Multimedia Tools and Applications (2019) 78: 12405–12419
https://doi.org/10.1007/s11042-018-6792-9

Efficient image splicing detection algorithm based on markov features

Nam Thanh Pham[1] · Jong-Weon Lee[2] · Goo-Rak Kwon[3] · Chun-Su Park[4]

Received: 13 February 2018 / Revised: 16 August 2018 / Accepted: 18 October 2018 /
Published online: 25 October 2018

Abstract
Image splicing is one of the most common methods for digital image tampering. In this paper, an efficient Markov features based algorithm is proposed for image splicing detection. The proposed algorithm first extracts two types of Markov features, coefficient-wise Markov features and block-wise Markov features in the discrete cosine transform (DCT) domain. The former are obtained by exploiting correlations between consecutive coefficients and the latter are computed by exploiting coefficient correlations between adjacent blocks. Then, a feature vector is obtained by combining these two Markov features and it is fed into support vector machine (SVM) for the classification of authentic and spliced images. The experimental results show that the proposed method not only achieves much higher detection accuracy but also reduces the total running time significantly in comparison with state-of-the-art methods.

Keywords Image splicing · Markov features · DCT domain · Support vector machine

1 Introduction

Nowadays, it is quite common for ordinary users to take pictures using portable digital cameras and share them on social networks. With the popularity of digital-image-editing software, more and more images are manipulated before they are shared. As a result, the reliability of images has become very low, and the field of digital forensics has emerged to restore public trust in digital images [30].

Image splicing, which copies one or several regions from the source images and pastes them to the destination image, is one of the most popular image manipulation methods.

✉ Chun-Su Park
cspk@skku.edu

[1] Department of Digital Contents, Sejong University, Seoul, South Korea

[2] Department of Software, Sejong University, Seoul, South Korea

[3] Department of Information and Communication Engineering, Chosun University, Donggu, Gwangju South Korea

[4] Department of Computer Education, Sungkyunkwan University, Seoul, South Korea

Figure 1 shows an example of image splicing [7]. In this example, two images in Fig. 1a and b are authentic. Figure 1c is a spliced image which is composed by copying the zebra in Fig. 1b into Fig. 1a. In the image splicing scenario, a spliced region usually undergoes a sequence of post-processing operations such as edge softening, blurring, denoising and smoothing for a better visual appearance. Further, in some cases, images are processed professionally with the intention of forgery. Therefore, it is often impossible to manually identify the authenticity of images even for practiced users [29, 30].

Image splicing detection (ISD) has been actively studied by many researches to challenge image tampering. Several approaches distinguish spliced images from authentic images by identifying different traces near the boundaries of spliced regions [4, 5, 8, 16, 28, 34]. In [28], two methods for improving the capability of the bicoherence features in detecting image splicing were proposed, which included estimating the bicoherence features of the authentic counterpart and incorporating features that characterize the variance of the feature performance. 2-dimensional (2-D) phase congruency and statistical moments of characteristic functions of wavelet sub-bands were used for splicing detection in [5]. Properties of camera response function are utilized in [4, 16] while they are combined with noise level inconsistency in [35] to detect image manipulations. Human visual system is less sensitive to the chrominance than to the luminance. Even if spliced image looks natural to human, spliced traces could be still left in chrominance, hence, chrominance information of the image was exploited for splicing detection in [34]. Run length based approach was first proposed in [8] and was recently improved in [14, 26]. Gamma transformation was detected in the input image to detect splicing whilst spliced region was localized based on the probability of overlapping blocks and pixels of that image being gamma transformed [33].

Support vector machine (SVM) has been widely used for classification in ISD problem [4, 5, 8, 9, 12–14, 16–18, 26, 28, 31, 33, 34, 36, 37] due to its simplicity and efficiency. To deal with high dimensionality of feature vectors, SVM-recursive feature elimination was used in [13, 37] and principal component analysis (PCA) was used in [9, 26]. Convolutional neural networks (CNNs) showed their efficiency in detecting tampering and localizing spliced regions in recent researches with potential results [1–4]. In order to classify spliced images and authentic images efficiently, along with learning methods, features selection and extraction are noteworthy [22–25].

It was shown in [12] that ISD techniques using Markov features on the transform domain achieved a greatly superior performance in detecting spliced traces compared to other methods using statistical features. In the very first research of Markov features in ISD [31], the statistical features included moments of characteristic functions and Markov transition probabilities, where the latter contributed the most and outperformed the former in splicing detection performances. In this paper, we propose a Markov features based algorithm for image splicing detection. In the discrete cosine transform (DCT) domain, the proposed algorithm extracts Markov features by exploiting correlations between adjacent coefficients

Fig. 1 Example of image splicing. **c**: a spliced image, which is composed from authentic images **a** and **b**

and adjacent blocks. Then, the proposed method performs spliced image classification by using Markov features. The proposed method not only detects spliced images more effectively but also reduces the running time dramatically in comparison with state-of-the-art methods.

The rest of this paper is organized as follows. In Section 2, we briefly review several conventional Markov features based splicing detection methods. In Section 3, we introduce the proposed algorithm in details. Section 4 presents experimental results to show that our algorithm outperforms state-of-the-art methods. Finally, Section 5 concludes the paper and discusses our future works.

2 Related works

In ISD problem, splicing operation usually changes the correlations between image pixels. According to random process theory, Markov random process is a tool to characterize these correlations [31]. Therefore, Markov based feature is a kind of measure which can reflect the statistical changes caused by splicing [36]. Markov features based approaches are successful in ISD problems and still promising to improve. Analysis of several Markov features based methods is given in the remaining of this Section.

The transition probability matrices (TPMs) which obtained from the difference arrays capture pixels or coefficients correlations to detect spliced artifacts [13]. Markov features were extracted in different kinds of domains such as spatial domain [9], DCT domain [9, 12, 13, 17, 31, 36], discrete wavelet transform domain (DWT) [13, 17, 31, 37] and quaternion DCT (QDCT) domain [18].

2-D arrays were generated by applying multi-size block DCT (MBDCT) with statistical features extracted from the input image, and then combined to form an image model for splicing detection in the original method of this research direction [31]. MBDCT was used to extract 168 moment features while 98 Markov features were extracted in horizontal and vertical directions. The statistical features included moments of characteristic functions of wavelet sub-bands and Markov transition probabilities of difference arrays. This method was tested with Columbia image splicing detection evaluation dataset [27], a dataset with low resolution grayscale images and simple tampering operations, and achieved the detection accuracy rate at 91.87%.

Expanded Markov features in DCT domain was proposed to capture correlation caused by 8×8 blocking artifact and more features are constructed in DWT domain to characterize the three kinds of dependency among wavelet coefficients across positions, scales and orientations [13]. The number of features was reduced to 100 and the performance peaked at 93.55% for Columbia dataset [27]. An enhanced threshold method was introduced in [17] to reduce the number of features by thresholding difference arrays to an arithmetic progression with common difference of 2. Nevertheless, Markov feature vectors in [17] were extracted from both DCT and DWT domains, their dimension was relatively high. The weakness of this method is the loss of details of feature vectors due to thresholding process. Consequently, it was shown in [17] that the detection performance was relatively low, i.e. the measured highest detection accuracy was 88.43% for Columbia dataset [27]. Another recently published study [37] also extracted Markov features in many sub-bands of DWT domain and the best detection performance for [27] was 89.88%. In [9], spatial and DCT domains were in charge of extracting Markov features and then PCA was used to reduce the dimensionality of feature vectors. The method in [9] achieved the highest detection result for [27] at 98.82%.

In [12], the maximum value among various directional difference values in the DCT domain of three color channels was used to choose Markov features. Threshold expansion reduced the information loss caused by the coefficient thresholding that was used to restrict the number of Markov features. To compensate the increased number of features owing to the threshold expansion, an even-odd Markov state decomposition algorithm was proposed. For different kinds of features, this method worked best for CASIA TIDE v1.0 [6] with accuracy of more than 97.86%. However, the performances for Columbia dataset [27] and CASIA TIDE v2.0 [7] were 91.98% and 94.50%, respectively.

Color information was extracted from blocked images to construct quaternion and then the QDCT coefficients of quaternion blocked images was obtained [18]. Expanded Markov features were generated from the TPMs in QDCT domain to capture intra-block and inter-block correlation among QDCT block coefficients. The detection performances were approximately 95% and 92% for two datasets CASIA TIDE v1.0 and v2.0 [6, 7], respectively.

3 Markov feature extraction

In this Section, we present how to extract 2 types of Markov features, coefficient-wise Markov features and block-wise Markov features, which are used for detecting spliced images. The former are obtained by exploiting correlations among consecutive DCT coefficients and the latter are computed by exploiting coefficient correlations between adjacent blocks. In DCT domain, coefficients can reflect the changes in the local frequency of the original authentic image, which were created by splicing operation [31]. The following Subsections provide a detailed explanation on Markov features extraction process.

3.1 Coefficient-wise Markov features

Consider an input color image of size $H \times W$, which will be converted to gray scale image, and then divided into non-overlapping 8×8 blocks. In the proposed method, 2-D DCT is applied to each 8×8 block. All DCT coefficients are taken their absolute values and then rounded to the nearest integers. Let us denote the image containing these coefficients by $\mathbf{I}$. The proposed method computes coefficient-wise difference arrays by subtracting consecutive DCT coefficients and then extracts coefficient-wise Markov features using these arrays.

Coefficient-wise difference arrays are computed in 4 directions as shown in Fig. 2. Let us denote 4 coefficient-wise difference arrays obtained in horizontal, vertical, main diagonal,

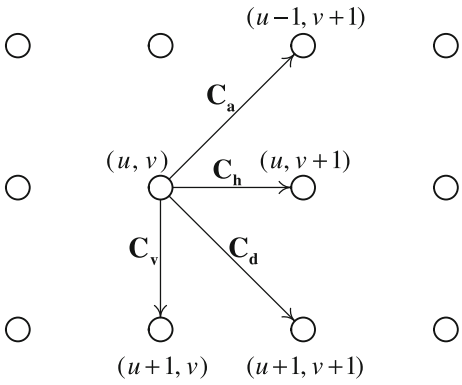

Fig. 2 4 directions used for calculating coefficient-wise difference arrays

and anti-diagonal directions as $\mathbf{C_h}$, $\mathbf{C_v}$, $\mathbf{C_d}$ and $\mathbf{C_a}$, respectively. For example, $\mathbf{C_h}$ is calculated by subtracting each coefficient of $\mathbf{I}$ from its nearest neighbor in horizontal direction as follows

$$C_h(u, v) = I(u, v) - I(u, v + 1) \tag{1}$$

where $u = 1, 2, \ldots, H$ and $v = 1, 2, \ldots, W - 1$. $\mathbf{C_v}$, $\mathbf{C_d}$ and $\mathbf{C_a}$ can be obtained similarly with corresponding directions. We apply a thresholding technique for difference arrays to reduce the computational complexity of preceding steps. Let $\tilde{\mathbf{C}}_h$ be the resultant array by thresholding $\mathbf{C_h}$ as follows

$$\tilde{C}_h(u, v) = \begin{cases} T_1, & \text{if } C_h(u, v) > T_1, \\ -T_1, & \text{if } C_h(u, v) < -T_1, \\ C_h(u, v), & \text{otherwise}, \end{cases} \tag{2}$$

where T_1 is a positive integer. Similarly, $\tilde{\mathbf{C}}_v$, $\tilde{\mathbf{C}}_d$ and $\tilde{\mathbf{C}}_a$ can be derived by thresholding $\mathbf{C_v}$, $\mathbf{C_d}$ and $\mathbf{C_a}$ using (2), respectively.

Next, we compute TPMs using $\tilde{\mathbf{C}}_h$, $\tilde{\mathbf{C}}_v$, $\tilde{\mathbf{C}}_d$ and $\tilde{\mathbf{C}}_a$ to represent the characteristics of these difference arrays. Specifically, TPMs are composed by probabilities of elements in $\tilde{\mathbf{C}}_h$, $\tilde{\mathbf{C}}_v$, $\tilde{\mathbf{C}}_d$ and $\tilde{\mathbf{C}}_a$ to be particular values given the values of their nearest neighbors in horizontal, vertical, diagonal and anti-diagonal directions, respectively. These probabilities capture the dependencies among neighboring DCT coefficients in corresponding direction. Let us define $\mathbf{P_h^C}$ as the horizontal TPM of $\tilde{\mathbf{C}}_h$. The element at row i and column j of this matrix, $P_h^C(i, j)$, is calculated by the following formula

$$P_h^C(i, j) = \frac{\sum_{u=1}^{H} \sum_{v=1}^{W-2} \delta(\tilde{C}_h(u, v) = i) \times \delta(\tilde{C}_h(u, v + 1) = j)}{\sum_{u=1}^{H} \sum_{v=1}^{W-2} \delta(\tilde{C}_h(u, v) = i)}, \tag{3}$$

where $i, j = -T_1, -T_1 + 1, \ldots, 0, \ldots, T_1 - 1, T_1$ and

$$\delta(a = b) = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{otherwise}. \end{cases} \tag{4}$$

Note that, in (3), the probability is assigned as 0 if the denominator in the right-hand side is 0. Each pair of i and j is corresponding to one transition probability of the TPM, which is equivalent to one Markov feature. Similarly, $\mathbf{P_v^C}$, $\mathbf{P_d^C}$ and $\mathbf{P_a^C}$ can be derived from (3) and (4). Then, we obtain $4(2T_1 + 1)^2$ coefficient-wise Markov features by subsequently combining all elements of 4 TPMs.

3.2 Block-wise Markov features

In this Subsection, we present how the block-wise Markov features are extracted using $\mathbf{I}$. In 8×8 DCT blocks, coefficients are arranged in zigzag order, from the DC coefficient at the top-left, followed by AC coefficients where low-frequency coefficients, which are more likely to be nonzero, are placed before high-frequency coefficients [32]. Moreover, we aim to generate block-wise Markov features which reflect strong correlations between adjacent blocks. For those reasons, DCT coefficients which are close to the bottom row or the rightmost column of 8×8 DCT blocks are not employed in computing block-wise difference arrays in the proposed method. In the previous Subsection, coefficient-wise difference arrays are computed using all DCT coefficients. By contrast, block-wise difference arrays

are calculated by using only coefficients of top-left $n \times n$ sub-blocks in DCT blocks where $n < 8$.

We introduce an algorithm which constructs a top-left $n \times n$ sub-block by exploiting coefficients of an 8×8 DCT block. For each 8×8 block, the first n^2 coefficients in the 8×8 zigzag order are taken and then rearranged in the $n \times n$ zigzag order to construct the top-left sub-block. The details of sub-block construction process are presented in Fig. 3. After all of sub-blocks are obtained, they are utilized for computing block-wise difference arrays to capture correlations between adjacent blocks. Figure 4 illustrates this algorithm for $n = 4$. The first 16 coefficients of 8×8 DCT block are denoted by letters from a to p where 10 coefficients from a to j remain and 6 coefficients from k to p replace the last 6 coefficients of the zigzag sequence of top-left 4×4 sub-block in corresponding order.

The block-wise Markov features are extracted from horizontal block-wise difference array $\mathbf{B_h}$ and vertical block-wise difference array $\mathbf{B_v}$. Figure 5 shows an example of calculating $\mathbf{B_h}$ for $n = 4$, where adjacent large dotted squares represent 8×8 blocks of $\mathbf{I}$ and small solid squares which are located inside large squares represent top-left 4×4 sub-blocks. Solid arrows connect pairs of top-left 4×4 sub-blocks and we subtract coefficients of sub-blocks on the ending from corresponding coefficients of sub-blocks on the beginning of solid arrows to obtain 4×4 blocks on the right, which are indicated by dashed arrows. In general case, the obtained blocks are combined to form $\mathbf{B_h}$. The size of $\mathbf{B_h}$ and $\mathbf{B_v}$ is $K \times L$ where $K = nH/8$ and $L = nW/8$. Let us define $\mathbf{B_h}$ as

$$B_h(p, q) = I(u, v) - I(u, v + 8), \tag{5}$$

```
1: procedure SUB-BLOCK CONSTRUCTION(B, n)        ▷ B is an 8 × 8 block; 3 ≤ n ≤ 6
2:     if n mod 2 = 0 then
3:         x ← n + 1, y ← 1
4:     else
5:         x ← 1, y ← n + 1
6:     end if
7:     S ← ∅
8:     S.push(B(x, y))
9:     for i ← 1, n(n − 1)/2 − 1 do
10:        if x = 1 and (x + y) mod 2 = 0 then
11:            y ← y + 1
12:        else if y = 1 and (x + y) mod 2 = 1 then
13:            x ← x + 1
14:        else if y > 1 and (x + y) mod 2 = 1 then
15:            x ← x + 1, y ← y − 1
16:        else if x > 1 and (x + y) mod 2 = 0 then
17:            x ← x − 1, y ← y + 1
18:        end if
19:        S.push(B(x, y))
20:    end for
21:    u ← n, v ← n
22:    while S ≠ ∅ do
23:        B(u, v) ← S.pop()
24:        if u = n and (u + v) mod 2 = 0 then
25:            v ← v − 1
26:        else if v = n and (u + v) mod 2 = 1 then
27:            u ← u − 1
28:        else if v < n and (u + v) mod 2 = 1 then
29:            u ← u − 1, v ← v + 1
30:        else if u < n and (u + v) mod 2 = 0 then
31:            u ← u + 1, v ← v − 1
32:        end if
33:    end while
34: end procedure
```

Fig. 3 Algorithm constructing top-left $n \times n$ sub-block by using the first n^2 coefficients in the zigzag order for each 8×8 DCT block

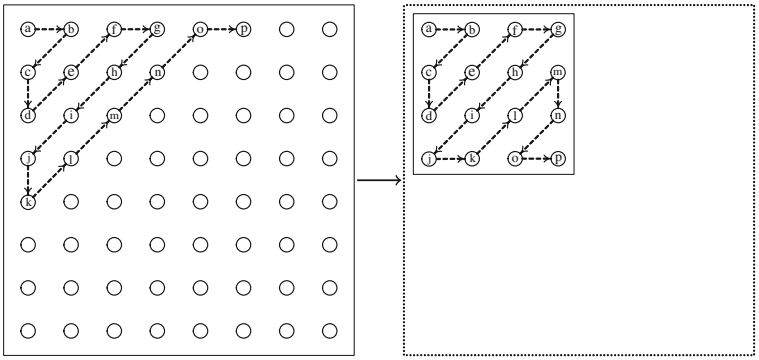

Fig. 4 A top-left 4×4 sub-block (right) is constructed from the first 16 coefficients in the zigzag sequence of an 8×8 DCT block (left)

where $p = 1, 2, \ldots, K$; $q = 1, 2, \ldots, L$; r and s are the remainders when p and q are divided by n, respectively; $u = \left\lfloor \frac{p}{n} \right\rfloor \times 8 + r$ and $v = \left\lfloor \frac{q}{n} \right\rfloor \times 8 + s$. Similarly, the vertical block-wise difference array $\mathbf{B_v}$ is obtained. $\mathbf{B_h}$ and $\mathbf{B_v}$ are also applied a thresholding technique. Let $\tilde{\mathbf{B}}_h$ be the resultant array by thresholding $\mathbf{B_h}$ as follows

$$\tilde{B}_h(p,q) = \begin{cases} T_2, & \text{if } B_h(p,q) > T_2, \\ -T_2, & \text{if } B_h(p,q) < -T_2, \\ B_h(p,q), & \text{otherwise,} \end{cases} \tag{6}$$

where T_2 is a positive integer. Similarly, $\tilde{\mathbf{B}}_v$ is obtained from $\mathbf{B_v}$ using (6). The horizontal TPM of $\tilde{\mathbf{B}}_h$ is denoted by $\mathbf{P_h^B}$. An element of this TPM, $P_h^B(i, j)$, is obtained as follows

$$P_h^B(i,j) = \frac{\sum_{p=1}^{K} \sum_{q=1}^{L-1} \delta(\tilde{B}_h(p,q) = i) \times \delta(\tilde{B}_h(p,q+1) = j)}{\sum_{p=1}^{K} \sum_{q=1}^{L-1} \delta(\tilde{B}_h(p,q) = i)}, \tag{7}$$

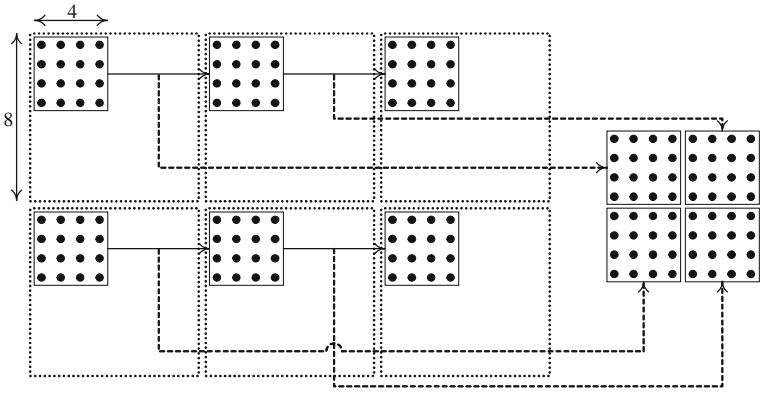

Fig. 5 Illustration of calculating horizontal block-wise difference array for $n = 4$

where $i, j = -T_2, -T_2+1, \ldots, 0, \ldots, T_2-1, T_2$. Similarly, we can obtain $\mathbf{P}_{\mathbf{v}}^{\mathbf{B}}$, the vertical TPM of $\tilde{\mathbf{B}}_v$. There are $2(2T_2+1)^2$ more block-wise Markov features, resulting in $4(2T_1+1)^2 + 2(2T_2+1)^2$ dimensional feature vector for classification. Fig. 6 depicts the overall features extraction process of the proposed method.

4 Experiments

4.1 Dataset and evaluation criteria

There exist several benchmarking datasets for evaluating the performances of ISD algorithms [6, 7, 27]. In our simulations, we used the challenging dataset CASIA TIDE v2.0 [7], which was widely used in recent researches of ISD topic [12, 13, 18]. This dataset contains 7491 authentic and 5123 tampered color images with various sizes from 240 × 160 to 900 × 600 pixels. In comparison with Columbia dataset [27] and CASIA TIDE v1.0 [6], CASIA TIDE v2.0 is a larger dataset with more realistic spliced images owing to complex post-processing operations across tampered regions. We randomly create the same number of authentic images and spliced images from this dataset for our experiments.

We adopt the metric accuracy [10] to quantitatively evaluate the detection performance of the proposed method. Accuracy, denoted by *acc*, is determined measuring the proportion of correctly classified images

$$acc = \frac{TP + TN}{S}, \tag{8}$$

where TP is the number of spliced images that are correctly detected as spliced images, TN is the number of authentic images that are correctly detected as authentic images and S is the total number of testing images.

4.2 Classification using Support Vector Machine

In the proposed method, SVM and radial basis function kernel are used to classify the images. We also use 10-fold cross validation to avoid overfitting and to get more accurate estimation of the performance [15]. Specifically, the dataset is divided into 10 equal subsets where the numbers of authentic and spliced images in each subset are identical. There are 10 folds, each fold is a test in which a subset is chosen as the testing set meanwhile 9 remaining subsets are combined to be the training set. All of the images in the dataset are totally different, therefore, there is no image in the testing set which has been trained before.

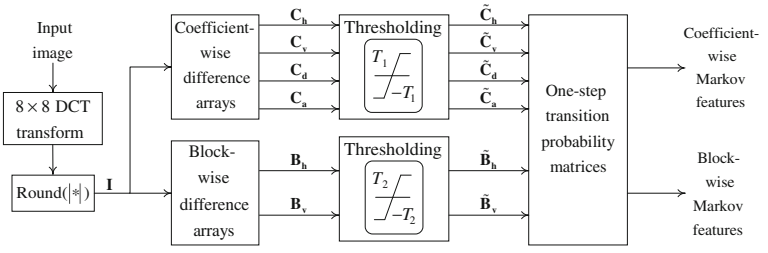

Fig. 6 Markov features extraction model

The feature vector should be normalized before classification [15]. In our experiments, we linearly scale both training and testing $1 \times m$ feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ to zero-mean vector $\mathbf{x}' = [x'_1, x'_2, \ldots, x'_m]$ by the following formula

$$x'_i = \frac{x_i - \mu(\mathbf{x})}{\sigma(\mathbf{x})}, \tag{9}$$

where $i = 1, 2, \ldots, m$; $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are mean and standard deviation of vector $\mathbf{x}$, respectively.

4.3 Discussion on parameters selection

We give a brief exposition of how parameters are chosen in the experiments of the proposed method. We choose 2 different variables for thresholding the difference arrays: T_1 for coefficient-wise difference arrays and T_2 for block-wise difference arrays. In 8×8 DCT blocks, coefficients are arranged in zigzag order, from the DC coefficient at the top-left, followed by AC coefficients where low-frequency coefficients (which are more likely to be nonzero) are placed before high-frequency coefficients (which are more likely to be zero) [32]. The elements of coefficient-wise difference arrays are formed from subtracting all coefficients of DCT blocks. On the contrary, the elements in block-wise difference arrays are the differences of DCT coefficients of top-left sub-blocks of adjacent 8×8 DCT blocks. Since the first n^2 DCT coefficients in zigzag order of each block are usually large and they may vary in a vast range, the numbers in block-wise difference arrays vary widely and unpredictably. Hence, we set $T_1 \leq T_2$.

In order to select appropriate values for T_1 and T_2, there are some factors should be taken into account. If these thresholds are too small, it is difficult to capture the spliced artifacts. By contrast, if T_1 and T_2 are too large, the dimensionality of feature vectors will be very large, thus the computational cost might be unmanageable. As a consequence, there will be a trade-off between detection accuracy and computational complexity of the algorithm in the choice of T_1 and T_2. Empirically, we choose $T_1 = 4$ and $T_2 = 4, 5$ in our simulation.

In the proposed method, $n = 8$ is the special case when the all the DCT coefficients are used to extract block-wise Markov features and DCT blocks are not affected by the sub-block construction algorithm. In Section 3.2, we explained why we use top-left $n \times n$

Table 1 Percentages of detection accuracy of the proposed method for $T_2 = 4$

Fold	$n = 3\ A$	$n = 3\ B$	$n = 4\ A$	$n = 4\ B$	$n = 5\ A$	$n = 5\ B$	$n = 6\ A$	$n = 6\ B$	$n = 8$
1	95.99	95.99	95.26	96.72	95.99	98.18	92.70	95.62	95.62
2	96.72	93.07	95.62	96.35	95.62	97.81	94.16	94.89	95.62
3	93.43	95.62	96.35	96.72	97.08	94.53	97.08	95.62	95.26
4	94.53	97.45	97.45	93.43	94.89	95.26	93.43	95.26	97.45
5	95.62	94.89	93.43	95.62	96.35	98.18	96.35	92.70	97.45
6	98.54	94.89	96.72	98.91	93.07	97.45	97.45	88.69	98.18
7	97.45	93.80	97.45	93.80	98.54	96.35	94.16	95.99	96.35
8	95.26	95.26	96.35	94.53	96.35	96.72	94.16	96.35	96.35
9	96.35	97.81	95.99	95.62	95.99	95.62	95.26	97.08	95.99
10	95.26	95.99	95.99	95.99	93.80	98.54	95.99	97.08	89.42
Average	95.92	95.48	96.06	95.77	95.77	96.86	95.07	94.93	95.77

Table 2 Percentages of detection accuracy of the proposed method for $T_2 = 5$

Fold	$n = 3$ A	$n = 3$ B	$n = 4$ A	$n = 4$ B	$n = 5$ A	$n = 5$ B	$n = 6$ A	$n = 6$ B	$n = 8$
1	95.62	95.99	96.72	94.53	96.35	97.45	96.72	95.26	94.16
2	97.45	96.35	95.62	96.35	97.08	97.08	95.99	95.99	97.81
3	95.26	95.26	95.62	95.62	95.99	95.26	97.08	95.99	97.81
4	95.62	93.43	94.16	95.99	97.45	95.99	96.72	96.72	94.16
5	94.53	93.80	95.62	93.43	97.81	96.72	96.35	95.26	95.26
6	97.81	96.35	97.08	98.18	96.72	97.81	93.80	96.72	98.54
7	97.45	97.08	96.72	97.81	98.18	97.08	96.72	94.89	97.81
8	95.26	95.26	94.53	95.99	95.26	97.45	95.62	95.99	94.89
9	95.99	95.62	95.26	97.45	98.18	95.62	97.08	95.99	95.62
10	96.72	96.35	94.53	96.72	95.99	97.81	97.81	97.45	96.72
Average	96.17	95.55	95.59	96.21	96.90	96.83	96.39	96.03	96.28

sub-block of 8×8 DCT blocks to calculate block-wise difference arrays. Furthermore, the high value of n can cause the very large number of computations, and hence, the running time would increase. In contrast, if n is small, for example $n \leq 2$, we may not get enough important coefficients to capture strong correlations among adjacent blocks. As a result, in the simulation, we set $n = 3, 4, 5, 6$ and the experimental results in Subsection 4.4 will show the optimized value of n for the proposed method.

4.4 Experimental results

To investigate performance of the proposed method, in all different parameters selections, T_1 is set to 4. We compare the detection accuracy with different values of n and T_2. In each scenario, we perform the simulations for two cases, Case A and Case B. In Case A, the top-left sub-blocks are constructed using sub-block construction algorithm and they are used to calculate block-wise difference arrays. By contrast, in Case B, the original top-left sub-blocks of 8×8 DCT blocks are used for computing the difference arrays. Table 1 and Table 2 show the detection performance of the proposed method for $T_2 = 4$ and $T_2 = 5$, respectively. Specifically, Case A achieves higher average detection accuracy than Case B

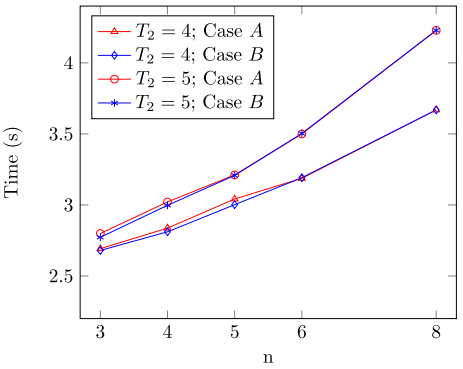

Fig. 7 Average processing time per image of the proposed method for different cases

Table 3 Comparison of average running time per image with state-of-the-art methods

Feature vector	Feature vector dimension	Extraction time (s)	Selection time (s)	Total time (s)	Accuracy (%)
He et al. [13]	100	2.218	2.158	4.376	89.76
Li et al. [18]	972	4.61	0	4.61	92.38
$T_2 = 4$, $n = 3$	486	2.692	0	2.692	95.92
$T_2 = 5$, $n = 5$	566	3.211	0	3.211	96.90

for $T_2 = 4$; $n = 3, 4, 6$ and $T_2 = 5$; $n = 3, 5, 6$, which proves the efficiency of sub-block construction algorithm. Note that, for $n = 8$, Case A and Case B are identical; the accuracy obtained for this case is lower than for $n = 5$ in both Cases A and B when $T_2 = 4$ (Table 1) and $T_2 = 5$ (Table 2). In addition, the proposed method performs well in terms of detection accuracy when $n = 5$ and it reaches the highest accuracy of 96.90% when $T_2 = 5$; $n = 5$.

In the proposed method, the computation of block-wise difference arrays using $n \times n$ sub-blocks instead of 8×8 blocks helps reducing the number of calculations considerably, hence, the computational complexity, the feature extraction time and the total running time drop drastically. Figure 7 shows that the average processing time for each image increases when T_2 or n increases. Moreover, these quantities for Case A are slightly higher than those of Case B with the same value of T_2.

As can be seen from Table 3, the proposed method performs better than state-of-the-art methods [13, 18] in terms of not only the running time but also the detection accuracy. The feature vector dimensions of He et al. [13] are 2250, 4410 and 7290 for different parameters, therefore, they used SVM recursive feature elimination to reduce the number of features to 100. We compare the processing time per image, which is sum of feature extraction time and feature selection time with previous works. For $T_2 = 4$; $n = 3$, we observe the fastest run of the proposed method in different parameters setups where the running time is approximately 60% and the detection accuracy is much higher in comparison with [13, 18]. In addition, the detection accuracy of the proposed method for $T_2 = 5$; $n = 5$ is from about 4.5% to 7% higher than that of those methods with shorter running time.

5 Conclusions and Discussion

In this paper, a novel Markov features based method in DCT domain is introduced for detecting spliced images. The coefficient-wise Markov features contribute considerably to the detection accuracy of the proposed method. Further, the newly proposed sub-block construction algorithm of extracting correlation between neighboring blocks can not only increase the detection accuracy notably but it can also decrease the computational complexity of the whole splicing detection process. Experimental results demonstrate that the proposed method succeeds in reducing the total running time and improving detection accuracy in comparison with state-of-the-art methods using the same data testing set. Despite the fact that spliced images can be detected with high precision, one drawback of the proposed method is that spliced regions are not specified.

In future works, we plan to solve the optimization problem between detection accuracy and computational complexity to improve the performance of splicing detection [11, 19–21] and to apply CNNs for image splicing localization [1–3, 11, 19–21].

Acknowledgements This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00312) supervised by the IITP (Institute for Information & communications Technology Promotion). This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2016R1C1B1009682).

References

1. Bappy JH, Roy-Chowdhury AK, Bunk J, Nataraj L, Manjunath BS (2017) Exploiting spatial structure for localizing manipulated image regions. In: Proceedings of the IEEE international conference on computer vision, pp 4970–4979
2. Bondi L, Lameri S, Guera D, Bestagini P, Delp EJ, Tubaro S (2017) Tampering detection and localization through clustering of Camera-Based CNN features. In: IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 1855–1864
3. Bunk J, Bappy JH, Mohammed TM, Nataraj L, Flenner A, Manjunath BS, Chandrasekaran S, Roy-Chowdhury AK, Peterson L (2017) Detection and localization of image forgeries using resampling features and deep learning, pp 1881–1889
4. Chen C, McCloskey S, Yu J (2017) Image splicing detection via camera response function analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5087–5096
5. Chen W, Shi YQ, Su W (2007) Image Splicing Detection Using 2-D Phase Congruency and Statistical Moments of Characteristic Function. SPIE 6505, Security, Steganography, and Watermarking of Multimedia Contents IX https://doi.org/10.1117/12.704321
6. Dong J, Wang W (2011) CASIA tampered image detection evaluation database 1.0
7. Dong J, Wang W, Tan T (2013) Casia image tampering detection evaluation database. In: IEEE international conference on signal and information processing, pp 422–426
8. Dong J, Wang W, Tan T, Shi YQ (2009) Run-Length And edge statistics based approach for image splicing detection. International Workshop on Digital Watermarking, LNCS 5450:76–87
9. El-Alfy E-SM, Qureshi MA (2015) Combining spatial and DCT based markov features for enhanced blind detection of image splicing. Journal of Pattern Analysis and Applications 18(3):713–723
10. Fawcett T (Jun. 2006) An introduction to ROC analysis. Pattern Recogn Lett 27(8):861–874
11. Geng Y, Liang RZ, Li W, Wang J, Liang G, Xu C, Wang JY (2016) Learning Convolutional Neural Network to Maximize Pos@Top Performance Measure. arXiv:1609.08417
12. Han JG, Park TH, Moon YH, Eom IK (2016) Efficient markov features extraction method for image splicing detection using maximization and threshold expansion. J Electron Imaging 25(2):023031
13. He Z, Lu W, Sun W, Huang J (2012) Digital image splicing detection based on markov features in DCT and DWT domain. Pattern Recogn 45(12):4292–4299
14. He Z, Sun W, Lu W, Lu H (2011) Digital image splicing detection based on approximate run length. Pattern Recogn Lett 32(12):1591–1597
15. Hsu CW, Chang CC, Lin CJ (2007) A practical guide to support vector classification. Technical Report, National Taiwan University 1:1–16
16. Hsu YF, Chang SF (2007) Image Splicing Detection Using Camera Response Function Consistency and Automatic Segmentation. IEEE International Conference on Multimedia and Expo, https://doi.org/10.1109/ICME.2007.4284578
17. Kumar A, Prakash CS, Maheshkar S, Maheshkar V (2019) " Markov Feature Extraction Using Enhanced Threshold Method For Image Splicing Forgery Detection," Smart Innovations in Communication and Computational Sciences. Springer, Singapore, pp 17–27
18. Li C, Ma Q, Xiao L, Li M, Zhang A (2017) Image splicing detection based on markov features in QDCT domain. Journal of Neurocomputing 228:29–36
19. Li Q, Zhou X, Gu A, Li Z, Liang RZ (2018) Nuclear norm regularized convolutional max Pos@Top machine. Journal of Neural Computing and Applications 30(2):463–472
20. Liang RZ, Shi L, Wang H, Meng J, Wang JJ, Sun Q, Gu Y (2016) Optimizing top precision performance measure of Content-Based image retrieval by learning similarity function. In: IEEE international conference on pattern recognition (ICPR), pp 2954–2958

21. Liang RZ, Xie W, Li W, Wang H, Wang JJ, Taylor L (2016) A novel transfer learning method based on common space mapping and weighted domain matching. In: IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp 299–303
22. Liu Y, Nie L, Han L, Zhang L, Rosenblum DS (2015) Action2activity: recognizing complex activities from sensor data. In: Proceedings of AAAI conference on artificial intelligence, pp 1617–1623
23. Liu Y, Nie L, Liu L, Rosenblum DS (2016) From action to activity: sensor-based activity recognition. Journal of Neurocomputing 181:108–115
24. Liu Y, Zhang L, Nie L, Yan Y, Rosenblum DS (2016) Fortune teller: predicting your career path. In: Proceedings of AAAI conference on artificial intelligence, pp 201–207
25. Liu Y, Zheng Y, Liang Y, Liu S, Rosenblum DS (2016) Urban water quality prediction based on multi-task multi-view learning. In: Proceedings of international joint conference on artificial intelligence, pp 2576–2582
26. Moghaddasi Z, Jalab HA, Noor RM, Aghabozorg S (2014) Improving RLRN Image Splicing Detection with The Use of PCA and Kernel PCA. The Scientific World Journal, https://doi.org/10.1155/2014/606570
27. Ng TT, Chang SF (2004) A Data Set of Authentic and Spliced Image Blocks, ADVENT Technical report Columbia University
28. Ng TT, Chang SF, Sun Q (2004) Blind Detection of Photomontage Using Higher Order Statistics," IEEE International Symposium on Circuits and Systems https://doi.org/10.1109/ISCAS.2004.1329901
29. Park CS, Choeh JY (2017) Fast and Robust Copy-Move Forgery Detection Based on Scale-Space Representation. Journal of Multimedia Tools and Applications, https://doi.org/10.1007/s11042-017-5248-y
30. Park CS, Kim CJ, Lee JH, Kwon GR (2016) Rotation and scale invariant upsampled Log-Polar fourier descriptor for Copy-Move forgery detection. Journal of Multimedia Tools and Applications 75(23):16577–16595
31. Shi YQ, Chen C, Chen W (2007) A Natural Image Model Approach to Splicing Detection. ACM Workshop on Multimedia & Security https://doi.org/10.1145/1288869.1288878
32. Wallace GK (1992) The JPEG still picture compression standard, vol 38
33. Wang P, Liu F, Yang C, Luo X (2018) Blind forensics of image gamma transformation and its application in splicing detection. J Vis Commun Image Represent 55:80–90
34. Wang W, Dong J, Tan T (2009) Effective Image Splicing Detection Based on Image Chroma. IEEE International Conference on Image Processing (ICIP), https://doi.org/10.1109/ICIP.2009.5413549
35. Yao H, Wang S, Zhang X, Qin C, Wang J (2017) Detecting image splicing based on noise level inconsistency. Multimedia Tools and Applications 76(10):12457–12479
36. Zhang J, Zhao Y, Su Y (2009) A new approach merging markov and DCT features for image splicing detection. IEEE International Conference on Intelligent Computing and Intelligent Systems 4:390–394
37. Zhang Q, Lu W, Wang R, Li G (2018) Digital image splicing detection based on markov features in block DWT domain. J Multimed Tools Appl 77(23):31239–31260

Nam Thanh Pham received his B.S. and M.S. in computer science from Vietnam National University, Hanoi, Vietnam in 2012 and 2015, respectively. From 2015 to 2016, he was a visiting researcher at National Institute of Informatics, Tokyo, Japan. He is currently a Ph.D. candidate at Sejong University, Seoul, Korea. His research interests are in the areas of image processing, image forensics, and computer vision.

Jong Weon Lee received M.S. degree in Electrical and Computer Engineering from University of Wisconsin at Madison in 1991, and Ph.D. degree from University of Southern California in 2002. He is presently Professor of Department of Software at Sejong University. His research interests include virtual reality, augmented reality and human-computer interaction.

Goo-Rak Kwon received the Ph.D. degree at the Department of Mechatronic Engineering of Korea University in 2007 and the M.S. degree in the School of Electrical and Computer Engineering at the SungKyunKwan University in 1999. He has also served as Chief Executive Officer and Director of Dalitech Co. Ltd. from May 2004 to Feb. 2007. He joined the Department of Electronic Engineering at Korea University where he was a Postdoc supporting the BK21 Information Technique Business from Mar. 2007 to Feb. 2008. At present, he is working an associate professor at Chosun University.

Chun-Su Park received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, in 2003 and 2009, respectively. From 2009 to 2010, he was a visiting scholar with the Signal and Image Processing Institute, University of Southern California, Los Angeles. He was a senior research engineer at Samsung Electronics from 2010 to 2012. From 2012 to 2014, he was an assistant professor with Dep. of Info. and Telecom. Eng. at Sangmyung University. In 2014, he joined Dep. of Software at Sejong University where he is currently an assistant professor. His research interests are in the areas of video signal processing, parallel computing, and multimedia communications.