IET Image Processing

Research Article

IET Journals
The Institution of Engineering and Technology

Frame duplication/mirroring detection method with binary features

ISSN 1751-9659
Received on 28th May 2016
Revised 15th December 2016
Accepted on 12th February 2017
E-First on 24th March 2017
doi: 10.1049/iet-ipr.2016.0321
www.ietdl.org

Guzin Ulutas[1] ✉, Beste Ustubioglu[1], Mustafa Ulutas[1], Vasif Nabiyev[1]

[1]*Department of Computer Engineering, Karadeniz Technical University, Muhendislik Fakultesi, Trabzon, Turkey*

✉ *E-mail: guzin@ieee.org*

Abstract: Multimedia devices have become increasingly popular due to high quality and low cost products using advanced technology. These devices can capture multimedia files, which can be modified easily by video editing tools. One of the most frequently encountered forgery types in video forensics is the frame duplication (FD) forgery. Many methods have been proposed in the literature to deal with this type of forgery. These methods do not consider frame-mirroring (FM) attack which copy a sequence of frames and paste its mirrored versions somewhere else on the same video. A new FD/FM detection method is proposed in this work. The method extracts binary features from frames and determines the similarity among features. Peak-signal-to-noise ratio of the candidate frames is used to eliminate some of the large number of candidates to improve the detection of the forged frames. Experimental results show that the proposed method successfully detects FM/FD attacks and also yields better execution time and detection results compared to similar works reported in the literature.

1 Introduction

Many multimedia devices such as digital cameras, mobile phones and camcorders have become indispensable gadgets, thanks to the recent advances in technology. These gadgets can capture very high-resolution multimedia files and store them in high capacity flash storage. Wide availability of user-friendly multimedia editing tools has simplified image or video forgery to an extent, where even a novice can modify a video file. Therefore, digital video forensics has emerged to authenticate the trustworthiness of videos and it is used to determine the forged sequences in the videos.

The methods in digital video forensics fall into two categories: active and passive methods. The methods in the first category embed a watermark, specially created for the content, in the video while it is generated. However, many multimedia devices, constrained by the manufacturing cost, do not have such a module to embed a watermark. The techniques in the second category do not need any information to authenticate videos. They use statistical features of frames or group of frames to determine a possible forgery. The methods in the second category have gained popularity compared to others [1–10] recently. The achievements in this category can be grouped into three categories: camera-based, format-based and spatial–temporal analysis-based methods.

Camera-based methods use sensor artefacts and statistical properties of the noise residue. The authors extract noise residue of each frame and uses temporal noise correlation in [2]. Their method also proposed a statistical classification scheme based on a Gaussian mixture model. Kobayashi *et al.* [3] used noise characteristics to detect inconsistent regions in their work. Their method calculates noise characteristics of each pixel by using temporal averaging. Kobayashi *et al.* [4] extended their work in 2010. Their method estimates noise level function in an authentic region and then classifies pixels into two groups: authentic or tampered. In [5], Wang and Farid showed that the motion between fields of a single frame and across fields of neighbouring frames must be same for interlaced videos. Their method was only used for interlaced videos.

The methods in the second category use some codec properties to detect the forgery. Wang and Farid state that when a video is recompressed after forgery, it contains some spatial and temporal artefacts because of double Moving Picture Experts Group (MPEG) compression [6]. In 2008, Luo *et al.* [7, 8] extended their previous work in and used block artefacts to detect the video forgery. In 2009, Wang and Farid [9] stressed that double compression of MPEG videos introduced double quantisation effect. Statistical artefacts may indicate a possible forgery. In [10], the authors use the histogram of oriented gradients feature matching and MPEG compression properties. This method is robust against various signal-processing manipulations.

Spatial–temporal analysis-based methods extract some features from the frames and compare them using a similarity measure to determine forgery. The method in [11] is the first method proposed in this category to detect video forgery. Their method splits the video into overlapping (by one frame) n frame sub-sequences and computes temporal correlation matrix for all. In the first step, the method decides similarity between frames of sub-sequences based on correlation coefficient above a certain threshold. Then, the method computes spatial correlation matrix of all non-overlapping blocks from candidate frames. High temporal and spatial correlation is considered as a forgery in the video. After this, Lin and Tsay [12] proposed a coarse-to-fine approach: candidate clip selection, spatial correlation calculation and frame duplication (FD) classification in 2011. Their method calculates histogram differences of two adjacent frames in RGB colour space and used it as feature. They used histogram differences as a measure for temporal similarity and block-based similarity to detect spatial similarity to detect FD. Chao *et al.* [13] used optical flow consistency to detect forgery in 2013. Their method detects inter-frame forgery, which disturbs optical flow consistency. They proposed two different algorithms to detect FD and deletion attacks. In [14], Lin and Chang used colour histogram difference between adjacent frames and detect temporal correlation between subgroups. Spatial similarity measurement is also applied to decide the forgery. Experimental results show that their method exhibits better performance compared to [11]. In 2014, Yang *et al.* [15] proposed a singular value decomposition (SVD)-based forgery detection method. They used SVD to extract features from frames. Then Euclidean distance of each frame from the reference frame is calculated to determine temporal similarity. Candidate clips are also investigated with random block matching procedure. Their results have better precision, recall and detection accuracy compared to [11, 14]. Their work also improves execution time of [11, 14].

The methods in the last category use both spatial and temporal similarities to detect a specific type of forgery: FD. This forgery copies a part of the video and pastes it into another part to hide or

Fig. 1 ***(a)*** *Original video sequences,* ***(b)*** *Forged video sequences*

replicate a motion. Even though, it is relatively easy to realise such a forgery, it is difficult to detect visually as indicated in [11]. In this work, we propose a method that falls into the third category. The main considerations for the methods in this category are 'processing time', 'detection accuracy – *DA*', 'precision rate – *PR*' and 'recall rate – *RR*' of the method. In this work, we aim to improve the DA, PR and RR and processing time of the detection algorithm compared to others. Our work also contributes the literature with a new attack in video forensics, called frame mirroring (FM). While the works in [11, 14, 15] cannot detect FM attack, the proposed method detects FM attacks with high *DA*, *PR* and *RR* values.

The remaining of this paper is organised as follows. Section 2 gives the motivation of our method. The details and the results of the algorithm will be given in Sections 3 and 4, respectively. Finally, conclusions and future work are provided in Section 4.

2 Motivation

One of the most frequently encountered forgery types in video forensics is FD, which inserts into or replaces a selected part of the video sequence with another part from the same video. Fig. 1 shows an example of the FD forgery. Figs. 1*a* and *b* show the original and forged video sequences, respectively. The frames between the 7 and 10 are copied and inserted after the third frame to create the forged video given in Fig. 1*b*. Many methods are proposed by the researchers to detect FD forgery recently.

The methods in the literature use both spatial and temporal similarities between frames to determine duplication. General framework of the methods contains similar steps: divide the video sequence into overlapping sub-sequences, extract some robust features and determine candidate sub-sequences according to the features. Candidate sub-sequences are then spatially searched for similarity to decide the forgery as the last step.

The first method in the literature proposed by Wang and Farid [11] constructed correlation matrix from the sub-sequences and investigated the similarity between the correlation matrices to determine the forged sub-sequences. SVD-based method inspired by Wang *et al.*'s method extract singular values from the frames using SVD [15]. The algorithm calculates the Euclidean distance between the vector of singular values of the current frame and the vector of singular values of the reference frame. Distance is calculated for each frame and a new sequence *S* consisting of calculated distances is constructed. Then *S* is divided into sub-sequences and similarity between them is used to select the candidate sub-sequences. Deep scanning procedure is applied on the candidate sequences to decide the forged sequences. Randomly selected 500 blocks from the corresponding frames in the candidate sub-sequences are correlated and the ratio of the highly correlated blocks to the total number of blocks is used to decide similarity. Lin *et al.*'s method in 2013 used colour histogram difference of adjacent frames in the RGB colour space to determine temporal similarity [14]. A block-based algorithm is applied on the candidate sequences to determine forged sequences.

Techniques summarised above have the same algorithmic structure and so affected from the same problem: They cannot detect forged sub-sequences if the copied frame is mirrored before insertion. Wang *et al.*'s method cannot correlate mirrored frames with the original ones even if they have the same content [11]. Mirrored blocks cannot overlap in colour histogram-based method since the method eliminates candidate sequences with a block-based approach [14]. Also, SVD-based method cannot detect because it cannot extract similar distance values from mirrored frames [15]. Therefore, these methods cannot detect mirrored forged sequences.

Fig. 2 shows an example of FM. Figs. 2*a* and *b* show the original and forged videos, respectively. The video sequence given in Fig. 2*b* has mirrored frames denoted by M2, M3 and M4. Second, third and fourth frames in the original video are copied/mirrored and then inserted after sixth frame to create the forged video. Methods reported in the literature do not detect mirrored forged sequences because their features are not resistant to the mirroring attack.

We propose a method to detect FD even with mirroring attack in this work. It will be the first in the literature to detect mirrored forged sequence duplication forgery. We also aim to improve the

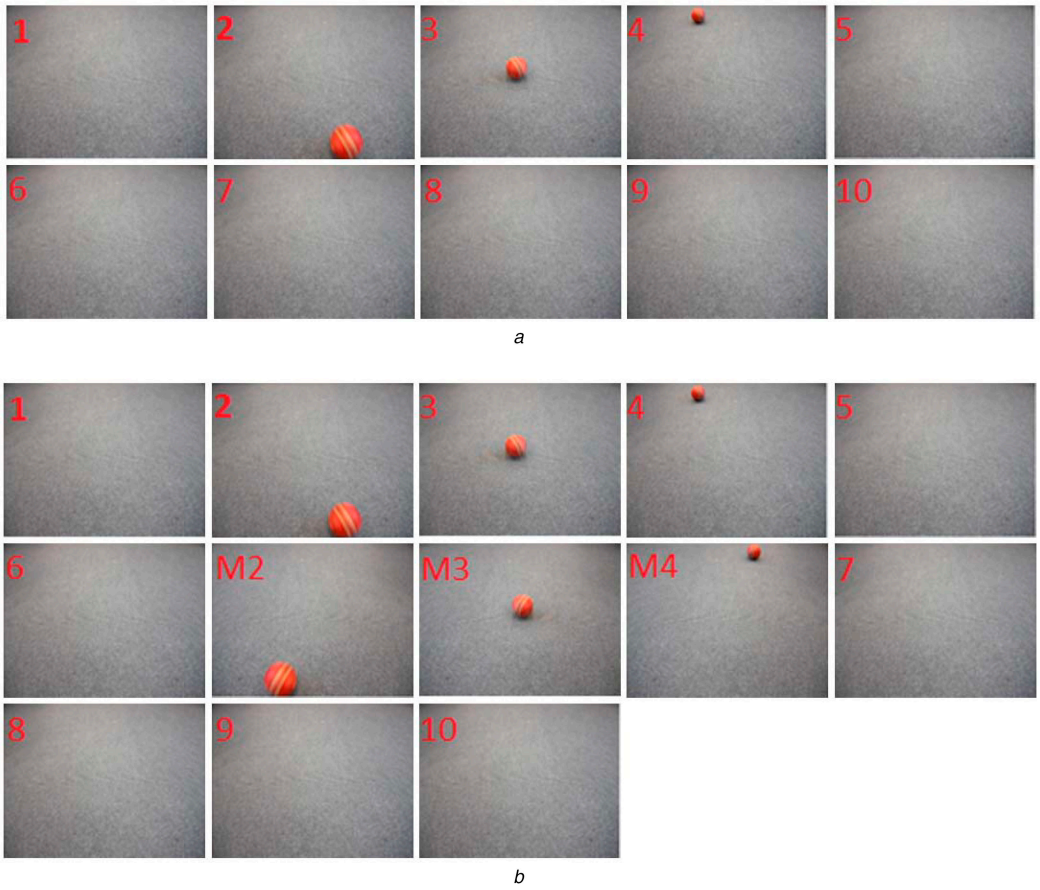

Fig. 2 ***(a)*** *Original video,* ***(b)*** *Forged video that has mirrored frames*

processing time and DA, PR and RR of similar works in the literature [11, 14, 15].

3 Proposed method

The details of the proposed method are given in this section. The method detects mirrored and/or duplicated forged sequences in four steps: feature extraction from frames, candidate frame selection using feature similarity, elimination of candidate frames with peak-to-signal noise ratio (PSNR) and finally post-processing to improve the detection.

The first step extracts features from binarised frames, which makes the method more robust against video compression artefacts. The malicious user modifies a video to hide or replicate a sub-sequence and then resave it with some codec properties. Thus, even if two frames are identical in the forged video, their pixel values will differentiate due to recompression. The feature extraction method, which extracts information from the frames, must be robust against compression artefacts. It must obtain similar features from the same frames in the forged video, after recompression. In this work, we proposed a binary feature extraction method to obtain robust features against the recompression, from the frames.

The second step uses the Euclidean distance between features to judge similarity among corresponding frames. The main motivation in this step is to determine the similar frames in a video sequence with coarser approach. If a sequence in the video file is copied and pasted into another location in the same video, corresponding frames in the copied and pasted sequences will be identical and thus their binary features will be similar. Euclidean distance is measured among each binary feature and the others to determine the similarity. If the binary feature is similar to another feature, corresponding frames are suspicious frames (the method investigates them in a finer manner at the following step) and their frame numbers are inserted into Candidate List. While the first column of this list accommodates source frame number, the second column contains the frame numbers of the candidate frames. Candidate list contains suspicious pairs for forgery operation.

The main motivation of the third step is to determine exact duplicated/mirrored frames with a finer approach. The previous step reports the frame pairs that give similar binary feature vectors. In this step, PSNR is calculated between similar frames that are in the candidate list to test the finer similarity. If two frames give high PSNR values, the algorithm decides that this pair is identical. Otherwise, the pair is removed from the candidate list to eliminate false candidates.

The performance of the detection algorithm is further enhanced by post-processing remaining candidates as the last step. In this step, the algorithm divides the candidate list into m groups according to the first column value to realise the post-processing operation. Each group represents a source frame and candidate frames similar to source frame, thus the number of different source frames determines the value of m. For example, if candidate list obtained from the previous step contains (2, 23) (2, 24) (2, 25) (3, 24) (3, 25) (3, 26), the m value will be 2. Candidate frames are 23, 24 and 25 for source frame 2 and candidate frames are 24, 25 and 26 for source frame 3. If a source frame is found to be similar more than one frame in the candidate list like the example, the algorithm evaluates PSNR between the source frame and candidate frames. The candidate frame that gives best PNSR with source frame is selected to be forged version of the source frame.

The general framework of the algorithm is given in Fig. 3 and the details of the steps are given below.

3.1 Step 1: Feature extraction from the frames

The method extracts frames from video sequence V and converts each frame into binary form in this step. Otsu's method is used by the method to choose an optimum threshold to minimise the variance of the black and white pixels. Let video sequence V contains N frames of size $H \times W$, $V = F^1 \cdots F^N$. Otsu's algorithm is applied on each frame to compute an optimum threshold and frames are converted into binary images, $B^1 \cdots B^N$.

The following feature extraction algorithm is applied on each binary frame to represent the video sequence by corresponding

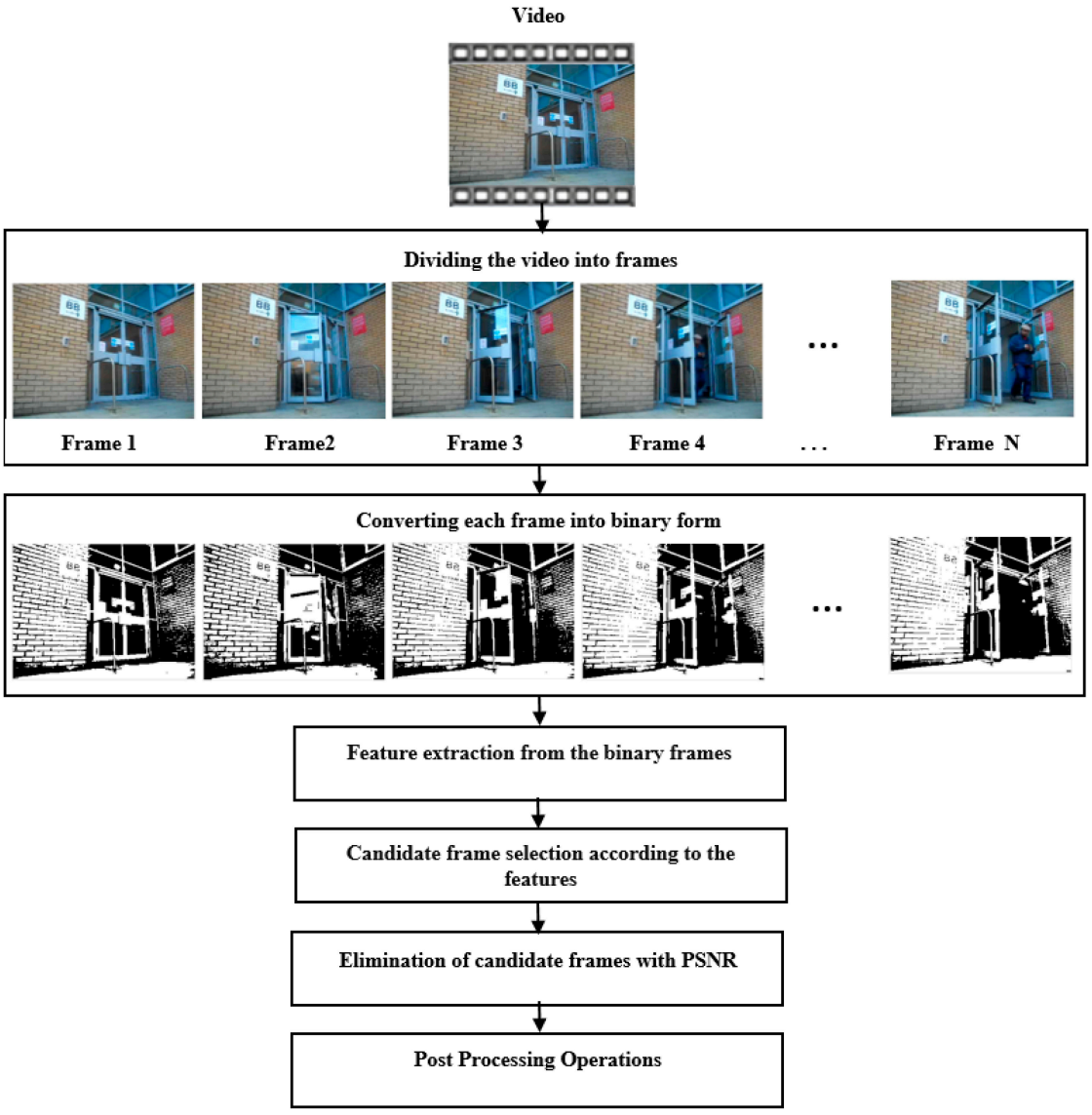

Fig. 3 *General framework of the algorithm*

feature vectors. The general framework of the feature extraction method is shown in Fig. 4*a*. Assume that current frame to be processed be B^i and rows of the current frame denoted by $B_1^i \cdots B_H^i$. Each row is represented by its Hamming distance with the following row. In other words, rows of the binary frame are XOR'ed with the following row and the number of 1's in the result is used to represent current row. Thus, feature extraction algorithm represents each row by a value in $[0 \cdots W]$ range.

Assume that feature vector is denoted by $\boldsymbol{X}^i = x_1^i \cdots x_H^i$. The equation given in (1) is used to calculate the corresponding feature vector elements for ith frame B^i. First element of feature vector will be zero all the time, $x_1^i = 0$.

$$x_j^i = \sum_{k=1}^{W} \left(B_{jk}^i \oplus B_{j-1,k}^i \right), \quad j \in \{2 \cdots H\}, \quad x_j^i \in \{0 \cdots W\} \tag{1}$$

Maximum value of the feature vector elements can be W, for the worst case, since two consecutive rows of totally different corresponding bit values results in all 1's. Minimum value of the feature vector elements can be 0, for the best case where the corresponding bits of two consecutive rows are all equal. The proposed feature extraction method is robust against mirroring attacks since the sum of XOR operations in (1) is not affected. Thus a feature vector produced from a frame is equal to the feature vector generated from the mirrored version of the same frame. Fig. 4*b* gives an example that demonstrates the mirror invariant feature of the method. Two vectors $\boldsymbol{B}_1^1, \boldsymbol{B}_2^1$ and their mirrored versions $m\boldsymbol{B}_1^1, m\boldsymbol{B}_2^1$ results in total difference of four as shown in the figure.

All binary frames $B^1 \cdots B^N$ are represented by their corresponding feature vectors $\boldsymbol{X}^1 \cdots \boldsymbol{X}^N$ at the end of this step.

3.2 Step 2: Candidate frame selection according to the features

Similar frames are determined using corresponding feature vectors in this step. Each feature vector is compared to the rest of the feature vectors to determine the list of candidates. The algorithm compares $\boldsymbol{X}^i$ with $X^{i+1} \ldots X^N$ by the Euclidean distance given in (2) to test the similarity between vectors. The equation given in (2) compares the ith feature vector that represents the ith frame with i + 1th frame.

$$\sqrt{\sum_{j=1}^{H} \left(x_j^i - x_j^{i+1} \right)^2} < t_d \tag{2}$$

where i denotes the source frame and j denotes the candidate frame. The algorithm selects pair $(i, i+1)$ and inserts it into the Candidate list C, if the distance between two vectors is less than a pre-determined threshold t_d. The proposed method evaluates the

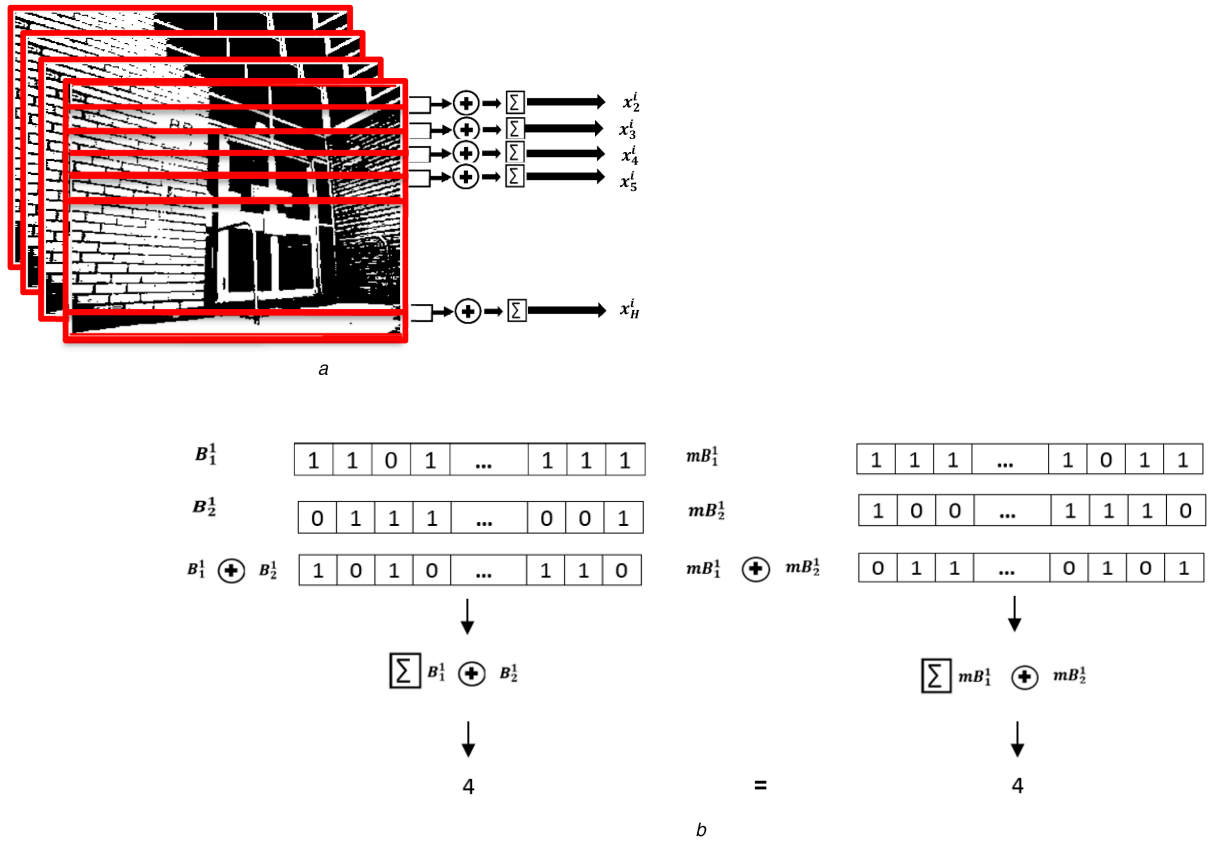

Fig. 4 ***(a)*** *General framework of the feature extraction method,* ***(b)*** *Proposed mirror invariant binary feature extraction technique*

worst (maximum) and the best (minimum) Euclidean distances between two vectors to determine an optimum threshold value t_d. The minimum distance between two vectors becomes 0 for the best scenario whereas two vectorsshould have opposite values for the worst case. The Euclidean distance between two vectors becomes maximum for the worst case if all the elements of one vector are maximum (can be W maximum) and all the elements of the other are minimum (can be zero minimum) $W\sqrt{(H-1)}$. It can be said that 1% of the feature vector elements are different if the distance between two vectors is approximately $0.01W\sqrt{(H-1)}$. The proposed method tests the values in $[0.01W\sqrt{(H-1)} - 0.012W\sqrt{(H-1)}]$ to determine an optimum threshold t_d. Corresponding experimental results for determination of t_d are given in Section 4.

PSNR between these frames and PSNR between the ith frame and mirrored version of the candidate frame is also calculated. Maximum PSNR is also inserted into the current row of C as the third element. The equation given in (3) shows the kth row of C assuming that the pair $(i, i+1)$ is chosen as a candidate. The function G in (3) converts a colour frame into grey level and function *mir* mirrors an image given as parameter

$$C^k = (i, i+1,\ \max(\text{PSNR}(G(F^i), G(F^{i+1})), \text{PSNR}(G(F^i), \text{mir}(G(F^{i+1})))) \tag{3}$$

Each feature vector is compared to the rest of the feature vectors to determine (source, candidate) pairs in this phase. Candidate list C is created at the end of this step where list items contain the index of similar frames and PSNR between them.

3.3 Step 3: Elimination of candidate frames with PSNR

Candidate list contains PSNR between the source and candidate frames at the third column. In this step, a histogram of the third column of C is created and the PSNR that corresponds to peak of the histogram is chosen to eliminate false candidates. Let t_{PSNR} be the PSNR corresponding to the peak value of the histogram. t_{PSNR} is determined to be the peak value for the current histogram of the third column of C. Peak value of the current histogram will determine the exact PSNR to be used as threshold by the algorithm because similar frames gives approximately equal PSNR and candidate list contains similar frames.

PSNR of the pairs in the list is compared with t_{PSNR}. Pairs in the current row are removed from the list if their PSNR is less than $\lfloor t_{\text{PSNR}} \rfloor$. The rule given in (4) is used to eliminate false candidates.

$$\forall i: \text{if } (C_3^i \le \lfloor t_{\text{PSNR}} \rfloor) \text{ then extract the current row } C^i \text{ from } C \tag{4}$$

The maximum of PSNR histogram is a clue about the compression artefacts. Candidate list contains candidate pairs and PSNR between them. Thus, it is expected that the list contains duplicated frames. PSNR between the candidate pairs should be infinite if the video does not have lossy compression artefacts. However, almost all forged videos are recompressed to reduce signs of forgery. Therefore, PSNR never reaches infinite even if the frames are the same. PSNR corresponding to the peak of the histogram indicates that, the PSNR between the duplicated frames should be at least t_{PSNR} due to the lossy compression.

Figs. 5*a–c* show determination of $\lfloor t_{\text{PSNR}} \rfloor$ values for some of the test videos. Three test videos from the dataset are used to show the PSNR threshold determination. The method applies Steps 1 and 2 on these test videos and obtains candidate lists. Histograms of the third columns of corresponding candidate lists for these test videos are shown in Figs. 5*a–c*. Peak values of the histograms determine the corresponding PSNR threshold $\lfloor t_{\text{PSNR}} \rfloor$ values for these test videos. Thus, the method determines t_{PSNR} for these test videos to be 34, 38 and 34, respectively. The method applies the procedure given in Fig. 5*d* to determine the t_{PSNR} value dynamically for the current test video.

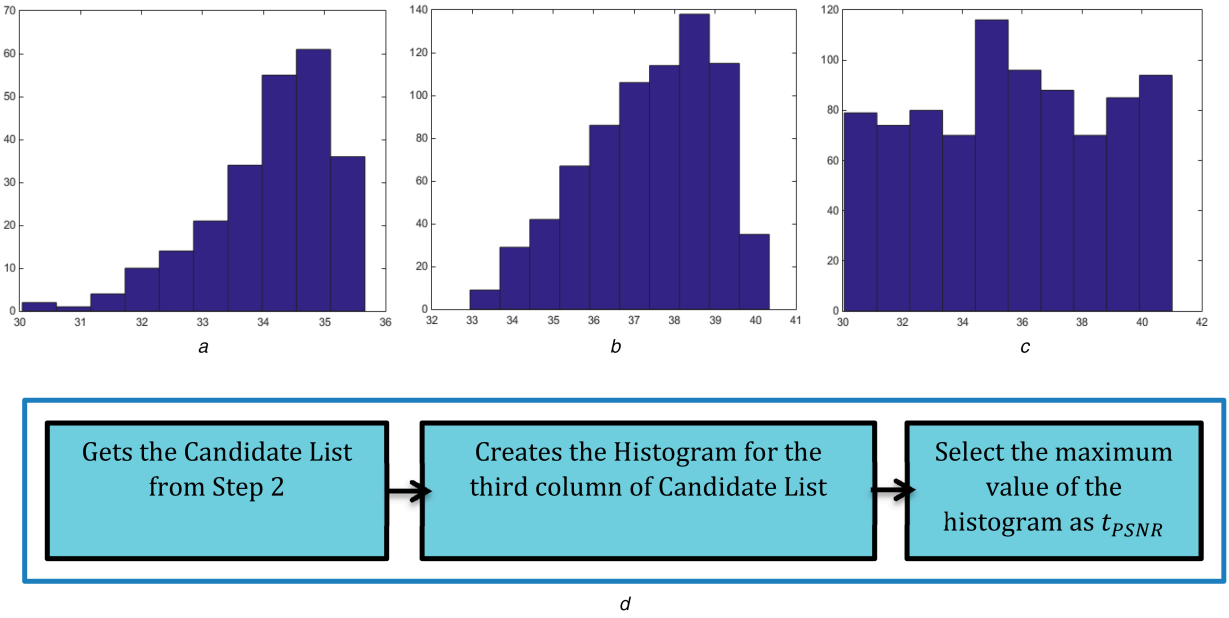

Fig. 5 ***(a)–(c)*** *$\lfloor t_{PSNR} \rfloor$ value determination for videos 3, 5 and 15, respectively,* ***(d)*** *the steps of the algorithm to determine $\lfloor t_{PSNR} \rfloor$ value*
(a) Video 3, $\lfloor t_{PSNR} \rfloor = 34$, ***(b)*** Video 5, $\lfloor t_{PSNR} \rfloor = 38$, ***(c)*** Video 15, $\lfloor t_{PSNR} \rfloor = 34$, ***(d)*** *Steps of the algorithm*

```
for i=1 to m
        for j=1 to k
                if (R(i, 3)>=dj-e) && (R(i, 3)<=dj+e)
                        R(i, 2)= R(i, 1) + dj;
                        break;
                end if
        end for
end for
```

Fig. 6 *Source code is applied on R to determine the forged sequences*

3.4 Step 4: Post-processing operations

Candidate list C now contains most probable candidates out of a larger list of candidates as explained in the previous section. Post-processing operations is applied to determine the exact location of the forged sequences in this section.

Candidate list C is divided into m groups, $G^1 \cdots G^m$ with equal first elements in the rows. Then, for each group the algorithm selects a row with the highest PSNR. Thus, best row from each group (with maximum PSNR) is selected and inserted into the result list R of size $m \times 3$. PSNRs are then removed from R leaving only indices of candidate frames and distance between them. The equation given in (5) is applied on each group. The function best_PSNR(G^i) gets the group index as the parameter and selects (returns as g) a row from the group that has the highest PSNR. Selected row g has three values, $g_1, g_2, |g_1 - g_2|$. The first two elements designate the indices of copied and pasted frames. Last element gives the distance between the frames.

$$\begin{aligned} &\forall i : i \in 1 \cdots m, \\ &g = \text{best\_PSNR}(G^i), \\ &R^i = [g_1 \quad g_2 \quad |g_1 - g_2|] \end{aligned} \tag{5}$$

The distances between consecutive source and duplicated frames should be equal if a video is forged by FD. In other words, the distance between source and destination indices stored in the third column in the result list must be approximately equal for frame-duplicated videos. R should contain two different set of distance values in the third column if there are two different forgeries on two different parts of the same video sequence. Assume that frames between (10, 20) and (30, 40) are copied on to frames at (110, 120) and (140, 150), respectively. In this case, R contains 100 and 110 distance values at the third column. Other distances are either artefacts caused by codec delay or can be false candidate pairs.

The algorithm (see Fig. 6) finds distance values that repeats itself more than ith at the third column R. Assume that k distance values are determined using this rule, $d_1 \cdots d_k$. The following source code is applied on R to determine the forged sequences. Three elements at the ith row of R are denoted by $R(i, 1)$, $R(i, 2)$ and $R(i, 3)$. e is the error tolerance of the method.

Fourth step uses PSNRs and distances to determine the exact location(s) of the forged sequences and the list R contains the indices of the copied and pasted frames in the video sequence.

4 Experimental results

The results of the experiments performed on a number of frames duplicated forged videos to find out forgery detection ratio of the proposed method are summarised in this section. The experiments are carried out on a notebook computer with 2.4 GHz dual core i7 processor running Matlab R2014b. Tests are performed on a set of forged videos created by Virtual Dub, an open-source video editor. Ten randomly picked videos from SULFA (Surrey University Library for Forensic Analysis) [1], are used to create forged samples. Videos, can_220_flap(1), can_220_flap(2), can_220_garden(1), can_220_hallway(2), can_220_road(1), fuji_2800_outdoor(4), fuji_2800_street(1), nik_s3000_bridge(1), nik_s3000_bridge and nik_s3000_indoor_stairs at 320×240 resolution are used to create the test video database. A total of 30 forged test videos, 10 frames duplicated, 10 frames mirrored and 10 both mirrored and duplicated are created and used during the tests.

Table 1 lists the details of the forgeries on the videos. For example, the frames between 100 and 160 in test video 1 are inserted after the 315th frame to create the forged video. The frames between 293 and 323 in test video 10 are mirrored and inserted after the 10th frame. The test videos after video 20 contain two types of attacks: Both FD and FM.

Tampered video lengths are greater than the originals because FD attack increases the length of the video by the number of duplicated frames. Randomly chosen frames will be randomly replaced/inserted into the test video to create the forged videos. Test videos given in Table 1 are encoded with open-source MPEG-4 (Level 5 Advanced Simple Profile, ASP@L5) algorithm of Xvid codec after the forgery with an open-source video editing tool, Virtual Dub.

The proposed method uses mirror invariant feature extraction technique to detect FD and FM forgery. Computation time, PR, RR

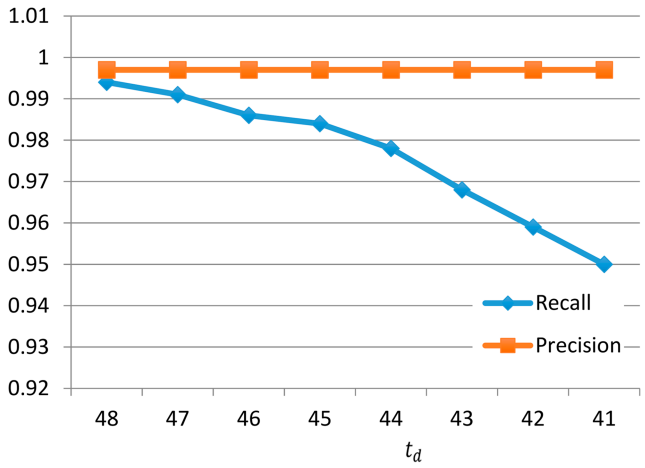

Fig. 7 *Effect of t_d on precision and recall curve*

and DA are evaluated to compare the method with similar detection methods reported in the literature. The definitions for PR, RR and DA are given in (6) where TP, TN, FP and FN represent 'authentic is detected as authentic', 'forged is detected as forged', 'authentic is detected as forged' and 'forged is detected as authentic', respectively,

$$\text{PR} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{RR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{DA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6}$$

A method's performance is satisfactory if it has higher precision and recall values. While the total number of detections is represented by (TP + TN), (TP + TN + FP + FN) is the total number of frames in the experiment. Therefore, higher DA means that the method has better detection rate.

The algorithm uses $t_d = 48$, th = 10, $e = 2$ as predefined parameter values during the tests. We select the value of t_d to be 48 during the experiments. The method is tested on the videos given in Table 1 for various t_d values in [41–48] range and precision, recall curves are created as can be seen in Fig. 7. The curves show that when threshold value is selected to be 48, *FP* and *FN* get the best value.

The first experiment shows the execution time of the proposed method for each test video given in Table 1. All 30 forged videos are used for this test and total execution time and per frame execution time is measured to show the method's performance. Table 2 shows the total and per frame execution time for each test video. Maximum per frame execution time in the tests is measured as 0.1 s, as shown in Table 2. Total execution for the method increases as the length of the video increased. Best per frame execution time for the method is 0.003 s.

We also test our work according to the parameters given in (6): *PR*, *RR* and *DA*. Figs. 8*a–c* show the *PR*, *RR* and *DA* of the proposed method for the three sets of forged set of test videos. The first set of forged videos contains first ten videos in Table 1 and the results are given in Fig. 8*a*. Figs. 8*b* and *c* also show the results of the proposed method for the test videos with only FM attack (test videos between 11 and 20) and with both FD and FM attacks (test videos between 21 and 30), respectively. Results show that the proposed method can detect forged sequences with high PR, RR and DA in the videos even if FD and FM attacks are applied together, as shown in Fig. 8*c*.

Table 1 Details of test videos

Test video	Original length	Tampered length	Frame rate, fps	Attack	Forgery operation
video 1	369	429	29.97	FD	100–160 are copied behind 315
video 2	332	372	29.97	FD	60–100 are copied behind 280
video 3	353	403	29.97	FD	200–250 are copied behind 10
video 4	122	142	29.97	FD	40–60 are copied behind 80
video 5	301	371	29.97	FD	150–220 are copied behind 240
video 6	270	310	30	FD	30–70 are copied behind 130
video 7	330	390	30	FD	110–170 are copied behind 200
video 8	188	228	30	FD	10–50 are copied behind 105
video 9	344	394	30	FD	120–170 are copied behind 220
video 10	293	323	30	FD	250–280 are copied behind 10
video 11	369	398	29.97	FM	60–89 are copied behind 1
video 12	332	371	29.97	FM	60–99 are copied behind 280
video 13	353	403	29.97	FM	5–55 are copied behind 75
video 14	122	142	29.97	FM	40–60 are copied behind 80
video 15	301	361	29.97	FM	5–65 are copied behind 105
video 16	270	320	30	FM	100–150 are copied behind 200
video 17	330	400	30	FM	2–72 are copied behind 102
video 18	188	218	30	FM	20–50 are copied behind 70
video 19	293	353	30	FM	90–150 are copied behind 180
video 20	344	403	30	FM	1–60 are copied behind 81
video 21	369	438	29.97	FD + FM	60–89 are copied behind 1 and 200–240 are copied behind 280
video 22	332	391	29.97	FD + FM	60–99 are copied behind 280 and 10–30 are copied behind 325
video 23	353	433	29.97	FD + FM	5–55 are copied behind 75 and 160–190 are copied behind 230
video 24	122	162	29.97	FD + FM	40–60 are copied behind 80 and 10–30 are copied behind 100
video 25	301	386	29.97	FD + FM	5–65 are copied behind 105 and 225–250 are copied behind 270
video 26	270	350	30	FD + FM	100–150 are copied behind 200 and 10–40 are copied behind 60
video 27	330	430	30	FD + FM	2–72 are copied behind 102 and 235–265 are copied behind 290
video 28	188	238	30	FD + FM	20–50 are copied behind 70 and 125–145 are copied behind 165
video 29	293	378	30	FD + FM	90–150 are copied behind 180 and 230–255 are copied behind 270
video 30	344	438	30	FD + FM	1–60 are copied behind 81 and 180–215 are copied behind 250

FD: Frame duplication, FM: Frame mirroring.

The proposed method's *PR*, *RR* and *DA* are compared with other methods in the literature in another experiment. Results of this experiment are given in Figs. 8*d–f*. Used test videos for this experiment do not contain FM forgery because the methods in the literature do not detect FM attack [11, 14, 15]. Fig. 8*e* indicates that *RR* of the method is approximately 9% higher than that of [15]. *RRs* for others are approximately 88 and 73%, respectively [11, 14]. *DA* performance of the method is also better than of [15] as shown in Fig. 8*f*. There is 8% difference between the *DA* of the proposed method and that of [15]. Fig. 8*d* shows that *PR* for the method is almost the same with the method given in [15].

Another experiment is done to show the effect of the double compression on the result of the proposed method. Forged video dataset given in Table 1 is encoded with different quality factors (QFs) after forgery. Quality factors 1 (almost lossless), 5, 10, 15 and 20 (compressed considerably) are used to encode the forged videos with MPEG-4 encoder. Fig. 9 indicates DA, PR and RR of the proposed method for the test videos with different quality factors. DA of the proposed method is approximately 99.3% for QF = 1 and decreases to 86.97% for QF = 20. PR is 99.98% for QF = 1and decreases to 97.76% for QF = 20. RR is 99.3% for QF = 1and decreases to 86.69% for QF = 20. This experiment indicates that DA decreases for increased compression, whereas PR (and RR) is/(are) not affected very much.

The last experiment compares robustness against various attack types, execution time, DA, PR and RR of the proposed method with others. Table 3 shows that the proposed method could determine forgery faster than other methods. While the proposed method can process a frame in 0.01 s, execution time for one frame for the other methods become approximately 294, 140 and 0.58 s, respectively [11, 14, 15]. Simple arithmetic of the proposed method makes it faster compared to others. At the same time, the proposed method can detect forged videos with better DA, PR and RR. The proposed method yields 0.99% DA, whereas the others yield 0.59, 0.87 and 0.91%, respectively. RR of the method is also higher than the others, but PR is approximately equal with [15]. Table 3 also shows that the proposed method has better detection performance.

Table 2 Execution time for each test video

Video	Length	Execution time	
		Total time, s	Time, s/frame
video 1	429	8.19	0.01
video 2	372	1.73	0.004
video 3	403	2.09	0.005
video 4	142	1.6	0.01
video 5	371	10.07	0.03
video 6	310	1.14	0.003
video 7	390	10.24	0.03
video 8	228	1.21	0.004
video 9	394	1.23	0.003
video 10	323	1.77	0.004
video 11	398	2.28	0.005
video 12	371	1.85	0.004
video 13	403	2.2	0.005
video 14	142	0.59	0.004
video 15	361	3.59	0.009
video 16	320	10.6	0.03
video 17	400	28.1	0.07
video 18	218	0.76	0.003
video 19	353	1.29	0.003
video 20	403	1.45	0.003
video 21	438	4.68	0.01
video 22	391	2.43	0.006
video 23	433	3.07	0.007
video 24	162	0.97	0.005
video 25	386	6.78	0.02
video 26	350	14.07	0.04
video 27	430	40.6	0.1
video 28	238	1.29	0.005
video 29	378	1.14	0.003
video 30	438	1.71	0.003

The method can detect FM forgery as well as other types of forgeries such as frame insertion (FI), FD and composite of them. FI type of forgery is formed when one frame from the video is copied and inserted into another location in the same video. On the other hand, copied frame is pasted on to another frame in the same video to create an FD type of forgery. Composite attacks such as FM + FD, FM + FI will be formed, when a forged group consists of various types of forged frames. For example, one half of the group consists of mirrored frames while the remaining part contains duplicated ones (for the FM + FD case). The proposed method also detects forged group in the video containing combined attacks (such as FM + FD, FM + FI) while the other methods cannot detect those as indicated in Table 3.

Table 3 emphasises that the other methods in the literature cannot detect FM forgery. Even though mirror invariant capability can be added to methods in [11, 14, 15] with some modifications to make them detect FM attack, extra computational cost will be added. The works in [11, 15] necessitate extra $\left(\sum_{k=1}^{G}\binom{G}{k}\right)^2 \times ((N-G)(N-G+1)/2)$ computations for feature vector comparisons, where G and N denote the group size and the number of frames, respectively. The number of extra comparisons for the method proposed in [14] becomes $((N-G)(N-G+1)/2)$. The methods in [11, 15] extract features from overlapping group of frames in the test video. However, their feature extraction techniques are not mirror invariant. So, they require evaluating various combinations for the current group. Assume that the current group consists of G frames. $\binom{G}{k}, k = 1 \cdots G$ different combinations for this group can be formed during the forgery process. The term $\binom{G}{1}$ corresponds to the number of groups with one mirrored frame, whereas $\binom{G}{G}$ corresponds to the number of groups with all frames mirrored. Thus, the methods in [11, 15] must consider all these combinations during determination of the candidate subgroups because their feature extraction technique is group based. The method in [14] uses mirror invariant group-based feature extraction technique on the test video to determine candidate subgroups. However, spatial similarity comparison in their work requires extra comparisons to detect FM forgery. The extra effort to make these methods [11, 14, 15] mirror invariant will make them even slower since per frame execution time of these methods are worse than the proposed method.

Experiments in this section emphasise that the proposed method detects FD, FM forgery very well. Similar works in the literature do not detect FM forgery. The proposed method also has better PR, RR and DA compared to others and also is faster than the others according to the per frame execution times.

5 Conclusion

The rapid development and wide availability of both open source and commercial video editing software has enabled video modifications by anyone. FD forgery is one of the easiest ways to manipulate videos to either cover up or repeat an action in the scene. Detection of FD forgery attracts considerable attention of the researchers and many methods are proposed recently. On the other hand, FM is another possible type of forgery in video forensics. Methods to detect FD type forgery reported in the literature do not detect mirrored forged sequences because of the features they use. We proposed a method that detects FD forgery even if the frames are mirrored in this work. A new mirror invariant binary feature extraction technique is proposed to make the method robust against FM operations. Both PSNR and distances between the candidate frames are used to eliminate false candidates and improve detection. Per frame execution time is measured to test the time efficiency of the method. Results indicate

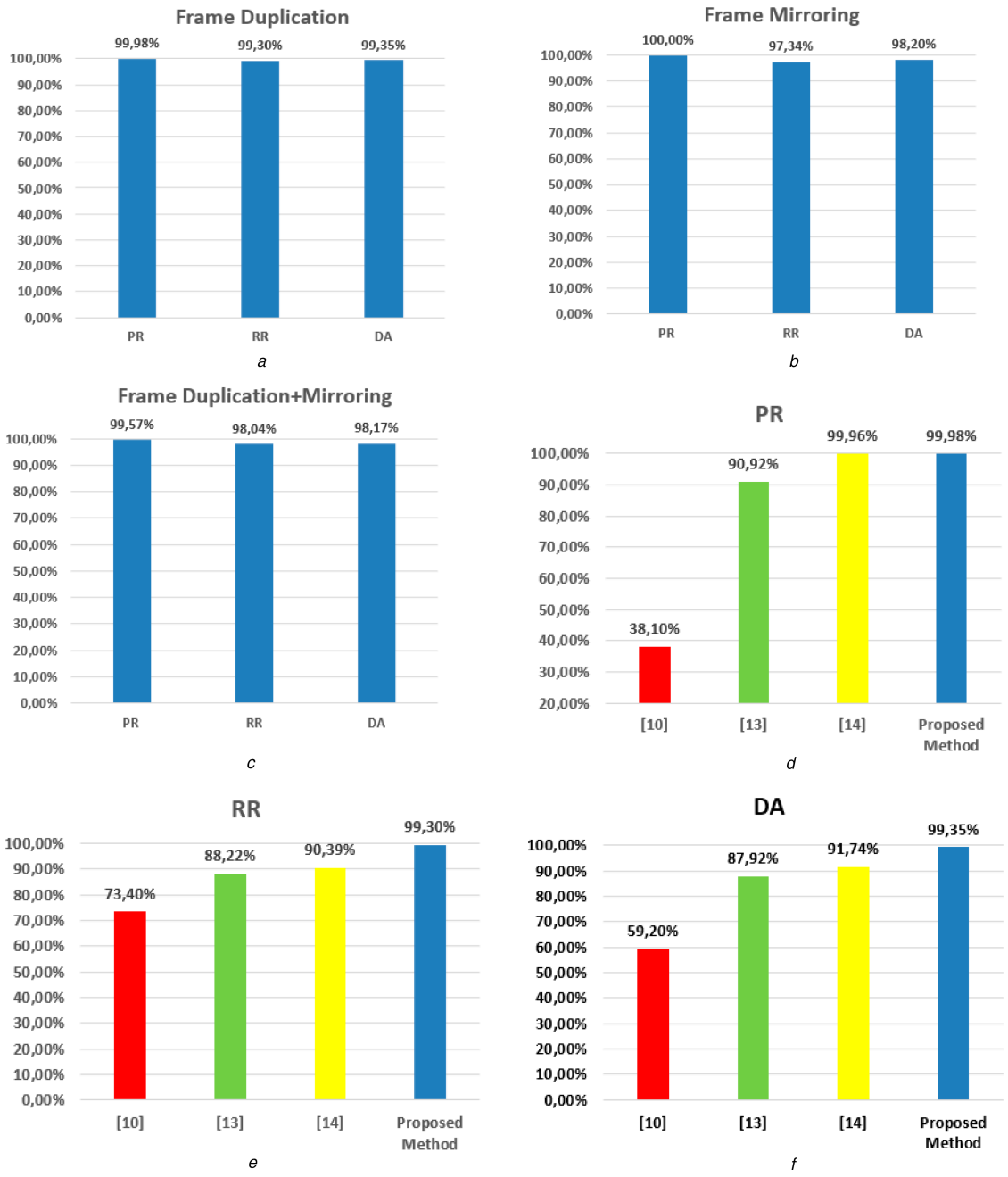

Fig. 8 ***(a)–(c)*** *Performance evaluation of the method for FD, FM and FD + FM attacks,* ***(d)–(f)*** *Comparison of the proposed method with other algorithms according to PR, RR and DA*

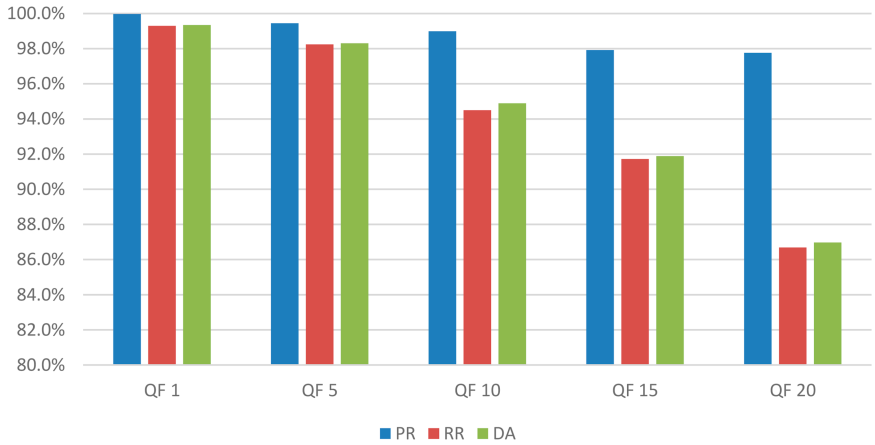

Fig. 9 *Detection results of the proposed method when recompression is applied after forgery operation*

that the proposed method is faster than the other methods [11, 14, 15] reported in the literature. *PR*, *RR* and *DA* are also measured to

Table 3 Comparison of the proposed method with others

Method	FM[a]	FD[a]	FI[a]	FM + FD	FM + FI	Execution time, s/frame	DA	PR	RR
Wang and Farid [11]	×	√	√	×	×	294.67	0.59	0.38	0.73
Lin and Chang [14]	×	√	√	×	×	140.6	0.87	0.9	0.88
Yang *et al.* [15]	×	√	√	×	×	0.58	0.91	0.99	0.90
proposed method	√	√	√	√	√	0.01	0.99	0.99	0.99

[a]FM: Frame mirroring, FD: Frame duplication, FI: Frame insertion.

quantitatively evaluate the detection capability of the method. The method yields better PR, RR and DR compared to others [11, 14, 15].

6 Acknowledgment

This work was supported by Tubitak with project number 115E214.

7 References

[1] Qadir, G., Yahahya, S., Ho, A.T.S.: 'Surrey university library for forensic analysis (SULFA)'. Proc. of the IET IPR 2012, London, 3–4 July 2012

[2] Hsu, C., Hung, T., Lin, C.: 'Video forgery detection using correlation of noise residue'. Proc.10th Workshop on IEEE Multimedia Signal Processing, 2008, pp. 170–174

[3] Kobayashi, M., Okabe, T., Sato, Y.: 'Detecting video forgeries based on noise characteristics', *Lect. Notes Comput. Sci., Adv. Image Video Technol.*, 2009, **5414**, pp. 306–317

[4] Kobayashi, M., Okabe, T., Sato, Y.: 'Detecting forgery from static-scene video based on inconsistency in noise level functions', *IEEE Trans. Inf. Forensics Sec.*, 2010, **5**, (4), pp. 883–892

[5] Wang, W., Farid, H.: 'Exposing digital forgeries in interlaced and deinterlaced video', *IEEE Trans. Inf. Forensics Sec.*, 2007, **2**, (3), pp. 438–449

[6] Wang, W., Farid, H.: 'Exposing digital forgeries in video by detecting double MPEG compression'. Proc. of the 8th workshop on Multimedia and security, 2006, pp. 37–47

[7] Luo, W.-Q., Qu, Z.-H., Huang, J.-W., *et al.*: 'A novel method for detecting cropped and recompressed image block'. IEEE Conf. on Acoustics, Speech, and Signal Processing, April 2007, pp. 217–220

[8] Luo, W.Q., Wu, M., Huang, J.W.: 'Mpeg recompression detection based on block artifacts'. Proc. of SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents, 2008, vol. **6819**, p. 68190X

[9] Wang, W., Farid, H.: 'Exposing digital forgeries in video by detecting double quantization'. Proc.11th ACM Workshop on Multimedia and Security, 2009, pp. 39–48

[10] Subramanyam, A.V., Emmanuel, S.: 'Video forgery detection using HOG features and compression properties'. IEEE 14th Int. Workshop on Multimedia Signal Processing (MMSP), 2012, pp. 89–94, doi: 10.1109/MMSP.2012.6343421

[11] Wang, W., Farid, H.: 'Exposing digital forgeries in video by detecting duplication'. Proc. of the 9th Workshop on Multimedia & Security, 2007, pp. 35–42

[12] Lin, C.S., Tsay, J.J.: 'A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis', *Digit. Invest.*, 2014, **11**, pp. 120–140

[13] Chao, J., Jiang, X.H., Sun, T.F.: 'A novel video inter-frame forgery model detection scheme based on optical flow consistency'. Digital Forensics and Watermarking, Berlin, Heidelberg, 2013, pp. 267–281

[14] Lin, G.S., Chang, J.F.: 'Detection of frame duplication forgery in videos based on spatial and temporal analysis', *Int. J. Pattern Recognit. Artif. Intell.*, 2012, **26**, (7), pp. 1–18

[15] Yang, J.M., Huang, T.Q., Su, L.C.: 'Using similarity analysis to detect frame duplication forgery in videos', *Multimedia Tools Appl.*, 2014, pp. 1–19