance at both the image level and the pixel level, even under some challenging circumstances (e.g., large-scale resizing).

The remainder of this paper is organized as follows. Section II gives a brief introduction of the SIFT algorithm. Section III describes the feature extraction procedure, and Section IV explains the hierarchical feature point matching scheme. A novel forgery localization algorithm is detailed in Section V. Experimental results on the copy-move forgery detection are presented in Section VI, and a short conclusion is finally drawn in Section VII.

II. Introduction to the SIFT

As one of the most popular algorithms in computer vision to extract and describe image local features, the SIFT [23] has been shown to be excellently robust against noise distortion and geometric transformations [26], [27]. In this Section, we briefly review the SIFT feature generation and matching algorithm.

A. SIFT Feature Generation

The SIFT algorithm can be roughly divided into four phases: i) candidate keypoint identification through the scale space extrema detection; ii) keypoints refinement according to the contrast and edge thresholds; iii) dominant orientation assignment of each keypoint; and iv) feature descriptor generation.

Fig. 3 shows the construction of the scale space. At phase i), the candidate keypoints are identified at different scales. Given an input image $\mathbf{I}$, successive Gaussian-blurred images are generated by repeatedly convolving $\mathbf{I}$ with Gaussian filters at multiple scales. Then, the candidate SIFT keypoints are selected as local extrema within a $3 \times 3 \times 3$ cube of the *Difference of Gaussians* (DoG) domain. Specifically, the DoG image at scale σ is given by

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma), \tag{1}$$

where k is a predefined constant and $L(x, y, \sigma)$ denotes the Gaussian-blurred image calculated by

$$L(x, y, \sigma) = \mathbf{I}(x, y) \otimes G(x, y, \sigma). \tag{2}$$

Here $G(x, y, \sigma)$ is the Gaussian kernel.

At phase ii), all the candidate keypoints are further refined according to a contrast threshold and an edge threshold. This procedure plays a key role for rejecting unstable extrema in the SIFT algorithm. At phase iii), a dominant orientation is assigned to each survived keypoint to achieve rotation invariance. For each point (x, y, σ), its orientation is computed as

$$\theta(x, y, \sigma) = \tan^{-1}\left(\frac{d_y}{d_x}\right), \tag{3}$$

where d_y and d_x are the vertical and horizontal gradients of (x, y, σ). An orientation histogram is then constructed by gathering the gradient orientation information of points in a local window centered at the SIFT keypoint. The peak in the orientation histogram corresponds to the dominant orientation. At phase iv), a 128-dimensional descriptor is calculated by encoding the surrounding information in a local area (16×16 sized in the scale space) centered at the SIFT keypoint.

Through the above four phases, a list of n keypoints $\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_n\}$ and their corresponding descriptors $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ are generated for a given image $\mathbf{I}$. Let $\mathbf{k}$ be a generic SIFT keypoint, which is represented as a four dimensional vector

$$\mathbf{k} = (x_k, y_k, \sigma_k, \theta_k), \tag{4}$$

where (x_k, y_k) are the coordinates in the image plane, σ_k denotes the scale and θ_k serves as its dominant orientation. For more details about the SIFT, please refer to [23].

B. SIFT Feature Matching

To find a reliable match (may not exist though) of the keypoint $\mathbf{k}$, simply evaluating the distances with the other $(n-1)$ keypoints against a global threshold does not perform well in the high dimensional feature space [3], [23]. The widely used matching algorithm was suggested in the original SIFT paper [23], where the matching procedure is conducted by evaluating the ratio of the closest distance to the second-closest one. The rationale behind is that for those false matches, there will very likely be several other false matches with similar distances. This is because the distances are computed in the high dimensional feature space. Specifically, let vector $\mathbf{d} = \{d_1, d_2, \ldots, d_{n-1}\}$ record the Euclidean distances between the keypoint $\mathbf{k}$ and the remaining $(n-1)$ keypoints in an increasing order, i.e., $d_1 \leq d_2 \leq \cdots \leq d_{n-1}$. Then, the keypoint $\mathbf{k}$ is matched with one of the other $(n-1)$ keypoints if and only if

$$d_1/d_2 < t, \tag{5}$$

where $t \in (0, 1)$ is a predefined parameter commonly set as 0.6.

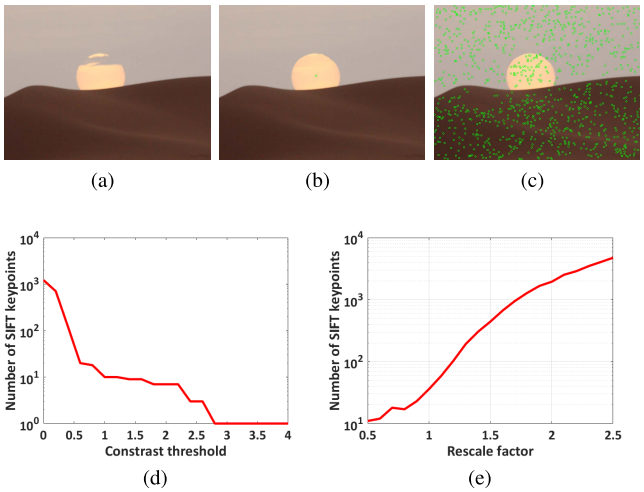


Fig. 4. (a) Original image; (b) forged image through copy-move operations, where the keypoints are obtained by setting $C = 4$; (c) forged image through copy-move operations, where the keypoints are obtained by setting $C = 0.1$; (d) and (e) show how the contrast threshold and rescaling factor affect the number of keypoints, respectively.

III. Feature Extraction

Due to its excellent robustness against the noise distortion and geometric transformations, the SIFT algorithm [23] is also employed in this work for the feature extraction. As discussed in Section I, one critical problem for the keypoint-based approaches (including SIFT-based ones) is that they cannot generate a sufficient number of keypoints in smooth or small regions, thus leading to inferior detection performance [6], [12], [28]. In this Section, we suggest two simple yet effective strategies to generate a much more number of SIFT keypoints, even in smooth or small regions, namely, 1) lowering the contrast threshold and 2) resizing the input image.

A. Lowering the Contrast Threshold

As stated in Section II (phase ii), the contrast threshold denoted by C, is predefined to reject those unstable extrema with low contrast values.[1] Typically, for each point $\mathbf{x} = (x, y, \sigma)$ in the scale space, its contrast value is given by

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\left(\frac{\partial D}{\partial \mathbf{x}}\right)^T \hat{\mathbf{x}}, \tag{6}$$

where D is defined in (1), and $\hat{\mathbf{x}}$ is the refined location of $\mathbf{x}$ in the continuous space [23]. Any extremum with contrast value smaller than C is rejected to be a final SIFT keypoint.

However, we find that in smooth regions, the contrast values of extrema tend to be very low. As a consequence, few or even no extrema are able to pass the contrast refining procedure and finally survive as SIFT keypoints. Fig. 4 gives an example, where a part of the moon is forged. When setting the contrast threshold $C = 4$ (a common setting for the VLFEAT implementation), only one keypoint can be detected, which obviously cannot provide enough copy-move forgery evidence.

To ensure that a sufficient number of keypoints can be generated in smooth regions, we suggest to lower the contrast threshold C in the SIFT algorithm, allowing lots of extrema with low contrast values to be survived. However, we need also to avoid adopting a very small C (e.g., $C = 0$ in the extreme case), as it will trigger too many unstable keypoints, thus leading to many unreliable matches. What is worse, a very small C will also aggravate the *keypoint matching problems* (to be discussed in Section IV-A). In order to find a good tradeoff, we first manually select 100 images containing smooth regions such as sky, glass, desert, moon etc., from the datasets list in Section VI-A. Then for each selected image, we extract 5 smooth patches with sizes ranging from 100×100 to 500×500. Variance is the criterion for the patch selection. For example, to extract the patch of size 100×100 from a certain image, we first divide it into overlapped patches of the same size, and then extract the one with the minimum variance. By repeating similar operations, we finally obtain 500 smooth patches. Let S_i be the size of the i-th patch. By considering the fact that the RANSAC estimation needs at least 4 correct matches, and the copy-move patches are generally no smaller than 1200 pixels [12], we can choose a new contrast threshold C by solving the following optimization problem

$$\begin{aligned} C^* = &\max C \\ &\text{s.t. } \frac{S_i}{1200} \times 4 \le N_i^C, \quad i = 1, \ldots, 500, \end{aligned} \tag{7}$$

where N_i^C is the number of SIFT keypoints associated with the i-th patch, triggered by the contrast threshold C. The constraint in (7) forces that every region of 1200 pixels generates at least 4 keypoints on average. We solve (7) by searching the solution from 4 to 0, with a small step size 0.01. We find that the optimal solution is slightly bigger than 0.1. By considering the existence of false matches, we set $C = 0.1$ in our experiments.

Fig. 4(c) shows that a much more number of keypoints are generated in the moon patch by reducing C from 4 to 0.1. To give a better understanding, Fig. 4(d) also provides the curve illustrating the relationship between the contrast threshold and the number of keypoints.

B. Resizing the Input Image

Solely lowering the contrast threshold cannot fully solve the problem of generating a sufficient number of keypoints, when the copy-move forgery is conducted on small regions. Our complementary strategy is to resize the input image by a factor of s prior to calculating SIFT keypoints. We have conducted extensive experiments, showing that enlarging the input image will highly increase the number of keypoints. The curve in Fig. 4(e) demonstrates that a much larger number of keypoints are generated as the scaling factor s increases. As two special cases, $s = 1$ and $s = 2$ correspond to the cases that SIFT keypoints are calculated from the octaves 0 and -1, respectively. Though a larger s triggers more keypoints, it will cause the keypoints to tightly cluster in the image plane, exacerbating the keypoint matching problems (to be discussed

[1] In the original implementation [23], C is set as 0.03. In the OpenCV implementation, C is set as 0.04 (the image pixel values are normalized into $[0, 1]$), while in the VLFEAT implementation [29], the commonly used contrast threshold is 4 (the image pixel values are in the range $[0, 255]$).

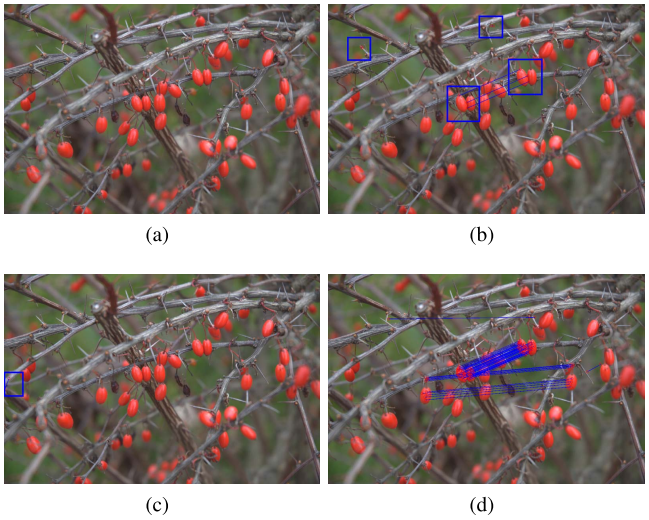


Fig. 5. Illustration of the matching problem. The matched keypoints in (b) and (c) are enclosed in blue boxes for better view. (a) The forgery image; (b) 5 matches are found through direct matching (total 2990 keypoints, contrast threshold 4 and resizing factor 1); (c) 1 match is found through direct matching (total 35601 keypoints, contrast threshold 0.1 and resizing factor 2); and (d) 54 matches are found through scale clustering (total 35601 keypoints, contrast threshold 0.1 and resizing factor 2).

in Section IV-A). In our experiments, we set $s = 2$ to achieve a good tradeoff.

Through the above two strategies, the number of keypoints can be significantly increased, even in smooth or small regions. However, such a massive number of keypoints also cause some critical matching problems (Section IV-A) and also increase the possibility of false matching. Thanks to the proposed matching strategy (Section IV-B and IV-C) and the new forgery localization technique (Section V), such problems can be greatly mitigated.

IV. Hierarchical Feature Point Matching

In the copy-move forgery detection scenario, the feature point matching operation aims to identify similar local regions in the image. In this Section, we first explain the problems of the point matching over a massive number of keypoints. A novel hierarchical feature point matching scheme is then proposed to alleviate such problems.

A. Keypoint Matching Problems

By resorting to the strategies proposed in Section III, a much larger number of SIFT keypoints are generated. One perhaps straightforwardly thinks that the number of matched pairs would also be highly increased accordingly. Unfortunately, we experimentally obtain an opposite result. Fig. 5 gives an example, where we can see, through lowering the contrast threshold and enlarging the input image, more than 10 times larger number of keypoints are generated; however, the number of matches drops from 5 to 1. The reason behind is that after lowering the contrast threshold, more keypoints are generated at the nearby locations or even the same location (but in different scales). As a result, the corresponding descriptors can be quite similar, thus violating the matching condition given in (5). As can be expected, this problem will become even worse if the image is enlarged. In this work, we call such phenomenon as *keypoint matching problem-I*.

Furthermore, since the computational complexity of the matching algorithm is $O(n^2)$, the significantly increased number of keypoints will exceedingly aggravate the computational burden. We call this problem as *keypoint matching problem-II*.

To mitigate the *keypoint matching problem-I and II* simultaneously, we now propose a novel hierarchical feature point matching algorithm. The framework of our matching scheme is depicted in Fig. 6, which consists of two components: 1) group matching via scale clustering; and 2) group matching via overlapped gray level clustering.

B. Group Matching via Scale Clustering

Recall that all SIFT keypoints are detected in the scale space, where the Gaussian images are grouped by octave as shown in Fig. 3. When lowering the contrast threshold C and enlarging the input image, the keypoints detected at different scales would be tightly clustered. This aggravates the *keypoint matching problem-I*. In this work, our solution is to maximumly separate the clustered keypoints detected at different scales. To this end, we propose to conduct the matching procedure within each single octave of lower scales *separately*, while *jointly* within multiple octaves of higher scales. The rationale is two-folds: 1) the number of keypoints in higher-scale octaves is much less than that of lower-scale ones, and hence do not suffer from the *keypoint matching problems*; 2) jointly matching the keypoints in higher-scale octaves achieves the robustness against large-scale resizing attack.

Specifically, let σ_k be the scale value of the keypoint $\mathbf{k}$, which can be readily obtained accompanying with calculating the SIFT keypoint (refer to Section II for details). Denote γ_i (a constant in the SIFT algorithm) as the scale value of the first DoG image in the i-th octave. In our work, the keypoints are clustered into three groups according to their scale values, which are respectively denoted by $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$. Formally,

$$\begin{aligned}
\mathcal{C}_1 &= \left\{ \mathbf{k}_i \,\middle|\, \gamma_1 \le \sigma_{k_i} < \gamma_2, i = 1, \ldots, n \right\}, \\
\mathcal{C}_2 &= \left\{ \mathbf{k}_i \,\middle|\, \gamma_2 \le \sigma_{k_i} < \gamma_3, i = 1, \ldots, n \right\}, \\
\mathcal{C}_3 &= \left\{ \mathbf{k}_i \,\middle|\, \sigma_{k_i} \ge \gamma_3, i = 1, \ldots, n \right\}.
\end{aligned} \tag{8}$$

Then, the matching procedure is conducted in $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ separately. Namely, for the first and second octaves, we apply the matching procedure within each single octave separately. While for the higher octaves, we do it in multiple octaves jointly. Through the scale clustering, the keypoints in different clusters are separated. We find that such strategy can greatly alleviate the *keypoint matching problem-I*. As an example, Fig. 5(d) shows that the number of matches is considerably increased through the group matching via scale clustering.

Remark: It should be pointed out that the benefits of the above scale clustering strategy are achieved at the cost of sacrificing the robustness against scaling attack to some extent. This is because no matches can be generated *across* the clusters $\mathcal{C}_1$,

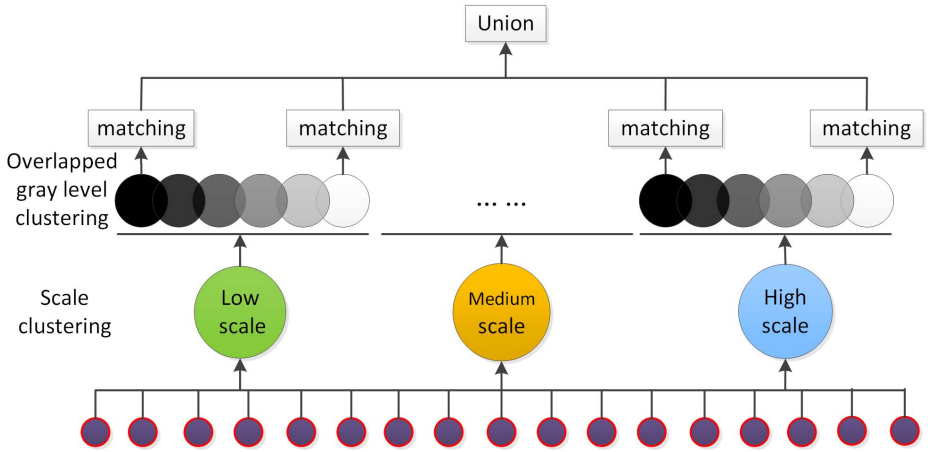


Fig. 6. Framework of the proposed hierarchical feature point matching.

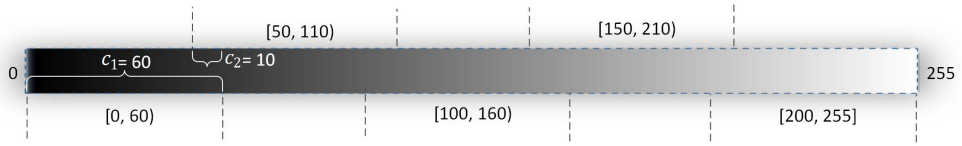


Fig. 7. An example of the overlapped clustering of gray level. Here $c_1 = 60$ and $c_2 = 10$.

$\mathcal{C}_2$ and $\mathcal{C}_3$. However, by considering the fact that $\mathcal{C}_3$ contains all the keypoints in higher octaves, the scaling robustness property can still be preserved to a certain degree. Furthermore, since each octave consists of several Gaussian images of different scales, the matching procedure conducted in $\mathcal{C}_1$ and $\mathcal{C}_2$ also keeps the robustness property against small scaling distortions. We experimentally find that the robustness of our scheme against scaling distortions is still one of the best among all the existing algorithms. Extensive experimental results to be provided in Section VI will validate this conclusion.

C. Group Matching via Overlapped Gray Level Clustering

To find the matching correspondence of the keypoint $\mathbf{k}$, the distance vector $\mathbf{d}$ is calculated over all the other keypoints in the same cluster ($\mathcal{C}_1$, $\mathcal{C}_2$ or $\mathcal{C}_3$) according to (5). Evidently, the computational burden would increase exceedingly as the number of keypoints grows. In our framework, an efficient matching algorithm becomes more essential, since a huge number of keypoints are generated in the feature extraction stage, especially for the first and the second octaves. To speed up the matching process, we further propose a matching strategy via overlapped gray level clustering. With the proposed algorithm, the matching procedure can be conducted in a much more efficient way without deleting original correct matches.

As can been seen from Section II-B, the previous matching strategy totally ignores the fact that, in the copy-move forgery detection scenario, the correctly matched pairs should locate in two similar local regions, hence have similar pixel values. By incorporating this prior knowledge, our method first clusters the keypoints according to their gray values, and then applies the matching algorithm (5) over each cluster, rather than the detected keypoints entirely.

However, brutally dividing keypoints into separate clusters may delete many correct matches *across* different clusters. To tackle this issue, we propose to partition keypoints into overlapped clusters. Namely, the adjacent clusters will share a part of keypoints. Specifically, we first uniformly divide the vector $[0, 1, .., 255]$ into L overlapped sub-levels, with a step size c_1 and an overlapped size c_2 ($c_1 > c_2$). Mathematically, we have

$$L = \left\lceil \frac{255 - c_1}{c_1 - c_2} \right\rceil + 1. \tag{9}$$

An illustration of the overlapped clustering is given in Fig. 7.

Let $\mathcal{C}_p = \{\mathcal{C}_{p,1}, \mathcal{C}_{p,2}, \ldots, \mathcal{C}_{p,L}\}$, $p \in \{1, 2, 3\}$, where $\mathcal{C}_{p,i}$ records all the keypoints in $\mathcal{C}_p$ with gray values belonging to the i-th sub-level. Formally,

$$\mathcal{C}_{p,i} = \left\{ \mathbf{k}_j \,\middle|\, a_i \le Gr(\mathbf{k}_j) < b_i, \mathbf{k}_j \in \mathcal{C}_p \right\}, \tag{10}$$

where

$$a_i = (i-1) \times (c_1 - c_2), \quad b_i = \min(a_i + c_1, 255),$$

and $Gr(\cdot)$ calculates the gray value associated with a keypoint, by taking the average over all the points in the 3×3 region centered at that keypoint. Then, the matching procedure is conducted over each cluster $\mathcal{C}_{p,i}$ independently. Compared with the original matching strategy described in Section II-B, for each keypoint $\mathbf{k} \in \mathcal{C}_{p,i}$, we only need to compute the distances between $\mathbf{k}$ and the other keypoints in $\mathcal{C}_{p,i}$, where the number is much smaller than n. This makes the proposed matching procedure much more efficient.

Define $\mathcal{P}_{p,i}$ as the set containing the matched pairs over $\mathcal{C}_{p,i}$, and $\mathcal{P}$ as the set recording all the matched pairs. Mathematically, $\mathcal{P}$ can be calculated as the union of the matched pairs over all the clusters.

$$\mathcal{P} = \bigcup \mathcal{P}_{p,i}, \quad p \in \{1, 2, 3\}, \quad i = 1, \ldots, L. \tag{11}$$

To eliminate any potential confusions, we hereafter use $(\mathbf{k}, \mathbf{k}')$ to denote a matched pair having no matching order,

namely, $(\mathbf{k}, \mathbf{k}') \triangleq (\mathbf{k}', \mathbf{k})$. On the other hand, we use $< \mathbf{k}, \mathbf{k}' >$ to denote a matched pair having the matching order, which means $\mathbf{k}$ is matched to $\mathbf{k}'$ through a specific affine transformation. Obviously, $< \mathbf{k}, \mathbf{k}' > \neq < \mathbf{k}', \mathbf{k} >$. It should be noted that all the matched pairs in $\mathcal{P}$ have no matching order. This is because the matching process cannot ensure a consistent matching direction (from the source region to the forged one, or vice versa), as it takes no consideration of the location information. This phenomenon will be further discussed in Section V-B.

V. Iterative Forgery Localization

Forgery localization in the copy-move forgery detection scenario is to identify the duplicated regions in dense fields. For the keypoint-based image copy-move forgery detection algorithms, there are two problems when localizing the forged regions:

1) The homography is generally not unique when multiple clones are conducted, and the number of duplicated regions is unknown;
2) All the matched pairs typically have no matching order, and hence the forged points and the corresponding original ones are not separated through the matching process.

The above problems restrict the use of the RANSAC algorithm [24], which works only for a single homography estimation. Further, the matched pairs fed into RANSAC should have the matching order, otherwise they could be treated as outliers, thus making an inaccurate estimation.

To tackle these problems, two strategies are widely utilized in previous literatures: 1) keypoint-clustering-based algorithms such as [3] and [15] proposed to first cluster the matched keypoints according to their locations, so that the forged points and the corresponding original ones could be separated into different clusters; then the matches linking each two clusters were assigned a consistent matching order (from one cluster to the other cluster) [3]; and 2) keypoint-segmentation-based algorithms such as [5] suggested to first segment the whole image into small non-overlapped regions, and then conducted the matching procedure between each two segmented regions. Regions are regarded as the copy-move correspondences if they contain a sufficient number of matched points.

However, both the keypoint-clustering-based algorithms and the keypoint-segmentation-based algorithms suffer from serious problems. Due to the truth that the copy-move forgery can be conducted in arbitrary ways, it is difficult (even impossible) to find a *universally good* clustering/segmentation algorithm and associated parameters applicable for all images.

For the keypoint-clustering-based algorithms, one critical problem is that the number of true clusters is unknown when multiple clones are conducted; properly performing the clustering in this case is generally very difficult. Furthermore, the keypoint-clustering-based algorithms perform poorly when the locations of copy-move regions are close in the image plane [16]. This is because the forged region and the original region are very likely to be grouped into the same cluster.

For the keypoint-segmentation-based algorithms, apart from the high computational cost of the segmentation process when the image is of large size, it is difficult to find *universally* good parameters of the segmentation algorithms. For example, it is expected to have different segmentation parameters for the images shown in Fig. 8(a) and Fig. 8(b), avoiding to segment the moon as a single region. Adjusting the segmentation parameters however is challenging in practice, since images usually contain both smooth and highly structured regions, and we generally have no prior information whether and where the copy-move forgery is conducted. This makes the second strategy fail often in real applications.

In this Section, we propose an iterative localization method without involving any clustering and segmentation procedures. Our proposed scheme is designed by fully exploiting the robustness properties (including the dominant orientation and scale information) and the color information of each matched keypoint. As will be clear shortly, our scheme achieves very high accuracy of the forgery localization. The framework of our proposed forgery localization scheme is depicted in Fig. 2 (phase 3). Typically, our method comprises four steps:

Step 1): Removal of the isolated matched pairs;
Step 2): Estimation of the local homography;
Step 3): Homography validation and inliers selection using the dominant orientation;
Step 4): Forgery localization using the scale and color information.

A. Removal of Isolated Matched Pairs

In the copy-move forgery detection scenario, one prior knowledge is that the forgery is conducted in a contiguous shape. This implies that correctly matched keypoints should not be isolated in a local region. In order to reduce the false alarm rate, we first remove those isolated matched pairs. This is helpful especially when the image contains roads, windows and other objects with proximately periodic changes, where several isolated false matches may still satisfy the same homography.

For each matched pair $(\mathbf{k}, \mathbf{k}') \in \mathcal{P}$, where $\mathcal{P}$ is defined in (11), let N_k and $N_{k'}$ be the numbers of matched keypoints with location distances to $\mathbf{k}$ and $\mathbf{k}'$ smaller than a threshold T_{iso} ($T_{iso} = 100$ in our implementation). In this work, we discard those matched pairs satisfying

$$\max\{N_k, N_{k'}\} < N_{iso}, \tag{12}$$

where $N_{iso} = 2$ in our implementation. Define $\mathcal{M}$ as the set consisting of all the survived matched pairs. Namely,

$$\mathcal{M} = \left\{ (\mathbf{k}, \mathbf{k}') \middle| \ \max\{N_k, N_{k'}\} \geq N_{iso}; (\mathbf{k}, \mathbf{k}') \in \mathcal{P} \right\}. \tag{13}$$

B. Estimation of the Local Homography

In the Step 2), an affine matrix will be estimated using only a part of matched pairs from two contiguous local regions. Specifically, we first randomly select a matched pair $(\mathbf{k}, \mathbf{k}') \in \mathcal{M}$, where $\mathcal{M}$ is defined in (13). Let $\mathcal{C}_k$ and $\mathcal{C}_{k'}$ record all the matched keypoints close to $\mathbf{k}$ and $\mathbf{k}'$, respectively. Formally,

$$\begin{aligned} \mathcal{C}_k &= \left\{ \mathbf{p} \middle| \ \forall \mathbf{p} \in \mathcal{M}_{keys}, Dis(\mathbf{p}, \mathbf{k}) < T_d \right\}, \\ \mathcal{C}_{k'} &= \left\{ \mathbf{p} \middle| \ \forall \mathbf{p} \in \mathcal{M}_{keys}, Dis(\mathbf{p}, \mathbf{k}') < T_d \right\}, \end{aligned} \tag{14}$$

where $\mathcal{M}_{keys}$ contains all the matched keypoints in $\mathcal{M}$, i.e.,

$$\mathcal{M}_{keys} = \left\{ \mathbf{k} \middle| \ \exists \mathbf{k}', \text{ s.t. } (\mathbf{k}, \mathbf{k}') \in \mathcal{M} \right\}, \tag{15}$$

T_d is a hyper-parameter ($T_d = 100$ in our implementation) and $Dis(\cdot)$ computes the Euclidean distance of two keypoints in the image plane.

Then we construct a set $\mathcal{M}_k$ containing all the matched pairs close to $(\mathbf{k}, \mathbf{k}')$, i.e.,

$$\mathcal{M}_k = \left\{ < \mathbf{k}, \mathbf{k}' > \middle| \ \mathbf{k} \in \mathcal{C}_k \wedge \mathbf{k}' \in \mathcal{C}_{k'}; (\mathbf{k}, \mathbf{k}') \in \mathcal{M} \right\}. \tag{16}$$

Here we need to emphasize that all the matched pairs in $\mathcal{M}_k$ have a consistent matching order (from $\mathcal{C}_k$ to $\mathcal{C}_{k'}$). On the other hand, because all the matched pairs in $\mathcal{M}_k$ are generated from two local contiguous regions, it is reasonable to assume that they obey the same homograhy $\mathbf{H}_k$. Then our strategy is immune from the two problems discussed in the beginning of Section V. Thus we can use the RANSAC algorithm to estimate the homography $\mathbf{H}_k$ between the correspondences of the matched pairs in $\mathcal{M}_k$. Note that in [16], it was suggested to estimate a set of affine matrices, each of which was also calculated by a part of matched pairs (three randomly selected *neighbored* pairs specifically). However, their goal is to cluster the matched pairs in a *conceptual space*. Different from [16], we estimate the affine matrix using all the matched pairs from two contiguous local regions, which will be further refined using all the inliers (see Section V-C below). Furthermore, no clustering procedure is involved in our proposed scheme.

C. Homography Validation and Inliers Selection Using the Dominant Orientation

It is a well-known fact that we cannot fully trust the result of RANSAC especially when the number of inliers is limited [23]. To discard those inaccurate estimations, we propose a novel homography validation and inlier selection approach by capitalizing the dominant orientation associated with each keypoint. As discussed in Section II, the dominant orientation plays a vital role in the SIFT algorithm for the rotation invariance. Let θ_k be the dominant orientation of the keypoint $\mathbf{k}$, which can be readily obtained accompanying with the SIFT keypoint extraction. Obviously, for each correctly matched pair (inlier) $< \mathbf{k}, \mathbf{k}' >$, the offset of the dominant orientations $\theta_{k'} - \theta_k$ should be compatible with the estimated affine homography $\mathbf{H}_k$. Though some incorrect matches may occasionally obey the same homography transformation (this happens often when the number of matches is limited), their dominant orientations can hardly be compatible with $\mathbf{H}_k$ simultaneously. This provides a valuable criteria for rejecting inaccurate estimations and selecting inliers. Specifically, the homography $\mathbf{H}_k$ can be written as

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{17}$$

where $\mathbf{t} = [t_x, t_y]^T$ is the transition vector, and $\mathbf{A}$ is a 2×2 nonsingular matrix, which can always be decomposed as

$$\begin{aligned} \mathbf{A} &= \mathbf{U}\mathbf{D}\mathbf{V}^T = (\mathbf{U}\mathbf{V})^T(\mathbf{V}\mathbf{D}\mathbf{V}^T) \\ &= R(\theta_H)R(-\phi_H)\mathbf{D}R(\phi_H), \end{aligned} \tag{18}$$

where $\mathbf{U}$ and $\mathbf{V}$ serve as the left-singular and right-singular vectors, respectively; $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2)$ represents the non-isotropic scale factor [30]; $R(\theta_H)$ and $R(\phi_H)$ denote the rotation operators with parameters θ_H and ϕ_H, respectively. Formally,

$$R(\theta_H) = \begin{bmatrix} \cos(\theta_H) & -\sin(\theta_H) \\ \sin(\theta_H) & \cos(\theta_H) \end{bmatrix} = (\mathbf{U}\mathbf{V})^T. \tag{19}$$

Note that copy-move patches can be rotated either clockwise or anticlockwise. For consistency, we map the value of θ_H into the range $[0, 2\pi)$, which can be readily calculated by

$$\theta_H = \begin{cases} \cos^{-1}(R_{11}), & \text{if } R_{11} \geq 0 \wedge R_{21} \geq 0 \\ & \text{or } R_{11} < 0 \wedge R_{21} > 0 \\ & \text{if } R_{11} \leq 0 \wedge R_{21} \leq 0 \\ 2\pi - \cos^{-1}(R_{11}), & \text{or } R_{11} > 0 \wedge R_{21} < 0, \end{cases} \tag{20}$$

where $R_{11} = \cos(\theta_H)$ and $R_{21} = \sin(\theta_H)$.

Finally, we validate the correctness of the estimated homography $\mathbf{H}_k$ by checking the offset between $\theta_{k'} - \theta_k$ and θ_H. To this end, we define a function

$$f(\mathbf{k}, \mathbf{k}', \mathbf{H}_k) = |\theta_{k'} - \theta_k - \theta_H|. \tag{21}$$

Clearly, for an accurately estimated $\mathbf{H}_k$ and a correctly matched pair $< \mathbf{k}, \mathbf{k}' >$, $f(\mathbf{k}, \mathbf{k}', \mathbf{H}_k)$ is close to zero.

Let $\hat{\mathcal{M}}_k$ be the inlier set returned by the RANSAC algorithm compatible with $\mathbf{H}_k$. Obviously $\hat{\mathcal{M}}_k \subseteq \mathcal{M}_k$. To avoid the existence of wrongly accepted inliers by chance, we accept the homography $\mathbf{H}_k$ if and only if over 90% matches in $\hat{\mathcal{M}}_k$ satisfy the following condition

$$f(\mathbf{k}, \mathbf{k}', \mathbf{H}_k) \leq T_\theta, \quad \forall < \mathbf{k}, \mathbf{k}' > \in \hat{\mathcal{M}}_k, \tag{22}$$

where T_θ is a predefined parameter ($T_\theta = 15$ in our implementation). Otherwise, we reject $\mathbf{H}_k$.

Upon having an accurately estimated $\mathbf{H}_k$, we can also use the dominant orientation information to select the inliers (denoted by $\mathcal{M}_H$) compatible with $\mathbf{H}_k$, over all the matched pairs. Apparently, for each correctly matched pair $< \mathbf{k}, \mathbf{k}' >$, we have

$$\begin{pmatrix} x_k' \\ y_k' \\ 1 \end{pmatrix} \approx \mathbf{H}_k \begin{pmatrix} x_k \\ y_k \\ 1 \end{pmatrix}. \tag{23}$$

For the sake of notation simplicity, throughout this Section, we interchangeably let $\mathbf{k}$ be a keypoint of four dimensions $(x_k, y_k, \sigma_k, \theta_k)$, or the coordinates $(x_k, y_k, 1)$ when there is no ambiguity.[2] Then $\mathcal{M}_H$ can be computed by

$$\begin{aligned} \mathcal{M}_H = \Big\{ < \mathbf{k}, \mathbf{k}' > \Big| \ & ||\mathbf{H}_k\mathbf{k} - \mathbf{k}'||_2^2 < \epsilon, \\ & f(\mathbf{k}, \mathbf{k}', \mathbf{H}_k) \leq T_\theta; (\mathbf{k}, \mathbf{k}') \in \mathcal{M} \Big\}, \end{aligned} \tag{24}$$

where $\mathcal{M}$ is defined in (13).

Finally, we further refine $\mathbf{H}_k$ using all the obtained inliers. This can be readily converted into the following optimization problem

$$\hat{\mathbf{H}}_k = \arg\min_{\hat{\mathbf{H}}_k} \sum_{<\mathbf{k}, \mathbf{k}'> \in \mathcal{M}_H} ||\hat{\mathbf{H}}_k\mathbf{k} - \mathbf{k}'||_2^2. \tag{25}$$

[2]The third dimension of $(x_k, y_k, 1)$ is a constant used for the affine transformation.

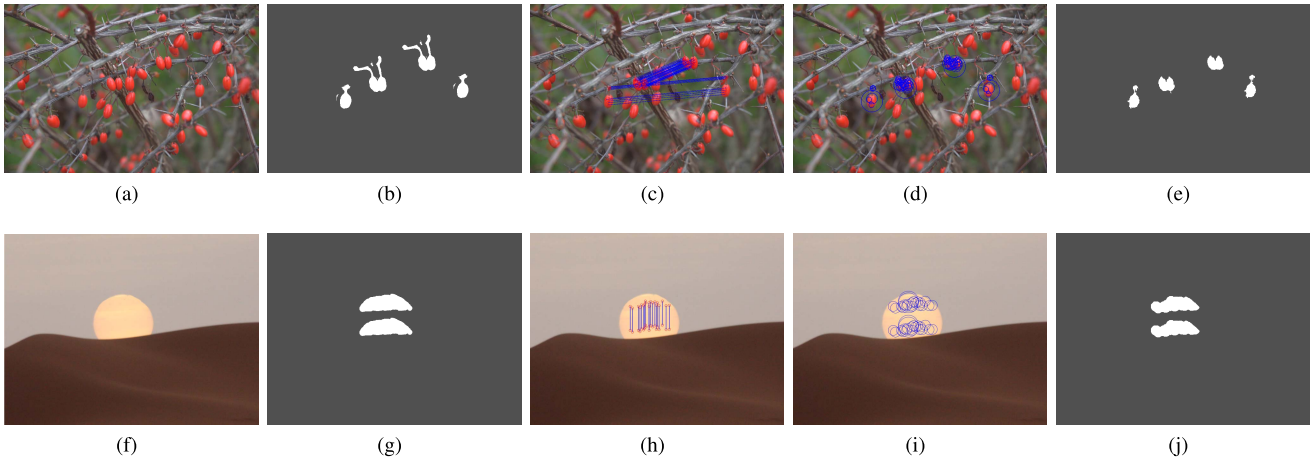


Fig. 8. Copy-move detection results for the copy-move forgeries only involving small (a) and smooth (f) regions. (a),(f) forged images, (b),(g) ground-truth tampered regions, (c),(h) matches (only inliers) obtained by our proposed scheme, (d),(i) suspicious tampered regions, and (e),(j) final detected tampered regions.

Compared with $\mathbf{H}_k$ estimated using only a part of matches, $\hat{\mathbf{H}}_k$ is estimated over all the inliers obeying the same transformation, which should be more accurate. As will be detailed in the next subsection, both $\mathcal{M}_H$ and $\hat{\mathbf{H}}_k$ play important roles for the forgery localization.

D. Forgery Localization in Dense Fields

Note that $\mathcal{M}_H$ defined in (24) only contains matches between points sparsely sampled in the image plane. In this subsection, we discuss the problem of localizing the forgeries in dense fields. For those segmentation-based methods, the forgery localization in dense fields can be simply conducted by merging the neighboring segmented regions containing a sufficient number of matched keypoints. However, as aforementioned, it is practically challenging to find a universally good segmentation algorithm and the associated parameters applicable for all images. Below we propose a new algorithm for the forgery localization in dense fields, without involving any troublesome clustering/segmentation procedures. Specifically, our method is composed of two phases: 1) construct the suspicious regions according to the scale information of each inlier; and 2) refine the suspicious regions by validating the consistency of the color information.

As discussed in Section II, the descriptor of the keypoint $\mathbf{k}$ is calculated by encoding its surrounding information in the scale space, where the size of the support region in the image plane positively correlates to its scale value. In other words, more larger support regions in the image plane are assigned to the keypoints in higher scales. Hence, in the first phase, we construct a local circular suspicious region for each inlier in $\mathcal{M}_H$, where the radius w.r.t. the keypoint $\mathbf{k}$ is given by

$$r_k = \alpha\sigma_k. \tag{26}$$

Here σ_k is the scale value of $\mathbf{k}$, and α is a hyper-parameter ($\alpha = 16$ in our implementation). Since all the matched pairs in $\mathcal{M}_H$ have the matching order, two suspicious regions $\mathcal{S}$ and $\mathcal{S}'$ can be readily obtained. Some examples are shown in Figs. 8(d),(i) and Figs. 9(d),(i).

In the phase 2), we refine the suspicious regions by exploiting the color information. Specifically, for each point in $\mathcal{S}$, we transform it as

$$\mathbf{k}_* = \hat{\mathbf{H}}_k\mathbf{k},\ \ \mathbf{k} \in \mathcal{S}. \tag{27}$$

Assume that the RGB components of a point $\mathbf{k}$ are respectively denoted by $R(\mathbf{k})$, $G(\mathbf{k})$ and $B(\mathbf{k})$. In (27), $\mathbf{k}$ and $\mathbf{k}_*$ are regarded as the copy-move points if their color values are similar. Let $\mathcal{Q}_1$ record all the copy-move points calculated based on $\mathcal{S}$, which is computed by

$$\mathcal{Q}_1 = \Big\{\mathbf{k}, \mathbf{k}_* \Big| \max\Big(|R(\mathbf{k}) - \overline{R(\mathbf{k}_*)}|, |G(\mathbf{k}) - \overline{G(\mathbf{k}_*)}|, |B(\mathbf{k}) - \overline{B(\mathbf{k}_*)}|\Big) < T_{rgb}; \mathbf{k} \in \mathcal{S}\Big\}. \tag{28}$$

Here $\overline{R(\mathbf{k})} = \frac{1}{Z}\sum_{\mathbf{k}\in\Omega(\mathbf{k})} R(\mathbf{k})$, where $\Omega(\mathbf{k})$ is a 3×3 patch centered at $\mathbf{k}$ and Z is the normalization factor; $\overline{G(\mathbf{k})}$ and $\overline{B(\mathbf{k})}$ can be calculated in a similar way as $\overline{R(\mathbf{k})}$; and T_{rgb} is a predefined parameter ($T_{rgb} = 10$ in our implementation). For each point in $\mathcal{S}'$, we transform it in an opposite way, i.e.,

$$\mathbf{k}'_* = \hat{\mathbf{H}}_k^{-1}\mathbf{k}',\quad \mathbf{k}' \in \mathcal{S}'. \tag{29}$$

Similarly, define $\mathcal{Q}_2$ as a set containing all the copy-move points calculated based on $\mathcal{S}'$, i.e.,

$$\mathcal{Q}_2 = \Big\{\mathbf{k}', \mathbf{k}'_* \Big| \max\Big(|R(\mathbf{k}') - \overline{R(\mathbf{k}'_*)}|, |G(\mathbf{k}') - \overline{G(\mathbf{k}'_*)}|, |B(\mathbf{k}') - \overline{B(\mathbf{k}'_*)}|\Big) < T_{rgb}; \mathbf{k}' \in \mathcal{S}'\Big\}. \tag{30}$$

Let $\mathbf{B}$ be the binary map of the forgery localization, with the same size of the input image. We use the number '1' to label the forged locations and the corresponding genuine ones, while '0' for the remaining parts. All the elements in $\mathbf{B}$ are initialized with 0s. Then, $\mathbf{B}$ is updated by marking those points belonging to $\mathcal{Q}_1 \cup \mathcal{Q}_2$, which can be simply written as

$$\mathbf{B}(\mathcal{Q}_1 \cup \mathcal{Q}_2) = \mathbf{1}. \tag{31}$$

Note that through the Steps 2)-4) presented above, we can at most reveal the copy-move regions satisfying one homography transformation. However, the homography may not be unique when multiple clones are conducted. Our localization

algorithm operates in an iterative manner as shown in Fig. 2. Specifically, in our method, Steps 2)-4) will be repeated for K iterations ($K = 15$ in our experiment). Within each iteration, we only estimate one homography matrix using a part of matched pairs.

After finishing all the iterations, we can finally generate the detected forgery regions by sequentially applying the following two procedures on **B**: 1) remove small regions; and 2) fill small gaps through the morphological close operation. An image is regarded as genuine if and only if all the elements in **B** are 0; otherwise, it is regarded as a forged one.

VI. Experiment Results

In this Section, we evaluate the proposed copy-move forgery detection method. The detection performance is measured at both the image level and the pixel level. At the image level, we focus on the ability that an image can be correctly recognized as forged or genuine; at the pixel level, we analyze the performance for forgery localization accuracy. To embrace the concept of reproducible research [33], the code of our paper is available at https://github.com/YuanmanLi/FE-CMFD-HFPM.

The first two measures are True Positive Rate (*TPR*) and False Positive Rate (*FPR*), which are respectively defined as

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{TN + FP}. \tag{32}$$

At the image level, *TP* (true positive), *TN* (true negative), *FN* (false negative) and *FP* (false positive) in the above definition count the number of correctly detected forged images, correctly detected genuine images, undetected forged images and wrongly detected genuine images, respectively. At the pixel level, *TP* denotes the number of correctly detected forged pixels. *FN* is the number of falsely undetected forged pixels, and *FP* is the number of falsely detected genuine pixels. Obviously, a well-designed copy-move forgery detector should keep *TPR* high and *FPR* low simultaneously. In addition to *TPR* and *FPR*, we also report the F_1 score, which gives the synthetical performance through a single value. Mathematically,

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{33}$$

In this work, we use F-image and F-pixel to represent the F_1 scores at the image level and the pixel level, respectively.

Considering the practical usefulness, computational complexity is another key performance measure. In our experiments, we also compare the average computational cost of different algorithms. All the experiments are conducted on a desktop equipped with Core-i7 and 8-GB RAM, operating in single-thread modality.

A. Datasets

Six test datasets, i.e., FAU [6], GRIP [12], MICC-F220 [3], MICC-F600 [16], CMH [15] and COVERAGE [34] are used to demonstrate the effectiveness of our scheme. Note that the tampered patches from FAU and GRIP are not further processed, while the ones from the other four datasets could be applied with different attacks (transforms) in a realistic fashion, such as rotation, scaling, or a combination of them. More details can be found below.

1. FAU: this dataset comprises 48 original images and 48 corresponding forged images with realistic copy-move manipulations, where the average resolution is about 3000×2300.
2. GRIP: this dataset contains 80 original images and 80 realistic copy-move forgeries, having an identical size 768×1024. It is worthy to note that some tampered patches in GRIP are very smooth, which are challenging for those copy-move forgery detectors based on sparse sampling (e.g., SIFT).
3. MICC-F220: this dataset consists of 110 tampered images and 110 untampered images, with resolutions ranging from 722×480 to 800×600.
4. MICC-F600: this dataset is composed by 160 tampered images and 440 original images, where the image resolution varies from 800×533 to 3888×2592.
5. CMH: this dataset includes 108 realistic cloning images, with resolutions varying from 845×634 to 1296×972.
6. COVERAGE[3]: this dataset has 100 original-forged image pairs, with an average resolution 400×486. Each image contains similar-but-genuine objects. In our experiments, we use a subset consisting of 91 original-forged image pairs, excluding the 9 image pairs where the released ground-truth maps are incorrect.

B. Detection Results on Six Datasets

We first evaluate the proposed scheme under plain copy-move attack, namely no further attack is performed on the copy-move regions. The experiments are conducted over FAU and GRIP datasets. Two examples are depicted in Fig. 8, where the first row and the second row show the cases that the copy-move forgeries are conducted in small regions and smooth regions, respectively. For the forged image shown in Fig. 8(a), the ratio of the smallest copied region over the whole image is about 0.27%, and the variance of the copied region in Fig. 8(f) is about 3.9. It can be easily observed from Fig. 8(c) and Fig. 8(h), that our proposed scheme can obtain a sufficient number of correct matches even in the small or smooth regions. Fig. 8(d) and Fig. 8(i) show the resulting suspicious regions by using the scale information of the inliers, which are further refined according to the color information validation, leading to the final detected tampered regions shown in Fig. 8(e) and Fig. 8(j). We can notice that our detection results are accurate compared with the ground-truth.

Table I lists the results on FAU and GRIP datasets obtained by different copy-move forgery detection methods, including keypoint-clustering-based [3], [15], keypoint-segmentation-based [5], [32], block-based [6], [12], [31] and our proposed approaches. To save space, we use F_I and F_P in Table I and Table II to denote F-image and F-pixel, respectively. For the algorithms [6] and [31], we use the code implemented by Christlein *et al.* [6], and the implementations of other algorithms are provided by the original authors. We can readily observe that the proposed algorithm presents

[3]https://github.com/wenbihan/coverage

TABLE I

TPR(%), *FPR*(%) (IMAGE LEVEL), $F_I(\%)$, $F_P(\%)$ AND CPU(SECOND) ON FAU AND GRIP DATASETS

methods	FAU					GRIP				
	TPR	*FPR*	F_I	F_P	CPU	*TPR*	*FPR*	F_I	F_P	CPU
Amerini [3]	66.67	10.42	75.29	-	15.2	70.00	20.00	73.68	-	2.1
Cozzolino [12]	97.92	8.33	94.95	93.79	165.2	98.75	8.75	95.18	92.99	14.8
Li [5]	72.92	22.92	74.47	74.47	4946.3	83.75	35.00	76.57	27.74	353.0
Bravo [31]	97.92	6.25	95.92	70.02	2689.8	100	0	100	84.82	39.4
Zernike [6]	100	10.42	95.05	89.33	2876.2	100	5.00	97.56	86.18	40.1
GoDeep [15]	97.92	39.58	82.46	88.24	1071.6	100	38.75	83.77	66.62	11.7
Zandi [32]	100	52.08	79.34	86.07	468.2	100	33.75	85.56	66.44	25.7
Proposed	100	2.08	98.97	94.28	86.6	100	0	100	94.66	13.9

TABLE II

TPR(%), *FPR*(%) (IMAGE LEVEL), $F_I(\%)$, $F_P(\%)$ AND CPU(SECOND) ON MICC-F220, MICC-F600, CMH+GRIPori AND COVERAGE DATASETS

methods	MICC-F220				MICC-F600					CMH+GRIPori					COVERAGE				
	TPR	*FPR*	F_I	CPU	*TPR*	*FPR*	F_I	F_P	CPU	*TPR*	*FPR*	F_I	F_P	CPU	*TPR*	*FPR*	F_I	F_P	CPU
Amerini [3]	98.18	9.09	94.74	3.5	68.13	15.68	64.50	-	11.4	70.37	20.00	75.76	-	3.4	85.71	54.95	71.23	-	1.6
Amerini [16]	-	-		-	81.60	7.27	81.11	-	-	-	-	-	-	-	-	-	-	-	
Cozzolino [12]	84.55	17.27	83.78	5.3	96.25	5.91	90.59	91.40	72.3	92.59	8.75	93.02	88.10	13.1	59.34	21.98	65.45	64.84	2.3
Li [5]	70.91	17.27	75.36	111.6	88.10	13.8	77.90	84.14	2623.7	95.37	35.00	86.19	44.00	685.2	87.91	63.74	69.87	57.09	132.6
Bravo [31]	19.09	6.36	30.43	17.4	80.63	7.50	80.12	63.94	2687.8	73.15	0	84.49	53.77	35.7	50.55	0	67.15	44.92	8.9
Zernike [6]	20.91	6.36	32.86	18.7	63.13	16.59	60.48	61.43	2866.9	66.67	5.00	78.26	53.05	36.9	46.15	15.38	57.14	37.99	9.2
GoDeep [15]	45.45	41.82	48.54	4.1	98.75	55.0	56.43	78.52	523.7	95.37	38.75	85.12	64.09	18.1	91.21	70.33	69.75	59.88	5.9
Zandi [32]	78.18	48.18	69.08	16.6	94.38	50.91	56.45	75.29	193.7	85.19	33.75	81.06	55.97	60.3	76.92	71.43	61.95	49.48	16.4
Proposed	100	1.82	99.10	3.0	97.50	5.68	91.50	91.80	45.3	96.30	0	98.11	90.61	11.2	80.22	41.76	72.28	66.26	2.3

superior performance at both the image level and the pixel level against the competing methods. Specifically, for GRIP dataset, our algorithm perfectly differentiates all the forged images and genuine ones; for FAU dataset, only one image is miss-classified. To our best knowledge, these are the best results ever reported. Besides the detection accuracy, we also present the average CPU-time for each algorithm in Table I. To make a fair comparison, all the methods operate in single-thread modality. We can see that the keypoint-clustering-based technique [3] is the most efficient. Its high efficiency is mainly due to the following two factors: 1) it doest not deal with the forgery localization problem, and hence, it can only give the image level detection results; and 2) the number of features extracted is often very limited without specifically handling smooth or small copy-move forgeries. Clearly, it would make the matching procedure very efficient even without any speed-up strategies; but at the cost of severe performance degradation, as can be seen from Table I. By resorting to the proposed hierarchical feature point matching and the iterative localization strategy, our method is also very efficient, and the computational efficiency gains over the other methods (except [3]) are quite remarkable, especially when images are of high resolutions. For example, our method is over 50 times faster than [5], 30 times faster than [6], [31], 10 times faster than [15], and 5 times faster than [32] on FAU dataset.

In addition to the plain copy-move forgery detection, we also evaluate our proposed method over the datasets MICC-F220, MICC-F600, CMH and COVERAGE, where some of the copy-move regions are further distorted by rotation, scaling, or a combination of them. Note that the original CMH dataset contains forgery images only. In this work, we adopt a combined dataset named CMH+GRIPori, consisting of all the 108 forgeries from CMH, and the 80 genuine images from GRIP dataset. The results over these four datasets are summarized in Table II.[4] We observe that our method achieves much better performance than the competing ones.[5] Due to the scaling and rotation attacks, the performance of the block-based algorithms [6], [31] drops rapidly on these four datasets. This phenomenon will be further discussed in the next subsection. We can also see that all the algorithms perform poorly on COVERAGE dataset, where each image contains very similar-but-genuine objects. GoDeep [15] achieves the best *TPR* performance over COVERAGE; but also leads to very high false positive rate (over 70%). Compared with the other algorithms, our method obtains the best F_1 measures at both the image level and the pixel level on COVERAGE. Fig. 9 also provides the representative results of two examples. We can see that our proposed scheme can accurately localize the forgeries even in the presence of scaling and rotation attacks.

C. Evaluation on Different Phases

In this subsection, we investigate how our matching scheme and localization approach affect the performance of the proposed algorithm.

Table III reports the results over FAU dataset with and without the hierarchical matching strategy proposed in Section IV-B and Section IV-C. For simplicity, we use 'matching' to denote the traditional matching algorithm described in Section II-B,[6] 'matching+S' to represent the group matching method using only scale clustering, and 'matching+SG'

[4] MICC-F220 does not release the ground-truth maps, so we do not provide the pixel level performance on this dataset.

[5] As the source code of [16] is not available, the data presented in Table II are extracted from [16, Table 2].

[6] We use the code provided by [3]. http://lci.micc.unifi.it/labd/2015/01/copy-move-forgery-detection-and-localization.

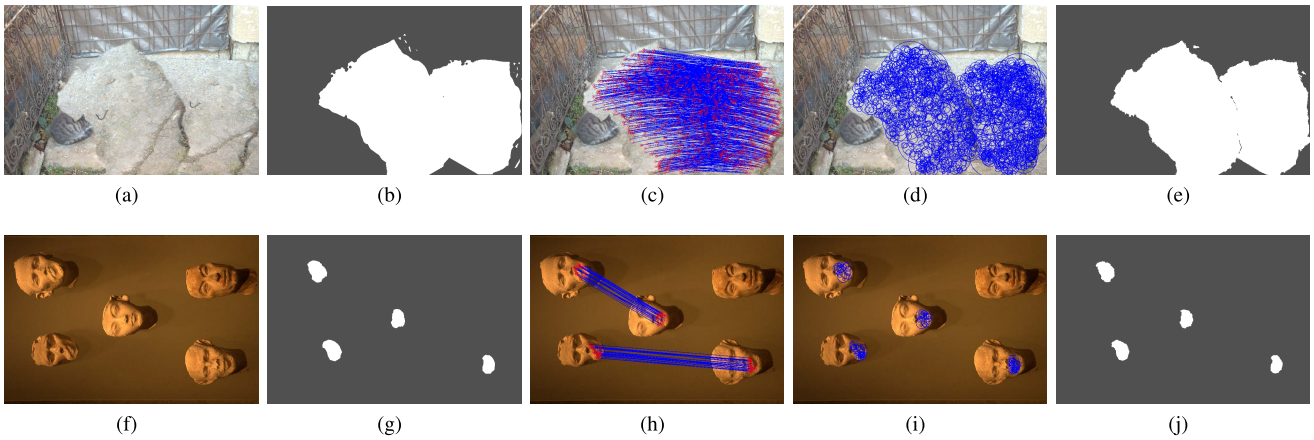


Fig. 9. Copy-move detection results for some images from MICC-F600, where the tampered regions are rotated 30° and scaled by a factor of 1.2. (a),(f) forged images, (b),(g) ground-truth tampered regions, (c),(h) matches (only inliers) obtained by our proposed scheme, (d),(i) suspicious tampered regions, and (e),(j) final detected tampered regions.

TABLE III

AVERAGE NUMBER OF KEYPOINTS, MATCHES AND CPU (SECOND) OF DIFFERENT MATCHING SCHEMES

Methods	Feature Extraction		Matching		Total CPU
	#keypoints	CPU	#matches	CPU	
matching	41150	48.6	38	176.0	224.6
matching+S			482	91.7	140.3
matching+SG			509	18.2	66.8

TABLE IV

F-IMAGE(%), F-PIXEL(%) AND CPU(SECOND) OF DIFFERENT LOCALIZATION ALGORITHMS

Methods	F-image	F-pixel	CPU
matching+ZNCC	93.20	76.37	386.3
Proposed	98.97	94.28	13.96

to refer our proposed hierarchical matching method adopting both scale clustering and overlapped gray level clustering. Based on the results in Table III, we can draw the following conclusions. Firstly, the traditional matching algorithm heavily suffers from the *keypoint matching problem-I and II* as stated in Section IV-A. We can observe that 'matching' can only find a limited number of matches though a great many keypoints are generated; further, the matching procedure is of very high computational complexity. Secondly, a large number of matches are found by resorting to the scale clustering strategy, showing its effectiveness to migrate the *keypoint matching problem-I*. Finally, it can be readily figured out that the computational cost is significantly reduced by the overlapped gray level clustering. We can see that 'matching+SG' is more than 5 times faster than 'matching+S', and approximately 10 times faster than 'matching'.

Estimating the correlation map is a popular algorithm for keypoint-based methods to localize the forged regions [14], [16]. Correlation-based algorithms usually require to first compute affine matrices, and then apply geometrical transformations on the whole image. However, these methods are very likely to cause false alarms when the image contains many similar patterns and periodical changes. For our method, in each iteration, we first construct suspicious regions by exploiting the scale information of keypoints, and then refine the map using the color information. This makes our method not only robust against the interference of similar patterns, but also very efficient. To validate this, we replace the localization strategy in Section V-D with the correlation-based algorithm ZNCC proposed in [16]. The resulting algorithm is then called 'matching+ZNCC'. Table IV summarizes the results obtained by 'matching+ZNCC' and our proposed algorithm. We can see that our method is much more efficient, and at the same time, achieves much better performance.

D. Detection Results Against Different Attacks

We now test our proposed technique against various attacks. Specifically, for the dataset FAU [6], we use the tool provided by Christlein *et al.* [6] to generate an image set containing 2016 images in total under different attacks, including scaling, rotation, JPEG compression and noise addition.

- Scaling: Each copied region is scaled between 90% and 110% of its original resolution by a step size 4%. In addition, some other large-scale resizing parameters are included: 50%, 80%, 120% and 200%. This results in a total of $10 \times 48 = 480$ images.
- Rotation: Each copied region is rotated between 10° and 90° by a step size 20°. In addition, some large rotation angles, i.e., 120°, 180°, 240° and 300°, are also considered. This results in a total of $9 \times 48 = 432$ images.
- JPEG compression: Each image is JPEG compressed using the quality factor (QF) ranging from 20 to 100 by a step size 10. This results in a total of $9 \times 96 = 864$ images.
- Noise addition: Each copied region is added with zero-mean Gaussian noise with std between 0.02 and 0.1 by a step size 0.02. This results in a total of $5 \times 48 = 240$ images.

The performance of many benchmark methods has been evaluated by Christlein *et al.* [6]. In our experiments, we choose several popular schemes, including keypoint-based approaches SIFT [3], [14], SURF [17], GoDeep [15],

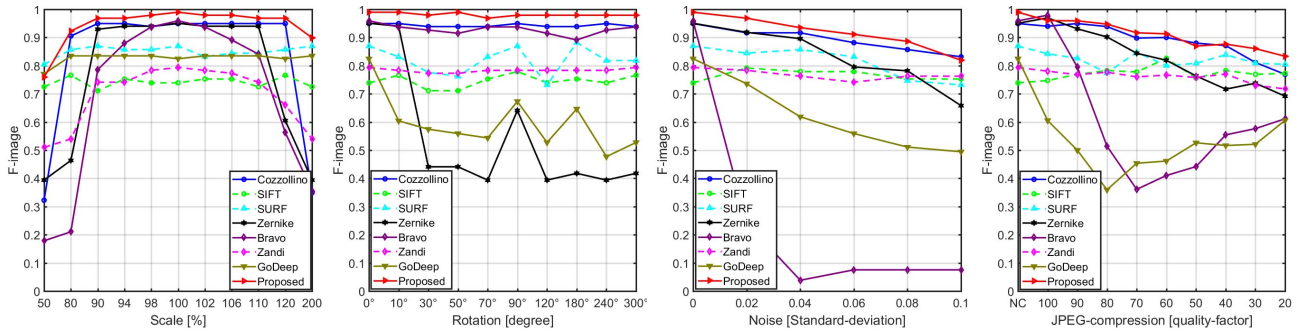


Fig. 10. Image level F_1 curves for different algorithms against (a) scaling, (b) rotation, (c) noise addition and (d) JPEG compression.

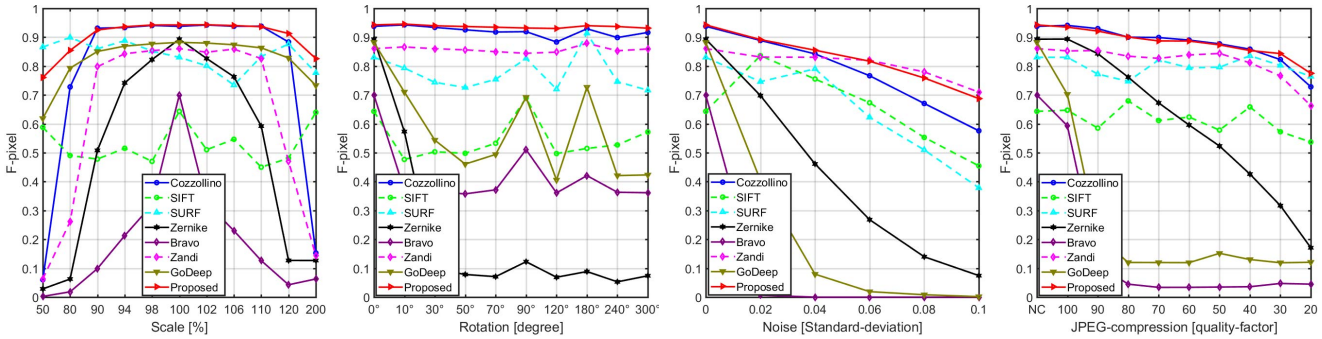


Fig. 11. Pixel level F_1 curves for different algorithms against (a) scaling, (b) rotation, (c) noise addition and (d) JPEG compression.

Zandi *et al.* [32], and block-based approaches Bravo-Solorio and Nandi [31], Zernike [6] and Cozzolino *et al.* [12]. Since the original SIFT-based algorithm Amerini *et al.* [3] does not handle the forgery localization problem, in this subsection, we use the code released by Christlein *et al.* [6]. The implementation of SIFT [3], [14] by Christlein *et al.* [6] adopts the matching and clustering algorithms proposed in [3], and employs the localization scheme devised in [14].

Fig. 10 and Fig. 11 respectively report the F_1-measure curves for the image level and the pixel level against different attacks. Since we use the same dataset as [6], a comparison with the other benchmark methods can be directly conducted using the results provided in [6]. As shown in Fig. 10 and Fig. 11, our proposed scheme achieves much better performance than the other keypoint-based approaches such as SIFT [3], [14] and SURF [17], at both the image level and the pixel level. We can also observe that all the keypoint-based approaches exhibit more stable performance than those block-based ones across all the conditions, including large-scale resizing (e.g., 50% and 200%), where all the considered block-based algorithms fail. Unfortunately, the overall performance of SIFT [3], [14] and SURF [17] is not satisfactory, due to the inability to discover the forgeries when copy-move is conducted in smooth or small regions. Our proposed method, fortunately can greatly overcome such limitation and achieves considerable performance gains over all the cases.

The block-based approaches Bravo-Solorio and Nandi [31], Zernike [6] and Cozzolino *et al.* [12] achieve similar performance under plain copy-move attack. However, all of them are vulnerable to one or several attacks. For example, Bravo-Solorio and Nandi [31] is very sensitive to Gaussian noise and compression noise as shown in Figs. 10(c),(d) and Figs. 11(c),(d). Zernike [6] is not robust against rotation attack as demonstrated in Fig. 10(b) and Fig. 11(b). We can also observe that all the block-based approaches perform poorly against large-scale resizing. In contrast, the proposed algorithm achieves consistently good performance across all the conditions.

E. Performance of Challenging Copy-Move Forgery Detection

In this Section, we show some challenging examples of the copy-move forgery detection. Figs. 12(a1)-(g1) depict the results when the copy-move forgery only involves smooth regions, where the variance of the copied region is about 9.0. As can be expected, *all* the previous keypoint-based algorithms [3], [5], [14] fail completely in this case, due to the lack of keypoints. Figs. 12(a2)-(g2), Figs. 12(a3)-(g3), Figs. 12(a4)-(g4) and Figs. 12(a5)-(g5) demonstrate the results against some severe attacks, i.e., large-scale resizing (200%), severe noise distortion (std is 0.1), rotation with a large angle (120°) and low rate JPEG compression (QF is 20), respectively. For Fig. 12(a4), the ratio of the smallest copied region over the entire image is only 0.15%, which is very small. As we can see, our proposed algorithm still achieves very accurate detection results under all these challenging conditions. The other algorithms, however, are not robust against one or several attacks. Meanwhile, it should also be emphasized that the proposed algorithm is of very low computational complexity as presented in Section VI-B.

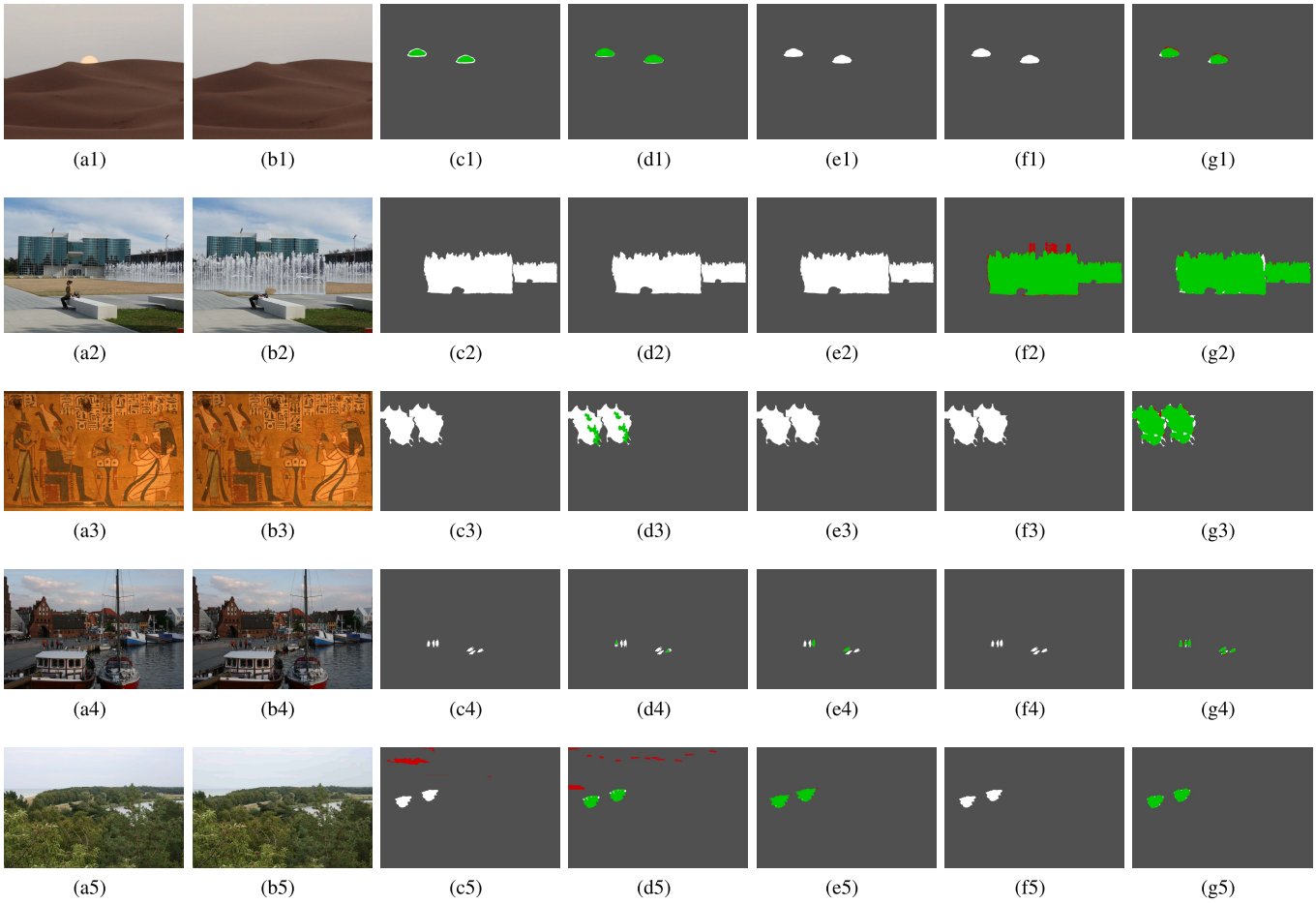
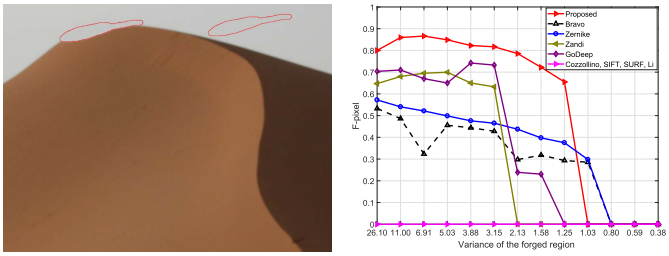


Fig. 12. Some challenging examples of copy-move forgery detection against different attacks. From top to bottom: plain, scaling 200%, rotation 120°, noise addition with normalized std 0.1, and JPEG compression with QF 20. From left to right: original images (a1)∼(a5), forged images (b1)∼(b5), masks output by Zernike [6] (c1)∼(c5), by Cozzolino *et al.* [12] (d1)∼(d5), by SIFT [3], [14] (e1)∼(e5), by Li *et al.* [5] (f1)∼(f5) and by our proposed algorithm (g1)∼(g5). White color indicates the ground-truth; the correct detection pixels are marked in green, while false alarms are in red.

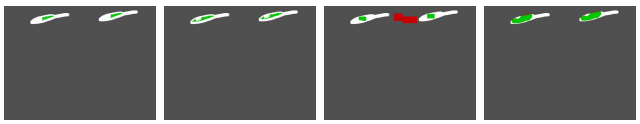
We now investigate the capability of our proposed method against the extremely smooth copy-move forgery. Fig. 13(a) gives an example where the copy-move forgery only involves smooth regions (the variance of the copied region is 26.10). In order to generate extremely smooth copy-move forgeries, we continuously smooth the copy-move regions using Gaussian filters with std varying from 2 to 7. Finally, we obtain a set of copy-move images with variances of the copied regions gradually changing from 26.10 to 0.38. Fig. 13(b) draws the pixel level F_1 curves for different algorithms over the generated images. Note that the image is successfully detected as a forgery when the F-pixel is bigger than 0. As can be seen, our method is able to detect the forgery with high localization accuracy even when the variance of the copied region is about 1, in which case the copy-move areas are extremely smooth (the average offset from the mean is only about 1). We can also observe that the algorithms SIFT [3], [14], SURF [17], Li *et al.* [5] and Cozzolino *et al.* [12] fail completely for all the variances. Though Bravo-Solorio and Nandi [31] and Zernike [6] are able to detect the forgery under even more extreme cases, the localization accuracy is much worse than ours. Fig. 14 shows an example of the masks obtained by Bravo-Solorio and Nandi [31], Zernike [6], GoDeep [15] and our proposed method.



(a) (b)

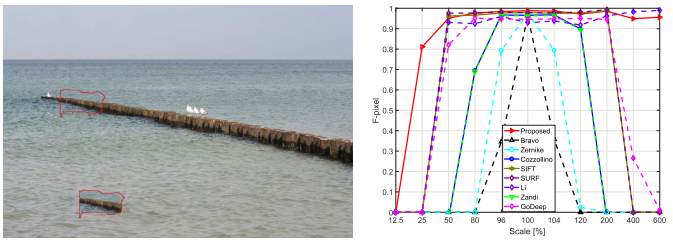
Fig. 13. An example of extremely smooth copy-move forgery detections. (a) The forgery image, where the copy-move regions are marked in red curves and (b) localization performance of extremely smooth copy-move forgery detections.

Fig. 15 gives an example to show the performance of different algorithms on the copy-move forgery detection with extreme scaling factors. For the copy-move image shown in Fig. 15(a), we generate a set of forgeries by resizing the copied region with factors from 12.5% to 600%, where the smallest ratio of the resized region over the whole image is only 0.017%. Fig. 15(b) reports the pixel level F_1 curves for different algorithms over the generated images. We can notice that all the keypoint-based algorithms exhibit high robustness against scaling attacks with extreme factors. Specifically, our



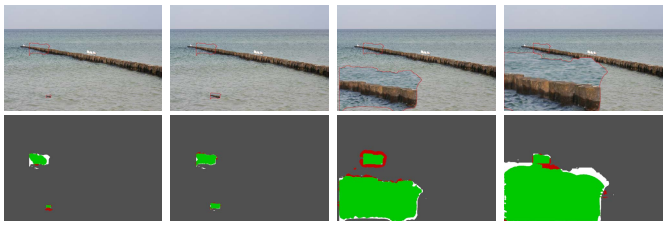
(a) (b) (c) (d)

Fig. 14. The localization maps obtained by different algorithms. The variance of the copied region is 2.13 (the average offset from the mean is about 1.46). (a) Bravo-Solorio and Nandi [31]. (b) Zernike [6]. (c) GoDeep [15]. (d) Proposed.



(a) (b)

Fig. 15. An example of copy-move forgery detections with extreme scaling factors. (a) The forgery image and (b) localization performance of copy-move forgery detections with respect to different scaling factors.



(a) (b) (c) (d)

Fig. 16. The localization maps obtained by our method when the copied region is resized by a factor of (a) 25%, (b) 50%, (c) 400% and (d) 600%.

method can successfully detect the forgeries over a rather wide scaling factor interval $[25\%, 600\%]$, where the size of the copied region varies from $\frac{1}{16}$ to 36 times of the original size. Note that the keypoint-based method Li [5] is also very effective against scaling attacks of extremely large factors, e.g., 600%; however, it takes over 5 hours to detect each generated image, while our algorithm only needs about 2 minutes. Cozzolino *et al.* [12] achieves the best performance among the dense-field approaches; but only works over a narrow scaling factor interval $[80\%, 120\%]$. Fig. 16 shows the localization masks obtained by our algorithm against scaling attacks of extreme factors.

VII. Conclusion

In this paper, we have proposed a fast and effective keypoint-based copy-move forgery detection and localization technique. By carefully studying the keypoint extraction algorithm (SIFT), we have first demonstrated that it is possible to generate a sufficient number of keypoints even in smooth or small regions, by lowering the contrast threshold and resizing the image. Then a novel hierarchical feature point matching strategy has been proposed to alleviate the critical matching problems. To reduce the false alarm rate and accurately localize the copied regions, we have further proposed a novel iterative localization scheme without involving any clustering and segmentation procedures. By fully exploiting the robustness properties of the SIFT algorithm (including the dominant orientation and the scale information) and the color information of each keypoint, our proposed technique achieves very high detection accuracy. Extensive experimental results have been provided to demonstrate the superior performance of our proposed scheme.

References

[1] A. J. Fridrich, B. D. Soukal, and A. J. Lukáš, "Detection of copy-move forgery in digital images," in *Proc. Digit. Forensic Res. Workshop*, 2003, pp. 1–10.

[2] G. Muhammada, M. Hussain, and G. Bebis, "Passive copy move image forgery detection using undecimated dyadic wavelet transform," *Digit. Invest.*, vol. 9, no. 1, pp. 49–57, 2012.

[3] 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 Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 1099–1110, Sep. 2011.

[4] Y. Li, J. Zhou, A. Cheng, X. Liu, and Y. Y. Tang, "SIFT keypoint removal and injection via convex relaxation," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 8, pp. 1722–1735, Aug. 2016.

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

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

[7] Y. Li, J. Zhou, and A. Cheng, "SIFT keypoint removal via directed graph construction for color images," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 2971–2985, Dec. 2017.

[8] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting duplicated image regions," Dartmouth College, Hanover, NH, USA, Tech. Rep. TR2004-515, 2004.

[9] J. Zhao and J. Guo, "Passive forensics for copy-move image forgery using a method based on DCT and SVD," *Forensic Sci. Int.*, vol. 233, nos. 1–3, pp. 158–166, 2013.

[10] S. Bayram, H. T. Sencar, and N. Memon, "An efficient and robust method for detecting copy-move forgery," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 1053–1056.

[11] S.-J. Ryu, M. Kirchner, M.-J. Lee, and H.-K. Lee, "Rotation invariant localization of duplicated image regions based on Zernike moments," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1355–1370, Aug. 2013.

[12] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy-move forgery detection," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 11, pp. 2284–2297, Nov. 2015.

[13] X. Bi and C.-M. Pun, "Fast reflective offset-guided searching method for copy-move forgery detection," *Inf. Sci.*, vols. 418–419, pp. 531–545, Dec. 2017.

[14] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 857–867, Dec. 2010.

[15] E. Silva, T. Carvalho, A. Ferreira, and A. Rocha, "Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting processes," *J. Vis. Commun. Image Represent.*, vol. 29, pp. 16–32, May 2015.

[16] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, L. Del Tongo, and G. Serra, "Copy-move forgery detection and localization by means of robust clustering with J-linkage," *Signal Process., Image Commun.*, vol. 28, no. 6, pp. 659–669, 2013.

[17] B. L. Shivakumar and S. S. Baboo, "Detection of region duplication forgery in digital images using SURF," *Int. J. Comput. Sci.*, vol. 8, no. 4, pp. 199–205, 2011.

[18] J. Zhao and W. Zhao, "Passive forensics for region duplication image forgery based on harris feature points and local binary patterns," *Math. Problems Eng.*, vol. 2013, Dec. 2013, Art. no. 619564.

[19] J.-M. Guo, Y.-F. Liu, and Z.-J. Wu, "Duplication forgery detection using improved DAISY descriptor," *Expert Syst. Appl.*, vol. 40, no. 2, pp. 707–714, 2013.

[20] P. Kakar and N. Sudha, "Exposing postprocessed copy–paste forgeries through transform-invariant features," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 1018–1028, Jun. 2012.

[21] Y. Li and J. Zhou, "Image copy-move forgery detection using hierarchical feature point matching," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, Dec. 2016, pp. 1–4.
[22] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.
[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[24] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[25] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009.
[26] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.
[27] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 497–506.
[28] D. Cozzolino, G. Poggi, and L. Verdoliva, "Copy-move forgery detection based on patchmatch," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2014, pp. 5312–5316.
[29] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *Proc. 18th ACM Int. Conf. Multimedia*, 2008, pp. 1469–1472.
[30] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge Univ. Press, 2003.
[31] S. Bravo-Solorio and A. K. Nandi, "Exposing duplicated regions affected by reflection, rotation and scaling," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2011, pp. 1880–1883.
[32] M. Zandi, A. Mahmoudi-Aznaveh, and A. Talebpour, "Iterative copy-move forgery detection based on a new interest point detector," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2499–2512, Nov. 2016.
[33] P. Vandewalle, J. Kovacević, and M. Vetterli, "Reproducible research in signal processing," *IEEE Signal Process. Mag.*, vol. 26, no. 3, pp. 37–47, May 2009.
[34] B. Wen, Y. Zhu, R. Subramanian, T. Ng, X. Shen, and S. Winkler, "COVERAGE—A novel database for copy-move forgery detection," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2016, pp. 161–165.

Yuanman Li (S'15) received the B.S. degree in software engineering from Chongqing University, Chongqing, China, in 2012, and the M.S. degree in software engineering from the University of Macau, Macau, China, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Computer and Information Science, Faculty of Science and Technology. His research interests include multimedia security, pattern recognition, and machine learning.

Jiantao Zhou (M'11) received the B.Eng. degree from the Department of Electronic Engineering, Dalian University of Technology, in 2002, the M.Phil. degree from the Department of Radio Engineering, Southeast University, in 2005, and the Ph.D. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, in 2009. He held various research positions with the University of Illinois at Urbana–Champaign, The Hong Kong University of Science and Technology, and McMaster University. He is currently an Associate Professor with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau. He holds four granted U.S. patents and two granted Chinese patents. His research interests include multimedia security and forensics, and multimedia signal processing. He was a co-author of two papers that received the Best Paper Award at the IEEE Pacific-Rim Conference on Multimedia in 2007 and the Best Student Paper Award at the IEEE International Conference on Multimedia and Expo in 2016, respectively.