




Copy-move forgery detection using image blobs and BRISK feature

Patrick Niyishaka¹  · Chakravarthy Bhagvati¹

Received: 8 August 2019 / Revised: 5 June 2020 / Accepted: 15 June 2020 /
Published online: 09 July 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

One of the most frequently used types of digital image forgery is copying one area in the image and pasting it into another area of the same image; this is known as the copy-move forgery. To overcome the limitations of the existing Block-based and Keypoint-based copy-move forgery detection methods, in this paper, we present an effective technique for copy-move forgery detection that utilizes the image blobs and keypoints. The proposed method is based on the image blobs and Binary Robust Invariant Scalable Keypoints (BRISK) feature. It involves the following stages: the regions of interest called image blobs and BRISK feature are found in the image being analyzed; BRISK keypoints that are located within the same blob are identified; finally, the matching process is performed between BRISK keypoints that are located in different blobs to find similar keypoints for copy-move regions. The proposed method is implemented and evaluated on the copy-move forgery standard datasets MICC-F8multi, MICC-F220, and CoMoFoD. The experimental results show that the proposed method is effective for geometric transformation, such as scaling and rotation, and shows robustness to post-processing operation, such as noise addition, blurring, and jpeg compression.

Keywords BRISK · Blob · CMF · CMFD · DoG · LoG

1 Introduction

Image tampering is defined as adding or deleting some important features from the image for malicious purposes [20, 23]. In *copy-move* forgery (CMF), a part of the image is copied and pasted into another part of the same image [23]. Detection methods for this type of forgery are called *Copy-Move Forgery Detection* (CMFD) and they are generally categorized into *Block-based* [5] and *Keypoint-based* approaches [3, 29].

In a *Block-based* approach, the image is divided into small overlapping or non-overlapping blocks. These blocks are almost always rectangular or square in shape. Features

✉ Patrick Niyishaka
niyishakapatrik@gmail.com

¹ University of Hyderabad, Hyderabad - India

are extracted from the blocks and compared against each other to find which blocks or features match. The Block-based techniques are effective for forgery under Gaussian noise and jpeg compression. The limitations of Block-based techniques include the difficulty of finding the appropriate size of the block. Small blocks increase the computational cost of matching and also do not give robust features. Large blocks cannot be used to detect small forged areas and tend to detect uniform areas as duplicates [27].

In *Keypoint-based* approach, feature vectors are computed for high-entropy regions without subdividing the image. Feature vectors are then analysed to identify similarities. Keypoints-based techniques are effective for detecting forgeries under scaling and rotation. Their main drawbacks include large number of keypoints to match and the need for filtering techniques such as Random Sample Consensus (RANSAC) for reducing the number of false positives [21, 30].

Recently, a third approach, based on image segmentation has been proposed [14]. The main limitation of this method is the inability of segmentation to separate foreground regions and background regions (see Fig. 3c).

We propose a novel alternative approach using image blobs [15, 20] and BRISK feature to overcome some of the limitations of Block-based, Keypoints-based, and Segments-based CMFD techniques. Our primary contribution is to show experimentally that image blobs improve the performance of several previously studied features, and in particular BRISK features in CMFD. We show that the proposed Blobs+BRISK approach reduces the number of keypoints to match by almost 50%, and also reduces false positives without requiring a filter algorithm. We show in Section 3 that combining edge detection with blob detection also separates the foreground and background regions thus eliminating false positives occurring due to similarities in background areas (see Fig. 3b).

The rest of this paper is structured as follows. In Section 2, the related work along with state-of-art CMFD techniques are discussed. In Section 3, our proposed method is presented and we examine the advantages of image blobs over image blocks and image segments for CMFD. Section 4 describes our experimental results, analyzes them and discusses the role of blobs in the improved performance.

2 Related work

CMFD techniques in literature may be put in two groups. Block-based and Keypoints-based as stated earlier as well as in [3, 29]. In block-based methods, the input image is first divided into overlapping or non-overlapping blocks which are then compared to each other based on their extracted features. Finally, similar blocks are considered as forgery parts. Usually, all block-based methods have a similar framework, they typically differ in feature extraction step. Malviya and Ladhake [19] used Auto Color Correlogram (ACC) to extract feature vectors from each block, then L1 norm is used for comparing ACCs of two blocks to conclude if there has been a forgery. Aleksandra et al. [22] proposed a CMFD method based on multifractals. An input image is first divided into non-overlapping blocks of sizes 8×8 , 16×16 , 32×32 , and the multifractal spectrum is calculated for each block. Finally, images blocks are matched and the Basic Variable Neighborhood Search (BVNS) is applied to extract forged areas. Faten et al. [6] proposed two techniques, first method is based on trigonometric transforms, and second method is based on deep learning Convolution Neural Network(CNN). For a CMFD using trigonometric transforms, they divided the image into overlapping blocks and applied the trigonometric transforms on each block to extract

features. Then they determined the difference in features between the authentic and tampered images. For the deep learning method, CNN layers are used for feature extraction. After these layers, a Global Average Pooling (GAP) layer with a fully-connected layer are included. Finally, a dense layer deciding between two classes (authentic or forged) is used for the classification phase. Ojeniyi et al. [21] hybridized a block-based DCT (Discrete Cosine Transform) technique and a keypoint-based SURF (Speeded-Up Robust Feature) technique in a single CMFD technique. The main drawback of this method is the use of the filtering algorithm for the reduction of the false positive. The major drawbacks of block-based methods are less robustness against rotation and scaling attacks and computational complexity.

Keypoint-based CMFD methods try to tackle these drawbacks by focusing on points of interest in the input image. Keypoints are computed for high-entropy regions without subdividing the image, keypoints descriptors are then analyzed to identify similarities. Zhu et al. [30] proposed a method that extracts Oriented FAST and Rotated BRIEF (ORB) keypoints at different scales of a Gaussian pyramid. Then ORB keypoints are matched to find similar keypoints. This method is effective for geometric transformation but it is time-consuming for high-resolution images and uses RANSAC filter algorithm to remove false matches. Mahdi and Mahdi [17] used SIFT algorithm to extract key-points features and their descriptors. Then, the low-frequency components are used to compute a dynamic threshold rather than a fixed threshold. Finally, k-nearest neighbors of each keypoint are found, and matched keypoints are found. The main drawback of this method includes the need for a filter method to remove false positive areas. Jun et al. [9] presented a CMFD method based on SIFT and a reduced Local Binary Pattern (LBP) histogram. Keypoints are extracted in the image using SIFT. For each keypoint descriptor, reduced LBP features are generated and used to detect the CMF. This method uses the RANSAC filter algorithm for false match removal. A CMFD technique based on the superpixel segmentation algorithm and Helmert transformation was proposed by [8]. They used SIFT to extract keypoints in image, then matching pairs are obtained by calculating the similarity between keypoint descriptors. Finally, they grouped these matching pairs based on spatial distance and geometric constraints via Helmert transformation to obtain the coarse forgery regions. Behnaz et al. [2] used the over-segmentation as a preprocessing step to obtain superpixels which are then regarded as nodes of the Markov network. Simple Linear Iterative Clustering (SLIC) method is used to segment images in several superpixels, then PCT (Polar Cosine Transform) and SURF features are used for feature matching and detecting similar parts of the image. The major drawbacks of keypoint-based methods are large number of keypoints to match and the use of filter algorithm to remove false matches.

The above CMFD methods exhibit the main limitations of block-based, keypoints-based, and segments-based discussed earlier in Section 1. Therefore, we propose a CMFD method based on image blobs and BRISK feature (Blobs + BRISK). Our method reduces the number of keypoints to match by almost 50%, and also removes false positives without requiring a filter algorithm.

3 Proposed method

In this section, the details of the proposed CMFD method are discussed. In [28] authors have found that SIFT, SURF and BRISK features give a superior performance to others such as ORB. However, BRISK carries a significantly lower computational cost than SIFT and SURF. Hence, we based our approach on BRISK.

There are five main steps in our algorithm: pre-processing, detecting blobs, extracting BRISK features, identifying and listing BRISK keypoints located within a blob, and matching BRISK features between different blobs. Figure 1 shows the pipeline of the proposed method.

3.1 Step-1: pre-processing

In this step, large input images are scaled down to a maximum size of 1000×1000 pixels, and edge detection is performed. Edge detectors such as Sobel [26] find the boundaries of objects within image and work by detecting discontinuities in brightness. Edge detection results in a 2D map of the gradient at each point: the areas with high gradient visible as pixels of high intensity.

We used an edge detector prior to blob detection for two main reasons:

1. To enhance blob localization on foreground objects. The regions with high gradient are bright (foreground objects) and the regions with low gradient such as background are dark (see Fig. 3b). Our assumption is that backgrounds are generally lacking in detail while the foreground carries information (see Section 4)
2. If there is a copy-move attack, the original region and its duplicate are located in different blobs. Edge detection causes objects in the image to become distinct regions with edges separating them. As a result, these different regions are detected as distinct blobs.

3.2 Step-2: blob detection

Image blobs are image regions that differ in properties, such as brightness or color compared to surrounding regions and the goal of blob detection is to identify and mark these regions. Most common blob detectors include Laplacian of Gaussian (*LoG*) [11, 15] and the Difference of Gaussian (*DoG*) [15, 16] operators.

LoG blob filter is the second derivative of Gaussian filter. An image is convolved with the blob filter at multiple scales and blobs are detected from the extrema of filter responses in the resulting scale space. The radius of each blob is approximately $\sqrt{2}\sigma$. The second order derivative is extremely sensitive to noise, but the Gaussian blur smooths out noise and stabilizes the second order derivative making the LoG filter extremely popular in image processing. The 2D LoG filter is given by:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-(x^2+y^2)/2\sigma^2} \quad (1)$$

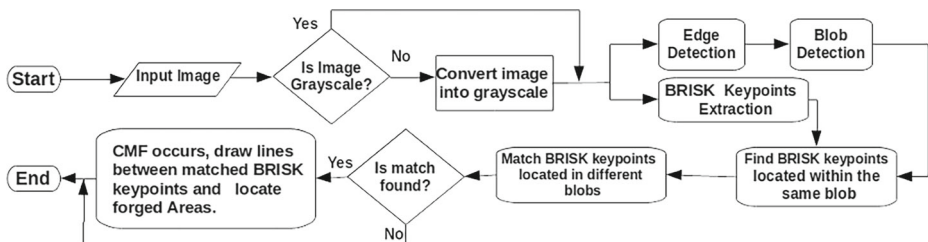


Fig. 1 Overview of the proposed technique

where σ is the standard deviation of the Gaussian. The magnitude of the Laplacian response is maximum at the center of the blob when the scale of the Laplacian matches the scale of the blob. To make the response independent of scale, *scale normalization* is performed by multiplying *LoG* with σ^2 . The main drawback of *LoG* is that computing the second order derivatives is computationally intensive. So *LoG* is approximated by a Difference of two Gaussians (*DoG*) [16] at different scales .

$$DoG = g(x, y, \sigma k) - g(x, y, \sigma) \quad (2)$$

$g(x, y, \sigma)$ is a Gaussian filter with σ as the standard deviation, and k is a scale variable. *Blobs* are scale-space extrema of differences of Gaussians [15, 16]. Figure 2 shows blob detection using *DoG* on an image from MICC-F8multi [1] dataset.

3.2.1 Advantages of image blobs over image blocks and image segments in CMFD

If we split an image $I(x, y)$ of size $M \times N$ in overlapping fixed-size blocks β with n as slide step size, the number of blocks N_b is given by [6, 18]:

$$N_b = (M - \beta + n)(N - \beta + n) \quad (3)$$

N_b increases as image size increases. This is computationally expensive for small β (needed for good localization and detection of small forged areas). Larger values of β tend to miss small forged areas while also increasing false positives although the computational cost decreases.

Figure 3(a) shows image blocks in contrast to the image blobs in Fig. 3(b). We can observe that if we apply a blob detector, blobs are the bright regions on the dark background, the uniform areas are considered not as duplicates but as the background. The image scaling does not affect blobs since blobs are detected in scale-space and the radius of each blob is approximately $\sqrt{2}\sigma$ [15]. Small forged areas are detected in small blobs and large forged areas are detected in large blobs. Figure 3(c) illustrates image segmentation approach from [8]. It may be seen that image blobs separate the foreground objects (cars) from the background. Image segmentation also separates the cars into different regions but it also has several segments corresponding to the background which need to be processed. The additional processing makes segmentation computationally more expensive than our blob-based approach. The elimination of background from processing also reduces false positives occurring because of similarities in background areas.

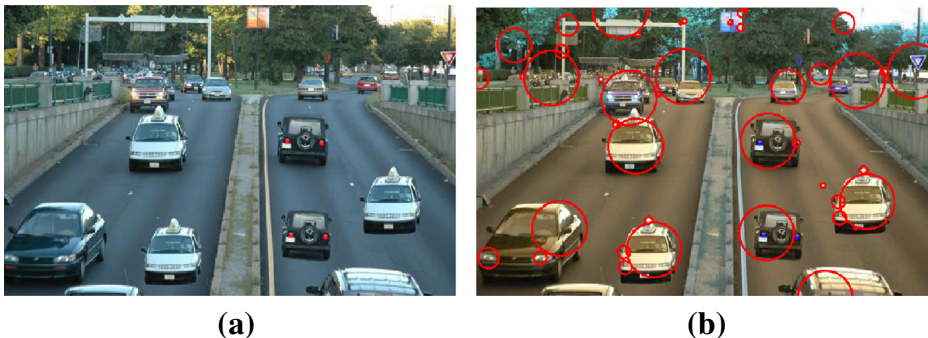


Fig. 2 **a** Sample image. **b** Red circles are blobs detected using DoG

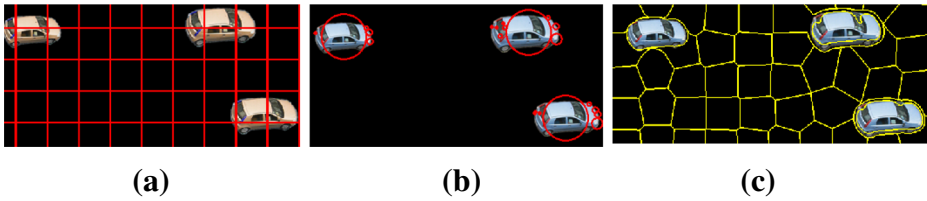


Fig. 3 **a** Image blocks. **b** Image blobs (DoG). **c** Image segments

On images of size 532×800 pixels from MICC-F220 dataset, the method in [6] uses $(532 - 8 + 1) \times (800 - 8 + 1) = 416325$ blocks/per image, whereas our method uses 240 blobs per image.

3.3 Step-3: BRISK Feature Extraction

This step computes Binary Robust Invariant Scalable Keypoints (BRISK) and the BRISK descriptor.

BRISK is a technique for scale-space Keypoint detection and binary description [12]. Keypoints are detected in octave layers of the image pyramid. The location and the scale of each keypoint is converted into a continuous domain representation via quadratic function fitting. Once the BRISK features are identified, BRISK descriptor is computed as a binary string in two stages. The first stage estimates orientation of the keypoints and helps in creating a rotation-invariant descriptor. The second stage involves robust brightness comparisons to result in a descriptor that effectively and efficiently captures local region properties. BRISK descriptor is briefly described below for ready reference in this paper but the reader is referred to the original paper [12] for details.

BRISK descriptor makes use of a sampling pattern in the form of concentric circles that define N locations. The intensity of each point \mathbf{p}_i in the pattern is smoothed with a Gaussian to prevent aliasing effects. The N sample points are gathered into pairs $(\mathbf{p}_i, \mathbf{p}_j)$ and divided into two classes: short pairs if the distance between $(\mathbf{p}_i, \mathbf{p}_j) < T_{max}$; and long pairs if the distance is greater than T_{min} . The long pairs are used to estimate rotation while the short-pairs are used to build the descriptor after rotation correction. BRISK descriptor local gradients are computed by:

$$\nabla(\mathbf{p}_i, \mathbf{p}_j) = (\mathbf{p}_j - \mathbf{p}_i) \frac{I(\mathbf{p}_j, \sigma_j) - I(\mathbf{p}_i, \sigma_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|^2} \quad (4)$$

$\nabla(\mathbf{p}_i, \mathbf{p}_j)$ is the local gradient between the sampling pair $(\mathbf{p}_i, \mathbf{p}_j)$ and $I(x, \sigma)$ is the smoothed intensity at x at scale σ . Rotation angle θ is estimated from the average gradient in x and y directions. The short pairs are rotated by angle $-\theta$ to provide the descriptor invariant to rotation. For building the descriptor, BRISK takes the set of short pairs (\mathbf{p}_j, σ_j) , rotates the pairs by the orientation $-\theta$ to get (p_j^θ, σ_j) and makes comparisons of smoothed intensity (I) such that each bit \mathbf{b} corresponds to:

$$\mathbf{b} = \begin{cases} 1 : I(p_j^\theta, \sigma_j) > I(p_i^\theta, \sigma_i) \\ 0 : \text{Otherwise} \end{cases} \quad (5)$$

Every keypoint will be described by an encoded binary string and this is called the binary descriptor.

3.4 Step-4: Determine the BRISK keypoints located within the same blob

To find out which BRISK keypoints are located inside the same blob as shown in Fig. 4(c), we extract 2D spatial coordinates for each blob (x_b, y_b, σ) , and for each BRISK keypoint (x_k, y_k) . The center of the blob is (x_b, y_b) and σ is the standard deviation of the Gaussian kernel which detected the blob. The radius \mathbf{r} of each blob is approximately $\sqrt{2}\sigma$ [15]. The distance \mathbf{D} from the center of the blob to the keypoint is given by :

$$\mathbf{D} = \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} \quad (6)$$

A keypoint is located inside the blob if $\mathbf{D} \leq \mathbf{r}$, and outside the blob if $\mathbf{D} > \mathbf{r}$. This is shown in Fig. 4(c) where the keypoints detected within the same blob (red circles in Fig. 4(b),) are shown in green.

3.5 Step-5: Feature matching.

Feature matching is the process for finding similar Keypoints from different blobs. Every BRISK keypoint is encoded as a binary descriptor $\beta D(i)$ of size Z . Similarity is given by the Hamming distance (H_d) between the two keypoint descriptors $\beta D(i)$ and $\beta D(j)$, i.e., the number of bits which differ between the two binary descriptors [30].

$$H_d = \sum_{k=1}^Z \text{XOR}(\beta D_k(i), \beta D_k(j)) \quad (7)$$

where XOR is the standard exclusive-OR operator. We use the 2-Nearest Neighbor matches and ratio criterion [16] to find the correct keypoint matches.

Two nearest neighbours, for each keypoint in the blob, are found from the Hamming distances. Let the distances of the two such neighbours be d_1 and d_2 . Given a predefined threshold $T \in (0, 1)$, a match is confirmed if

$$d_1/d_2 < T \quad (8)$$

T is determined empirically and is described in Section 4.2 on experimental results. In [1, 21] a minimum of three matching pairs between different clusters is required to consider a forgery, while for our method, a minimum of two matching pairs is sufficient ($\text{min_match} \geq 2$). This is found empirically by experiments on the various datasets.



Fig. 4 **a** Copy-move forged image. **b** Green circles are detected BRISK keypoints **c** Red circles are detected blobs and green circles are BRISK keypoints inside different blobs **d** Green lines indicate BRISK keypoints matching pairs from different blobs

3.5.1 Reducing the number of keypoints to match.

The input to the blob detector is an edge detected image. This increases the likelihood that if there is a copy-move forgery the authentic region and its duplicate are located in different blobs allowing us to reduce the number of keypoint pairs in matching. BRISK keypoints located in the same blob are not matched for CMFD. Table 1 reports the reduction of the number of BRISK keypoints to match on 8 images from the dataset MICC-F8multi [1] forged with multi copy-move regions. When image blobs are used the number of matches μt is given by:

$$\mu t = \frac{n!}{(n-k)!k!} \quad (9)$$

n is the number of BRISK keypoints for each image and $k=2$. When blobs are not used Kp_o is the number of keypoints and μt_o is the number of matches. When blobs are used Kp_b is the number of keypoints and μt_b is the number of matches. $dec(\%)$ shows the number of matches decreased for each image in % on 8 images from the dataset MICC-F8multi. DoG $\max\sigma=40$ and BRISK samples radius = 27.

Recent techniques [8, 9] match all of the keypoints detected, whereas our method matches half of all keypoints detected.

3.5.2 Tackling the use of filtering techniques

The main source of the false positives in many earlier methods is matches between spatially close areas and similar intensities between neighboring pixels. These false positives have been reduced by *Filtering* techniques based on robust statistics, especially RANSAC [21, 30]. The use of filtering is an additional step and adds to the computational costs.

When using image blobs, there is no need to use any filtering algorithms since the neighboring pixels with similar intensities are detected within the same blob and the spatially close areas that look similar at different scales are also detected within the same blob. There is no need to match the keypoints located in the same blob for CMFD since the original area and its duplicate are located in different blobs. Blobs also separate foreground and background areas which further minimizes or eliminates false positives occurring due to similar background areas.

Table 1 Reduction of the number of BRISK keypoints to match

Images	#Blobs	#Kp _o	#Kp _b	μt_b	μt_o	dec(%)
1	281	6438	4256	9054640	20720703	56.3
2	65	1754	1498	1121253	1537381	27
3	115	7002	3551	6303025	24510501	74.28
4	106	851	720	258840	361675	28.43
5	239	5980	4299	9238551	17877210	48.32
6	162	6664	4304	9260056	22261116	58.4
7	408	4435	3377	5700376	9882395	42.31
8	325	8297	5566	15487398	34415956	54.99
-	-	-	-	-	-	49%

The main result are shown in bold

4 Experiments and results

In this section, we describe the datasets used in our experiments, the evaluation metrics for performance analysis and running times. In Sec. 4.3 our method is evaluated against geometric transformations and post-processing operations. Finally, comparative results are discussed in Sec. 4.4.

4.1 Datasets description and evaluation metrics

During the experiments, we have used the datasets MICC-F8multi [1] for multi copy-move regions. We used MICC-F220 [1] dataset for geometric transformation operations where the rotation θ is in degrees and (S_x, S_y) are the scaling factors applied to the cloned patch. We used images from CoMoFoD dataset [4] for post-processing operations. Table 2 shows the parameters of MICC-F220 and CoMoFoD datasets.

The following popular metrics [3] are used to assess the performance of a CMFD methods: forged images (F_i) that are correctly detected are t_p (True positives), images wrongly detected as forged are f_p (False positives), tampered images that are undetected and marked as not forged f_n (False negatives), and untampered images(O_i) correctly detected are t_n (True negatives). TPR is true positive rate and FPR is false positive rate. Precision (p_r) denotes the probability that a detected forgery is truly a forgery, while Recall (r_c) shows the probability that a forged image is detected as being forged. f_1 score is a measure which combines p_r and r_c in a single value. Finally, accuracy acc is the combined fraction of true positives and negatives in the entire test set.

$$TPR = \frac{\# t_p}{\# F_i}, \quad FPR = \frac{\# f_p}{\# O_i}, \quad acc = \frac{tp + tn}{tp + tn + fp + fn} \tag{10}$$

$$p_r = \frac{tp}{tp + fp}, \quad r_c = \frac{tp}{tp + fn}, \quad f_1 = 2 \frac{p_r r_c}{p_r + r_c} \tag{11}$$

4.2 Setting the threshold T , experimental platform, and analysis of running time

1. Setting the threshold T for Nearest Neighbor Matching Ratio:
Our method requires setting the threshold (T in Eq. 8). It is empirically determined from tests on 220 images (110 original and 110 forged) from MICCF-220 dataset. It is shown from the results in Table 3, that as the T value begins to increase from a value

Table 2 Parameters of MICC-F220 and CoMoFoD datasets

MICC-F220				Parameters for post-processing operations. CoMoFoD dataset	
Attack	θ	S_x	S_y	Methods	Parameters
A	0	1	1	Jpeg compression	factor= [20, 30, 40, 50, 60, 70, 80, 90, 100]
B	10	1	1	Noise adding	$\mu = 0, \sigma \ 2 = [0.009, 0.005, 0.0005]$
C	20	1	1	Image blurring	averaging filter = [3x3, 5x5, 7x7]
D	30	1	1	Brightness change	(lower bound, upper bound) = [(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]
E	40	1	1	Contrast adjustment	(lower bound, upper bound) = [(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]
F	0	1.2	1.2		
G	0	1.3	1.3		
H	0	1.4	1.2		
I	10	1.2	1.2		
J	20	1.4	1.2		

of 0.2 to 0.43, the f_1 score also increases. As T value continues to increase to a value of 0.44 to 0.6, the f_1 score begins to decrease. This shows that the $T = 0.43$ has the best performance results and it is set as the operating point for the proposed CMFD. It is also seen that the performance varies by only about 2% for $0.4 \leq T \leq 0.6$, i.e., the choice of T is not extremely critical to performance. We repeat that $min_match \geq 2$.

Using the procedure described in Table. 3 we also implemented image blobs and SIFT feature [28] (Blobs+SIFT) which gave an f_1 score = 92.43% when $T = 0.4$ and $min_match \geq 3$, whereas image blobs and SURF feature [28] (Blobs+SURF) gave f_1 score = 86.36% when $T = 0.4$ and $min_match \geq 2$.

2. Experimental platform and analysis of running time:

Implementation is done with Python 3.6.7 and OpenCV 3.4.2 running on Ubuntu 18.04.2 LTS. All the timings are for an Intel i5 processor with 8GB RAM. Sobel edge detector with DoG is used in blob detection. DoG parameters are described in [16], we have set the minimum $\sigma = 1$ to detect smaller blobs and the maximum $\sigma = 40$ to detect larger blobs. Table 4 shows the number of blobs, keypoints, and the average running time in seconds recorded on the image car (See Fig. 2) of size 800X532 from the MICC-F8multi dataset.

4.3 Geometric transformation operation and post-processing operations

1. CMFD results for multiple copy-move regions in same image:

Table 5 reports CMFD results under multi copy-move regions in the same image on 45 images (15 original and 30 forged) from CoMoFoD and MICCF8-multi datasets, t_p is considered only if all forged areas within image are detected.

2. CMFD results for rotation and scaling operations:

We used images from dataset MICC-F220 with parameters reported in Table 2 to evaluate the proposed method on forgeries that include *rotation* on 55 images(11 authentic and 44 forged) with angles={10°, 20°, 30°, 40°}, scaling on 44 images(11 authentic and 33 forged), and rotation +scaling on 33 images (11 authentic and 22 forged). CMFD results are reported in Table 5, and Fig. 5 shows CMFD under rotation transformations.

3. CMFD results for post-processing operations:

To evaluate our method on post-processed copy-move forgeries, we used 70 images (20 originals, 20 forged with jpeg compression, and 30 forged with Blur, Noise addition

Table 3 Experimental results to determine the threshold value T

Treshhold	MICCF-220					
	t_p	f_p	f_n	$p_r(\%)$	$r_c(\%)$	$f_1(\%)$
0.2	25	0	85	100.00	22.72	37.02
0.3	86	2	24	97.72	78.18	86.86
0.4	98	6	12	94.23	89.09	91.58
0.43	103	6	7	94.49	93.63	94.05
0.44	103	7	7	93.63	93.63	93.63
0.5	109	14	1	88.61	99.09	93.55
0.6	110	20	0	84.61	100.00	91.66

The main result are shown in bold

Table 4 Comparison of running times

Methods	# Blobs	# Keypoints	Running time(seconds)
DoG+ORB [20]	239	4960	4.65
Blobs+SIFT	239	3027	4.10
Blobs+SURF	239	3467	4.18
Proposed(Blobs+Brisk)	239	5980	6.24

The main result are shown in bold

& contrast adjustment) from CoMoFod dataset with parameters reported in Table 2. CMFD results are reported in Table 5.

4.4 Comparative results

Table 6 reports the detection results in terms of TPR and FPR on MICC-F220 dataset. We compare the performance of our method with other methods known in the literature [21] on 220 images (110 authentic and 110 forged) from MICC-F220, and the results are reported in Table 7. The comparative results on 400 images (100 originals and 300 forged) from CoMoFoD dataset as in [19, 20] are reported in Table 8. The CMFD results obtained on MICC-F220 are $t_p : 103, t_n : 104, f_p : 6, f_n : 7$. The CMFD results obtained on CoMoFoD are $t_p : 276, t_n : 91, f_p : 9, f_n : 24$. From the comparative results in Table 7, our method exhibits a comparable matching performance to the best-known algorithms [10, 13, 21] that depend on refinement stage

Table 5 CMFD results on forged images with rotation, scaling, rotation+scaling, multi copy-move regions, and post-processing operations

Methods	t_p	f_p	f_n	tp_r	fp_r	t_p	f_p	f_n	tp_r	fp_r
	Rotation					scaling				
DoG+ORB [20]	34	2	10	0.77	0.18	24	2	9	0.72	0
Blobs+SIFT	40	0	4	0.9	0	32	0	1	0.96	0
Blobs+SURF	21	1	23	0.47	0.09	26	1	7	0.78	0.09
Proposed(Blobs+BRISK)	42	1	2	0.95	0.09	30	1	3	0.90	0.09
	Rotation + scaling					Multiple copy-move regions				
DoG+ORB [20]	17	2	5	0.77	0.18	24	1	6	0.8	0.06
Blobs+SIFT	20	0	2	0.90	0	24	1	6	0.8	0.06
Blobs+SURF	10	1	12	0.45	0.09	23	2	7	0.76	0.13
Proposed(Blobs+Brisk)	19	1	3	0.86	0.09	25	1	5	0.83	0.06
	Jpg compression					Blur, Noise add & contrast adj.				
DoG+ORB [20]	18	2	2	0.9	0.1	29	2	1	0.96	0.1
Blobs+SIFT	18	2	2	0.9	0.1	29	2	1	0.96	0.1
Blobs+SURF	17	3	3	0.85	0.15	28	2	2	0.93	0.1
Proposed(Blobs+BRISK)	19	2	1	0.95	0.1	29	2	1	0.96	0.1

The main result are shown in bold

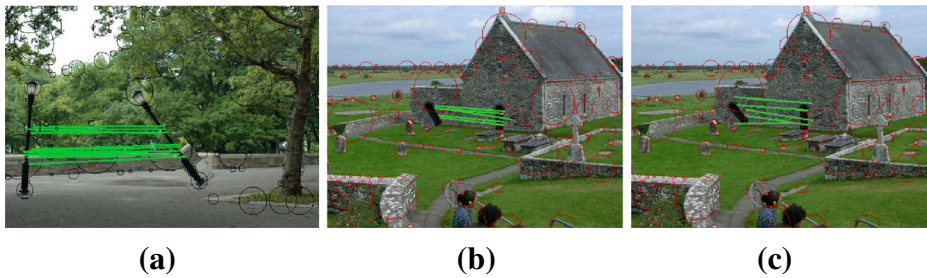


Fig. 5 **a** Lamp rotation angle = 40° . **b** Door rotation angle = 30° . **c** Door rotation angle = 40° . Two door areas indicate copy-move forgery, circles are detected blobs, and green lines are the BRISK keypoints matching pairs from different blobs

and filter algorithms to remove false matches. However, our method offers the advantage of reducing keypoints to match by around 50% and avoids the use of filter algorithms. In Table 8 our method gives better f_1 score, we highlight that images used are of size 512x512 and more than fifty percent are in PNG format. Thus, our algorithm is not dependent on a single image format such as JPEG.

Results in Table 5 show that our method is capable of CMFD when geometric transformations are done to the cloned regions as well as with postprocessing operations. Equally important is that blobs can be combined with a variety of image processing features proposed in literature for CMFD such as SIFT and SURF, not only BRISK. Thus, it may be safe to suggest that blobs provide an alternative to blocks with many advantages that include scale-space analysis to deal with varying sizes of forged regions, computational efficiency, separation of original and forged areas into different blobs, minimisation of false positives caused by background, elimination of filtering step, etc.

Finally, there are two potential weaknesses in the use of blobs. If the original and forged areas overlap in the copy-move forgery, then their corresponding blobs overlap too. However, such cases are very few in the datasets used in our experiments (see Fig. 6). In such images, if the forged area is large enough to contain more than one blob, then they are detected as forgeries because not all blobs overlap. The second is that if the background is detailed, it can have blobs too which result in false positives. Again, we found that in the datasets used there was not a single failure of our method due to this effect. The datasets did contain images with grass, rocks, trees, etc. in the background but the method appears empirically robust.

Table 6 Detection results in terms of TPR and FPR on MICC-F220 dataset

Methods	<i>TPR</i> (%)	<i>FPR</i> (%)
DoG+ORB [20]	90	9
Blobs+SIFT	90	5.4
Blobs+SURF	86	13
SLIC+PCT+SURF [2]	62	40
SIFT+Clusters [1]	100	8
SIFT+LBP [9]	99.1	5.4
Proposed(Blobs+BRISK)	93	5.4

The main result are shown in bold

Table 7 Comparative results between our method and other methods known in the literature [21] on 220 images from MICC-F220

Technique used	p_r (%)	r_c (%)	acc (%)
DyWT+SIFT [7]	88.89	80.00	85.00
PCA+SIFT [10]	93.04	97.27	95.00
DWT+SURF [25]	77.17	64.55	72.60
DyWT+SURF [24]	77.06	76.36	76.71
HDS [21]	93.86	97.27	95.45
DoG+ORB [20]	90.09	82.72	86.24
SIFT+LBP [9]	-	-	96.82
Deep Learning [6]	-	-	100
Blobs+SIFT	90.43	94.54	92.27
Blobs+SURF	86.36	86.36	86.36
Proposed Blobs+BRISK	94.49	93.63	94.09

The main result are shown in bold

Table 8 Comparative results between our method and other methods known in the literature [19, 20] on 400 images from CoMoFoD dataset

Technique used	p_r (%)	r_c (%)	f_1 (%)
ACC [19]	95.65	91.67	93.62
DoG+ORB [20]	96.47	91.33	93.82
Blobs+SIFT	98.18	90.00	93.91
Blobs+SURF	96.42	91.00	93.09
Proposed Blobs+BRISK	96.84	92.00	94.35

The main result are shown in bold

**Fig. 6** Left image shows overlapping cloned and original areas. Right image shows that there is an overlapping blob (red) and a correctly detected non-overlapping blob (arrow)

5 Conclusion

In this paper, a novel method for copy-move forgery detection based on image blobs and BRISK feature is presented. Image blobs are regions that are different from neighbors at different scales and they present various advantages over image blocks and image segments in CMFD. Edge detection is performed before blob detection to enhance blob localization on foreground objects, and to ensure that the authentic region and its duplicate are located in different blobs. To find copy-move regions we match BRISK keypoints from different blobs. Since the BRISK keypoints located within the same blob are not matched, the number of keypoints to match is dramatically reduced, and the need of the filter algorithm to remove false matches is eliminated. The experimental results show that the proposed technique is effective for geometric transformations and shows robustness to post-processing operations.

References

1. Amerini I, Ballan L, Caldelli R, Del Bimbo A, Serra G (2011) A sift-based forensic method for copy-move attack detection and transformation recovery. *Information Forensics and Security. IEEE Transactions on* 6:1099–1110. <https://doi.org/10.1109/TIFS.2011.2129512>
2. Behnaz E, Ahad H, Amirhossein T (2019) A probabilistic framework for copy-move forgery detection based on markov random field. *Multimedia Tools and Applications*
3. Christlein V, Riess C, Jordan J, Riess C, Angelopoulou E (2012) An evaluation of popular copy-move forgery detection approaches. *arXiv:1208.3665*
4. Dijana T, Ivan Z, Sonja G, Mislav G (2013) Comofod - new database for copy-move forgery detection
5. Emre G, Guzin u, Mustafa U (2019), Detection of free-form copy-move forgery on digital images. *Security and Communication Networks*
6. Faten MAA, Ahmed S, Moawad ID, Ghada AAMKM, El B, Ahmed SE, Fathi EAES (2020), An efficient method for image forgery detection based on trigonometric transforms and deep learning. *Multimedia Tools and Applications*
7. Hashmi MF, Anand VG, Keskar A (2014). In: Copy-move image forgery detection using an efficient and robust method combining un-decimated wavelet transform and scale invariant feature transform AASRI *Procedia* 9:84–91, <https://doi.org/10.1016/j.aasri.2014.09.015>
8. Hui-Yu H, Ai-Jhen C (2019) Copy-move forgery detection for image forensics using the superpixel segmentation and the helmert transformation
9. Jun YP, Tae AK, Yong HM, Eom IK (2020) Copy-move forgery detection using scale invariant feature and reduced local binary pattern histogram. *Symmetry*
10. Kaur H, Saxena J, Singh S (2015) Simulative comparison of copy-move forgery detection methods for digital images. *International Journal of Electronics Electrical and Computational System IJECS* 4:62–66. ISSN 2348-117X, Special Issue September 2015
11. Kong H, Cinar-Akakin H, Sarma S (2013) A generalized laplacian of gaussian filter for blob detection and its applications *Cybernetics*. In: *IEEE Transactions on* 43:1719–1733, <https://doi.org/10.1109/TSMCB.2012.2228639>
12. Leutenegger S, Chli M, Siegwart RY (2011) Brisk: Binary robust invariant scalable keypoints. In: *Proceedings of the 2011 International Conference on Computer Vision, IEEE Computer Society, Washington, DC, USA, ICCV '11*, pp 2548–2555 <https://doi.org/10.1109/ICCV.2011.6126542>
13. Lin C, Lu W, Huang X, Liu KZ, Sun W, Lin H, Tan Z (2018) Copy-move forgery detection using combined features and transitive matching. *Multimedia Tools and Applications* pp 1–16
14. Lin C, Lu W, Sun W, Zeng J, Xu T, Lai JH (2018) Region duplication detection based on image segmentation and keypoint contexts. *Multimedia Tools and Applications* 77(11):14,241–14,258. <https://doi.org/10.1007/s11042-017-5027-9>
15. Lindeberg T (2008) Scale-space: a framework for handling image structures at multiple scales. *Encyclopedia of Computer Science and Engineering* 4:2495–2504. <https://doi.org/10.1002/9780470050118.ecse609>
16. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2):91–110. <https://doi.org/10.1023/B:VISI.0000029664.99615.94>

17. Mahdi M, Alsaad S (2020) Detection of Copy-Move Forgery in Digital Image Based on SIFT Features and Automatic Matching Thresholds
18. Mahmood T, Mahmood Z, Shah M, Khan Z (2018) An efficient forensic technique for exposing region duplication forgery in digital images. *Applied Intelligence* 48(7):1791–1801. <https://doi.org/10.1007/s10489-017-1038-5>
19. Malviya AV, Ladhake SA (2016) Pixel based image forensic technique for copy-move forgery detection using auto color correlogram. In: *Procedia Computer Science* 79:383–390, proceedings of International Conference on Communication Computing and Virtualization (ICCCV) 2016 <https://doi.org/10.1016/j.procs.2016.03.050>
20. Niyishaka P, Bhagvati C (2018) Digital Image Forensics Technique for Copy-Move Forgery Detection Using DoG and ORB: International Conference. In: , *ICCVG 2018*, Warsaw, Poland, September 17 - 19, 2018 Proceedings 472–483. https://doi.org/10.1007/978-3-030-00692-1_41
21. Ojeniyi J, O Adedayo B, Idris I, Abdulhamid S (2018) Hybridized technique for copy-move forgery detection using discrete cosine transform and speeded-up robust feature techniques. *International Journal of Image Graphics and Signal Processing* 10:22–30. <https://doi.org/10.5815/ijigsp.2018.04.03>
22. Pavlović A, Glišović N, Gavrovska A, et al. (2019) Copy-move forgery detection based on multifractals. *Multimed Tools Appl* 78:20655–20678. <https://doi.org/10.1007/s11042-019-7277-1>
23. Redi JA, Taktak W, Dugelay JL (2011) Digital image forensics: a booklet for beginners. *Multimedia Tools and Applications* 51(1):133–162. <https://doi.org/10.1007/s11042-010-0620-1>
24. Saini GK, Mahajan M (2016) Study of copy move image forgery detection based on surf algorithm
25. Singh M, Singh E (2016) Detection of cloning forgery images using surf+ dwt and pca. *International Journal of Latest Engineering Research and Applications (IJLERA)* 1:1–10
26. Sobel I (2014) An isotropic 3x3 image gradient operator. *Presentation at Stanford AI Project* 1968:271–272
27. Soni B, Das PK, Thounaojam DM (2018) Cmfd: a detailed review of block based and key feature based techniques in image copy-move forgery detection. *IET Image Processing* 12(2):167–178. <https://doi.org/10.1049/iet-ipr.2017.0441>
28. Tareen SAK, Saleem Z (2018) A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. pp 1–10 <https://doi.org/10.1109/ICOMET.2018.8346440>
29. Warif NBA, Wahab AWA, Idris MYI, Ramli R, Salleh R, Shamshirband S, Choo KKR (2016) Copy-move forgery detection: Survey, challenges and future directions. *J Network and Computer Applications* 75:259–278
30. Zhu Y, Shen X, Chen H (2016) Copy-move forgery detection based on scaled orb. *Multimedia Tools and Applications* 75(6):3221–3233. <https://doi.org/10.1007/s11042-014-2431-2>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.