Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 4558 – 4574

doi:10.3906/elk-1703-125

A new video forgery detection approach based on forgery line

Işılay BOZKURT[1,*], Mustafa Hakan BOZKURT[1], Güzin ULUTAŞ[2]
[1]Department of Software Engineering, Of Faculty of Technology, Karadeniz Technical University, Trabzon, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Karadeniz Technical University, Trabzon, Turkey

Received: 13.03.2017 • **Accepted/Published Online:** 05.09.2017 • **Final Version:** 03.12.2017

Abstract: In recent years, we have continually encountered multimedia files in daily life, as evidence in courts, medical records in hospitals, etc. Many multimedia-editing software tools have been developed in parallel to the widespread usage of multimedia files. Thus, forgery operations can be committed by anyone with this software, even if s/he does not have any skill in this field. Authentication of the originality of multimedia files has recently become a popular topic. In this work, we propose a new video forgery detection approach to detect forged frames with better execution and detection capability. Features are extracted from the frames and their correlations are represented as a correlation image. This method investigates a line on the correlation image to determine the forgery operation. Then two new procedures (shrinking/expanding) are applied to the detected line to determine the exact location of the forgery. The experimental results indicate that the proposed method can detect forgery operations with better execution time and detection capability when compared to similar works in the literature.

Key words: Video forensic, video forgery detection, copy–move forgery, video processing

1. Introduction

The development of many easy-to-use multimedia editing tools renders the authenticity of the multimedia files susceptible. Ensuring their authenticity is important, because they are used in all areas of life such as in medical systems, military communications, or as evidence in court. Thus, developing new techniques to ensure the originality of the multimedia files has become an active research topic because of their susceptibility.

In recent years, many techniques have been proposed to authenticate image files, and their number is increasing every year. On the other hand, even if the video forgery detection problem has drawn the attention of researchers, only a limited amount of work has been realized in this area compared to the image forgery detection problem.

The techniques that determine the originality of the video files can be categorized into two types: active and passive methods. Active methods, known as "video watermarking techniques", embed specially created information into the video to authenticate it later. Therefore, they necessitate specially equipped hardware to embed information during the capturing operation, or specially written software after the capturing process. However, passive methods only use statistical information about the video file to determine the forgery process. These draw more attention from researchers due to their easy applicability.

Forgery operations on the videos can be grouped into two categories: frame-based forgery and region-based forgery. In the first category, the malicious user can modify a video by using several of the following

*Correspondence: isilaybozkurt@ktu.edu.tr

operations: insertion of frames from another video, deletion of frames, replacement of frames from another video, replacement of frames from the same video, or insertion of frames from the same video. The second category contains forgery operations, which copy a region on the frame and paste it onto another region on the same frame. The methods proposed for "copy–move forgery detection on images" can be used to detect the second type of forgery operation in the videos. However, detection of forgery operations in the first category is harder and necessitates added effort from researchers. In this work, we focus on two forgery operations: replacement of frames and insertion of frames from the same video, which we call 'frame duplication', as in the literature.

Frame duplication modification can be used to remove scenes from the video or replicate them. Two examples of these scenarios are given below.

- A person can read a secret document in a room that is monitored by cameras. Camera records can be viewed by a particular person at a specific time each day. A malicious person can modify records, indicating that s/he has not read any documents in that room, by replacing the frames that accommodate him/her with ones that do not.
- A gun is actually fired one time in the original record. A malicious user can modify it so that the gun is fired twice.

Figure 1 contains an example of frame duplication forgery. Figure 1a contains original frame sequences, whereas Figure 1b indicates the forged version of this sequence. The man in the forged video sequence leaves the elevator twice.

Wang et al. proposed the first method in the literature, which uses temporal correlation matrices to detect frame duplication forgery [1]. Their work divides the entire video into overlapping subsequences and calculates correlation matrices from the subsequences. If any two matrices correlate, the method assumes that corresponding subsequences are a candidate for frame duplication. In the second phase of the algorithm, spatial correlation matrices are created among these candidate subsequences to decide the exact forgery. Following this work, many techniques have been proposed to detect frame duplication forgery, and their details are given in Section 2.

The methods in the literature aim to improve two factors: detection accuracy and runtime performance. If a method attempts to improve detection accuracy, it is possible to be affected by time complexity. In this work, we aim to improve detection accuracy with better runtime. The experimental data set is created from a set of compressed videos to render the method more realistic. The proposed method creates a correlation image from the feature vectors, which represent corresponding frames. The correlation image accommodates lines with a slope of -45°, referred to as forgery lines, if any forgery operation has been applied on the current test video. The method utilizes a line detection algorithm to determine the location of the forgery line. Then shrinking/expanding algorithms are applied to the forgery line to determine the exact location of the forgery (integer valued points on the line correspond to the indexes of the copied and pasted frames) on the test video. The outline of the algorithm can be summarized as follows.

The proposed method divides the test video into frames, and partitions each frame into nonoverlapping 12×12 sub-blocks. It applies discrete cosine transform (DCT) to each sub-block at each frame and transforms them into the frequency domain. Average DCT value for each sub-block is calculated, and a row vector is obtained from each frame that contains averaged DCT values. The obtained row vectors for each frame are then binarized. The proposed method calculates a correlation matrix from binary row vectors and creates a correlation image for the current test video. Brighter pixels in the correlation image denote similar frames. If

a)

b)

Figure 1. Frame duplication examples: a) original video image, b) forgery video images.

some frames in the video are copied and pasted into another point, the correlation image will accommodate a line with slope of -45°, called forgery line. The proposed method applies the Hough transform to the correlation image to find such a line, and then applies exact localization procedure to it to detect the correct location of the forgery operation. Exact localization procedure investigates the neighboring lines and then decides the exact location of the forgery line. Furthermore, the method applies the shrinking/expanding approach on the forgery line to eliminate false matches. Experimental results show that the proposed method gives better detection accuracy with improved run time when compared to similar works in the literature [1–4].

Although the average detection accuracy for the method is approximately 0.99 s, the others show approximately 0.59, 0.79, 0.85, and 0.87 s, respectively [1–4]. The method processes a frame in 0.07 s when average run time is considered. The run time durations for other methods are approximately 294.67, 140.6, 0.41, and 0.063 s, respectively.

The paper is organized as follows. Section 2 introduces related works for frame duplication detection. The outline of the proposed method is given in Section 3, and Section 4 contains experimental results. Conclusions are given in the last section.

2. Related work

The works in the literature can be grouped into three categories according to their goals [5]. The methods in the first category aim to detect forgery operations by using traces of double or multiple compression, because the malicious user must resave the forged video after the tampering operation.

Wang and Farid proposed the first work in this category in 2006 [6]. Their work showed that a forgery operation leaves some artifacts on static and temporal statistical perturbations. In 2009, the authors used the double quantization effect, which is the result of double MPEG compression, to detect forgery operations [7]. Their technique can detect tampered regions even if their size is as small as 16 $\times$ 16 pixels. Sun et al. indicated that double compression explains the violation on the parametric law for first-digit distribution of alternating current (AC) coefficients [8]. Their method extracts a feature vector with a size of 1 $\times$ 12 from each group of pictures (GOP), and the original bit rate scale is estimated by using a support vector machine (SVM). Vazquez-Padin et al. introduced a new footprint to detect forgery operation [9]. Their technique used a variation of the macroblock prediction types in the re-encoded P frames. However, their algorithm only works with a fixed-size GOP. Jiang et al. modeled first-order Markov statistics on the DCT coefficients [10]. Their results showed better results compared to other works, even if the first quality scale is a divisor of a second quality scale. Huang et al. emphasized that the number of different coefficients between the I-frames of singly and doubly compressed MPEG-2 videos is larger than doubly and triply compressed MPEG-2 videos [11].

The methods in the second category deal with "region-based forgery", which copies a region of the frame and pastes it onto another region on the same frame or other ones, to hide or replicate a region. In 2007, Wang and Farid showed that spatial and temporal correlations of the deinterlacing algorithms could be disturbed due to the tampering operation [12]. Their results showed that the method can localize tampering in a spatial and temporal domain. Li et al. extracted motion information from the frames to apply the block-based motion estimation method [13]. Orientation and gradient information of the motion vectors was used to determine the forged regions. Hsu et al. modeled the distribution of correlation of temporal noise residue as a Gaussian mixture model [14]. The block-level correlation of noise residue was used to localize the regions tampered with by the algorithm. In 2015, Su et al. proposed a method based on compressive sensing to detect a specific forgery type: moving foreground was removed from the background [15]. Their method utilized k-singular value decomposition (k-SVD) to obtain the features of the difference between the frames. The algorithm that projects the features onto a smaller subspace uses k-means as well. However, the method only works with static background videos. Bidokhti et al. proposed a method that divides each frame in a video sequence into suspicious and remaining areas [16]. An optical flow coefficient is calculated for each part, and abnormal condition on the optical flow coefficient designates the forgery operation.

The methods in the third category are used to detect "frame-based forgery". In 2011, Lin et al. divided the video into subsequences and calculated color histogram difference over them [2]. The correlation of histograms

for any subsequences indicates the forgery operation in a coarser manner. Additionally, their method employs the block-based approach on the candidate subsequences to determine the exact location of the forgery. In [17], the authors showed that optical flow in the X and Y direction of adjacent frames is consistent for nontampered videos. Frame-based forgery will destroy the optical flow, as can be seen in their results. This method divides the video into subsequences and calculates the optical flow in each subsequence. Optical flow is thresholded, and binary search is applied according to the result of the thresholding operation to determine the forgery point. However, their method gives better results when the inserted frame has a different background scene, resulting from obvious forgery operation. The method in [18] processes the video frame-by-frame and divides each frame into four sub-blocks. Nine features are used to represent each frame and lexicographical sort is applied to group the similar frames. Root mean square error (RMSE) is calculated for neighboring frames' feature vectors to determine the forgery. However, compression decreases the performance of their method, because it works with correlation. In [3], Yang et al. used singular value decomposition (SVD) to extract features from the frames. Then the Euclidean distance between each frame and the reference frame, which is the first frame of the video, is calculated to determine temporal similarity. Additionally, the authors investigate similar clips with random block matching.

3. Proposed method

In this section, we give a detailed outline of the proposed method. The main challenges considered in the literature are the running time of the detection process and detection accuracy of the method. We aim to propose a new approach to enhance these two factors in the frame duplication detection problem. Our work proposes three new algorithms, namely "correlation image generation", "coarser forgery line detection", and "finer forgery line localization", to ensure better detection performance and lower execution time. The details of these algorithms are given in the following subsections.

A general outline of the proposed method is presented in Figure 2.

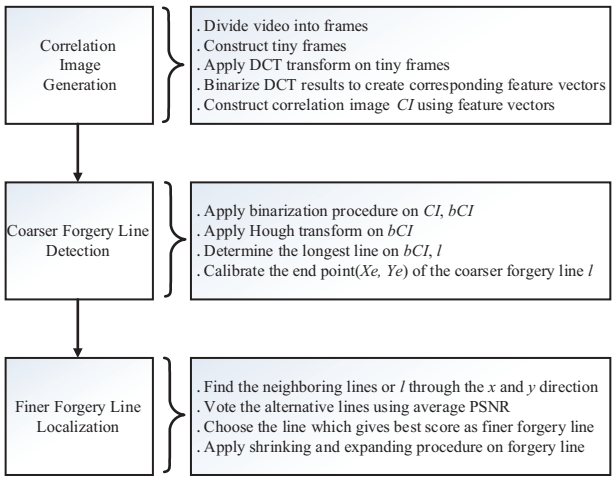


Figure 2. General outline of the proposed method.

3.1. Correlation image generation

The main goal of this function is to construct a correlation image that represents the similarity between the frames using brightness values. The function divides the video V into frames $F_1, F_2, \ldots, F_k$ and converts them

into gray scale. Each frame $F_1, F_2, \cdots, F_k$ is divided into nonoverlapping 12×12 cells. The size of each cell becomes $\lfloor N/12 \rfloor \times \lfloor M/12 \rfloor$ for a frame of size $N \times M$.

Assume that 144 cells for the ith frame F_i are denoted by $C_1, C_2, \cdots, C_{144}$, where C_1 and C_{144} represent the upper left and lower right cells, respectively. The function calculates average intensity values in the cells and constructs ith tiny frame TF_i of size 12×12 from the cells. Discrete cosine transform (DCT) is applied to the tiny frame. Then DCT coefficients are binarized according to the phase of the DCT transform. Binary value 0 is used to represent the DCT coefficients, which have negative values, whereas Binary 1 is used to denote the other coefficients. After the binarization procedure, the zigzag-scanning procedure is applied to the binarized 12×12 cells to construct binary feature vector V_i of size 1×144. Figure 3 shows the tiny frame generation and binary feature vector construction phase of the proposed method.

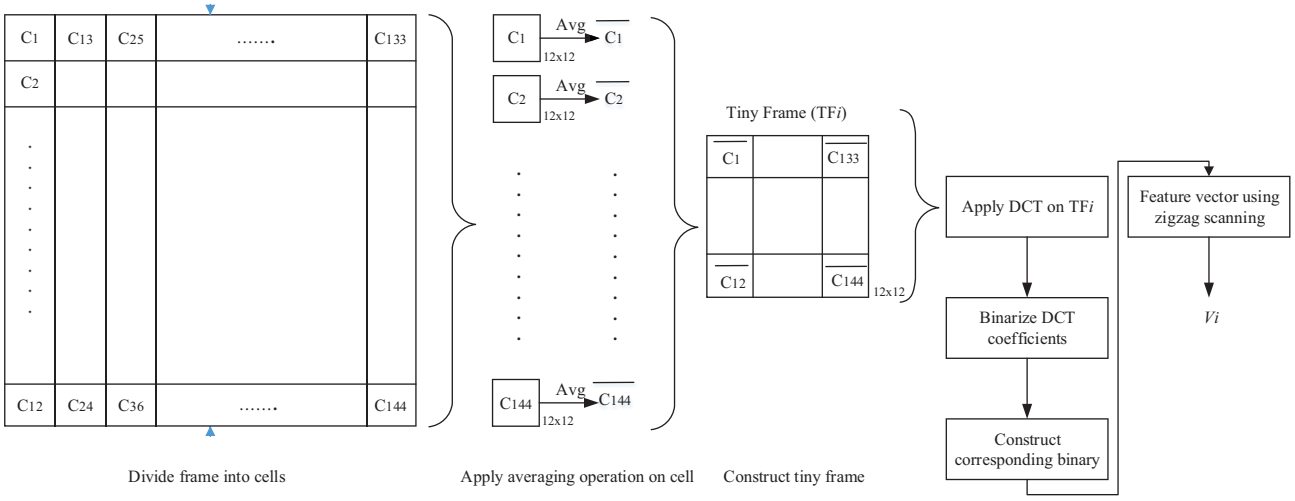
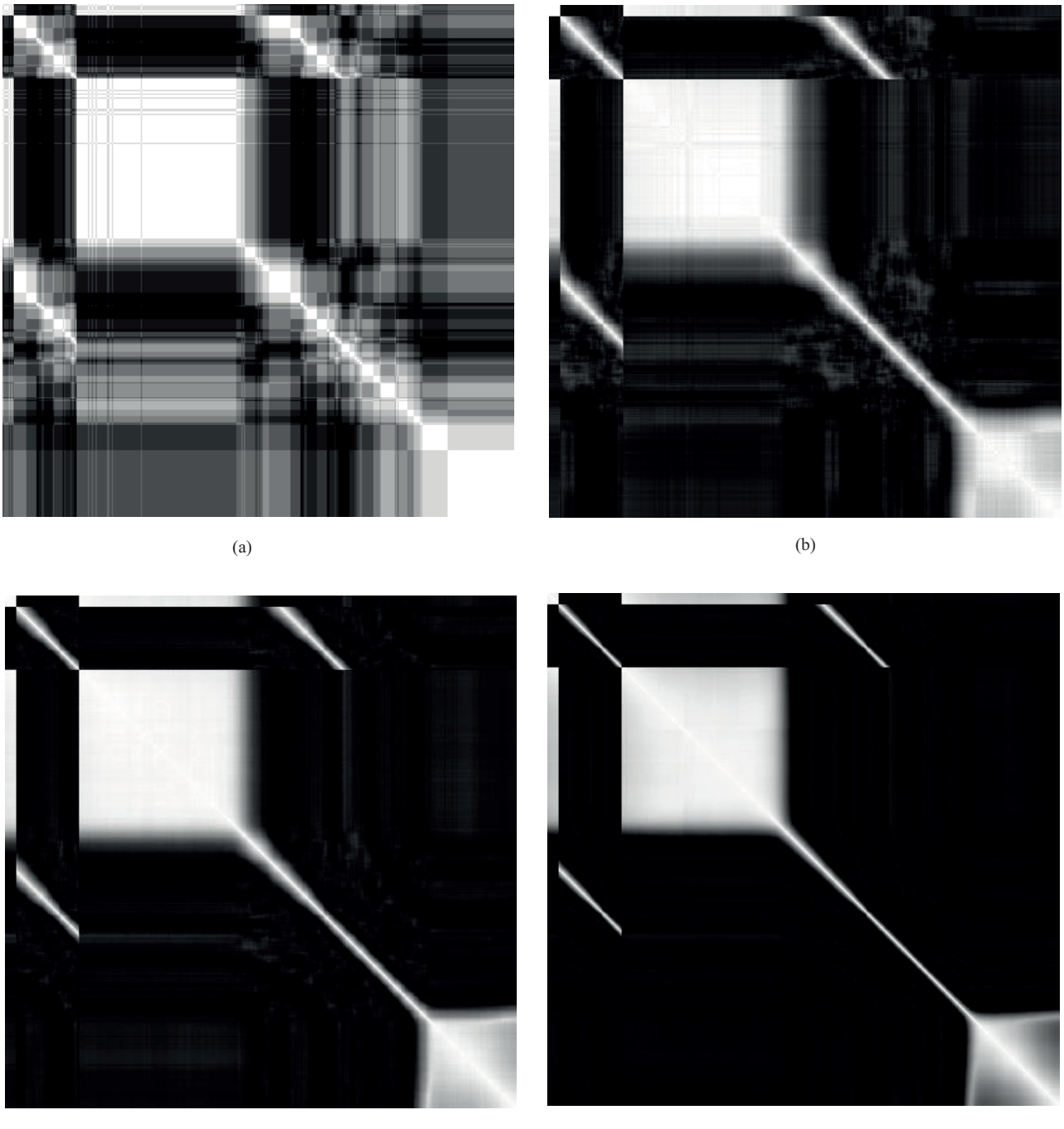


Figure 3. Tiny frame generation and feature construction.

All frames are processed in the same way, and binary feature vectors $V_1, V_2, \cdots, V_k$ are created for each frame. The last step of the "correlation image generation" function is to construct a correlation matrix denoted by $I = \{c_{xy} \,|c_{xy} \in [0,1], x \in 1 \cdots k, y \in 1 \cdots k\}$, where c_{xy} denotes the correlation value between V_x and V_y. The matrix CI is represented by the proposed method as a normalized image, where values 0 and 1 correspond to the least similar and most similar feature vectors, respectively.

We perform an experiment to show the reason for choosing 12 as a cell size during the construction of the correlation image. Figure 4 indicates the results of the experiment. Sizes 4, 12, 24, and 48 are chosen as cell sizes, and the created correlation images are given in this figure. When cell size becomes 4, the forgery line cannot be viewed clearly, as can be seen in Figure 4a. When cell size increases, the forgery line can be viewed exactly, as can be seen in Figures 4b–4d. However, when we choose cell size to be larger (such as 48), the running time of the proposed method deteriorates. The main reason for worse running time is the size of the feature vector. The feature vector for each frame becomes 24^2 and 48^2 for cell sizes 24 and 48, respectively. Thus, even though larger cell sizes give better forgery lines, we must consider running time performance during the construction of the algorithm. In this regard, we choose to use 12 as cell size because it gives moderate forgery line visibility, as can be seen in Figure 4, and better running time performance when compared to larger cell sizes (24 and 48).



(a) (b)

(c) (d)

Figure 4. Different sizes of block representation: a) 4 × 4, b) 12 × 12, c) 24 × 24, d) 48 × 48.

3.2. Coarser forgery line detection

The function obtains the correlation image *CI* as input, and outputs the coarser forgery line. When a malicious user makes a forgery operation on the video, a line with slope –45° will appear on *CI*. Let us assume that frames 10–30 are copied and pasted onto frames 110–130. If the malicious user saves the forged video in uncompressed

format, CI will contain a line that starts from $(10, 110)$ and ends on $(30, 130)$ with slope $-45°$. However, the attacker will often save the forged video with an encoder. Thus, some effort must be made to find the exact location of the forgery line. In this work, we apply the coarser-to-finer approach to determine the line. While the "coarser forgery line detection" function estimates the approximate location of the forgery line, the next function (Section 3.3) will detect its exact location.

The proposed method extracts the longest line from CI as the first step. The Hough transform is applied to CI to detect lines with slope $-45°$. CI must be binarized before the line detection algorithm is applied. Pixels on the lower triangular part (the upper triangular part contains the same information) and at the diagonal (they represent autocorrelation values) are set to zero on CI during binarization, as can be seen in Figure 5. The region that is restricted by the yellow and blue lines is similarly set to zero, because it contains high correlation values that are the result of spatially near frames. The width of this region is determined to be 60 by the proposed method, which assumes that a forgery operation does not occur within two seconds for a video of 30 fps. The other pixel values in CI are binarized to obtain bCI, using threshold value th, and its initial value is set to 1. If the line detection algorithm, explained below, cannot detect any line on bCI, threshold value th is decremented by 0.1 and a new bCI is obtained. The threshold value determination process continues until a line is detected on bCI. The method stops the decrement of th when a line is detected on bCI. The following steps are applied on bCI by the line detection algorithm to obtain the longest line:

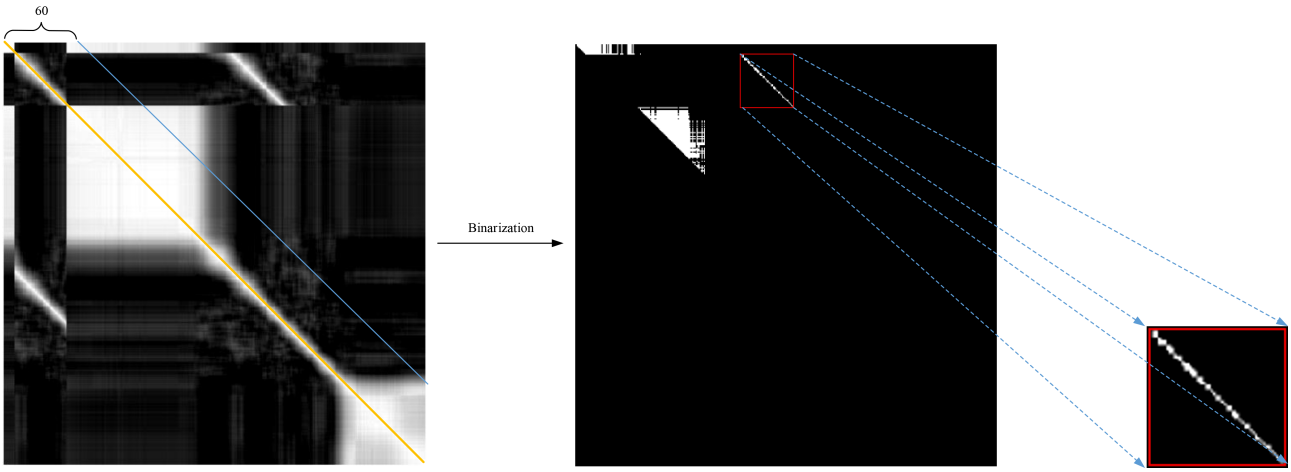


Figure 5. Coarser forgery line detection on CI.

- Apply Hough transform on bCI and return Hough transform matrix H, ρ, and θ values.
- Locate p peaks in H.
- Use peaks p, ρ, and θ values and determine the line segments with minimum length $length_{min}$.
- Return line $l = \{(x_s, y_s), (x_e, y_e)\}$, which has the maximum length.

The slope of the longest line l, returned by the line detection algorithm, can change in a range $(-46°, -44°)$, because the forged video is recompressed by the malicious user. The start (x_s, y_s) and end points (x_e, y_e) of the line are calibrated to give the correct slope, as in Eq. (1). The distance on the x-axis between the start and

end points of l must be equal to the distance on the y-axis, because the slope must be $-45°$. Assume that the distance on the x-axis is $d_x = |x_s - x_e|$ and similarly the distance on the y-axis is $d_y = |y_s - y_e|$.

$$\text{If } (d_y/d_x) > 1 \Rightarrow x_e = x_e + (d_y - d_x) \quad \text{If } (d_y/d_x) < 1 \Rightarrow y_e = y_e - (d_x - d_y) \tag{1}$$

The equation given in (1) updates the start and end points of line l to give the correct slope value $-45°$. Forgery line l is detected in a coarser manner in this section. The next function will locate the forgery line with higher precision.

3.3. Finer forgery line localization

In this function, the proposed method locates the forgery line at exact position and returns the index values of the forged frames. The function gets the coarser forgery line l from the previous function. The first step is to investigate the exact position of the line. The line l is slid in sequence on the x-axis, and lx_1, lx_2, lx_3 lines, which start from $(x_s + 1, y_s)$, $(x_s + 2, y_s)$, $(x_s + 3, y_s)$, respectively, are obtained. Furthermore, the sliding operation is realized through the y-axis, and the ly_1, ly_2, ly_3 lines are constituted. Figure 6 shows an example of sliding operation. Frames 10–30 are copied and pasted onto frames 110–130 to create the test video in the example. The red line represents the forgery line that is the output of the "coarser forgery line detection" function, whereas blue and yellow lines represent lx_1, lx_2, lx_3 and ly_1, ly_2, ly_3, respectively.

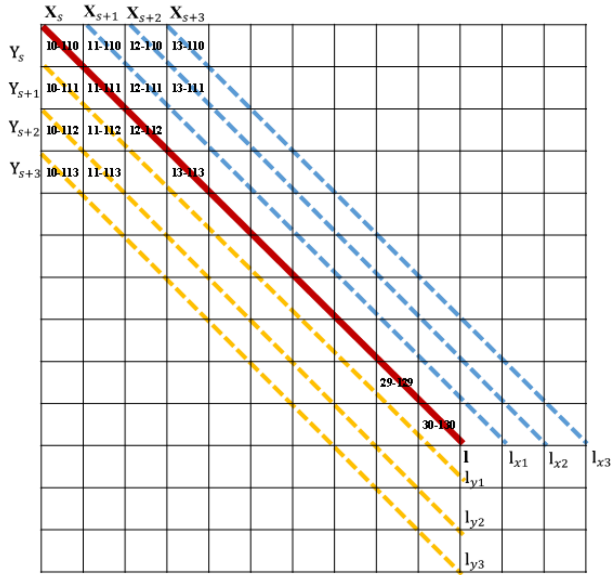


Figure 6. Example of a sliding operation.

Points on the line that have integer-valued coordinates correspond to the indexes of the similar frames. For example, point $(10, 110)$ means that the 10th frame is similar to the 110th frame. The function processes the lines separately and calculates a score to determine their exact forgery line. The score for each line is calculated as follows.

3.3.1. Score generation

Peak signal-to-noise ratio (PSNR) values for similar frames, which are represented by the points that have integer coordinates on the line, are calculated, and the score for the current line is determined to be the average

of all PSNRs. PSNR for the ith and jth frame F_i and F_j is calculated as in Eq. (2), where F_i^{xy} denotes the pixel values at xth row and yth column of the ith frame.

$$MSE = \frac{1}{MN}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}\left(F_i^{xy} - F_j^{xy}\right)^2 \quad PSNR\left(F_i, F_j\right) = 10log_{10}\left(\frac{255^2}{MSE}\right) \tag{2}$$

Figure 7 shows the score generation for each line. The function chooses as forgery line the line that gives maximum score $\max_{score}$. The last part of the "finer forgery line localization" is the shrinking/expanding procedure. Shrinking/expanding procedure steps are shown in Figure 8.

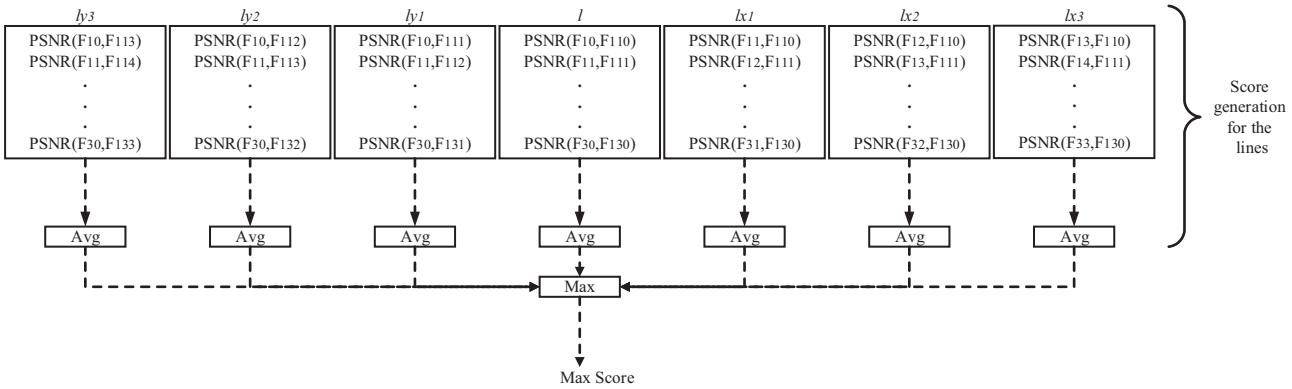


Figure 7. Score generation for each line.

3.3.2. Shrinking and expanding procedure

The shrinking procedure investigates the correctness of the start and end points of the forgery line, according to PSNR values. The procedure that is given below is termed separately for the start and end points of the forgery line. The values of the start (x_s, y_s) and end points (x_e, y_e) of the forgery line can be updated with the shrinking procedure, with which the third parameter determines the shrinking direction. If the third parameter is 'S', the first two parameters correspond to the start point of the line. If the third parameter is 'E', the first two parameters of the function are the end point of the line. The shrinking procedure is shown in Figure 9.

The expanding procedure will be applied according to the result of the shrinking procedure. If one of the start or end points of the forgery line is not modified by the shrinking procedure, the expanding operation will be applied, and the forgery line can be modified by the expanding operation. This procedure realizes the expanding operation in a window w and uses parameter "*StartOrEnd*" to determine the expanding direction. Thus, the expanding procedure is applied from the start position, and a new start position for the line is determined as in Figure 10. "Finer forgery line localization" function reports the integer points on the forgery line $l = \{(x_s, y_s), (x_e, y_e)\}$ as forged frame numbers.

4. Experimental results

In this section, we conduct experiments to show the effectiveness of the method and compare it to similar works. 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 with Virtual Dub, an open source video editor. Randomly picked videos from the Surrey University Library for Forensic Analysis (SULFA) [19]

Input: *x, y, StartOrEnd* where *(x, y)* represents a point on the line and StartOrEnd determines the point is (x_s, y_s) *or* (x_e, y_e)	
Shrinking Procedure	Expanding Procedure
1. Calculate PSNR between F_x and F_y as $PSNR_{current}$ 2. If $(PSNR_{current} < \max_{score} * 0.9) \wedge (StartOrEnd \equiv' S')$ • Get the next point on the forgery line (x, y) and set the value of (x_s, y_s) to (x, y) 1. Else 2. If $(PSNR_{current} < \max_{score} * 0.9) \wedge (StartOrEnd \equiv' E')$ • Get the former point on the forgery line (x, y) and set the value of (x_e, y_e) to (x, y) 1. Else 2. Go to Step (1) until the midpoint of the line. 3. $If\ (StartOrEnd \equiv' S') \Rightarrow$ return (x_s, y_s) 4. $If\ (StartOrEnd \equiv' E') \Rightarrow$ return (x_e, y_e)	1. $If\ (StartOrEnd \equiv' S') \Rightarrow$ Get the former point from (x, y) on the forgery line and set (x, y) to designate the new point. 2. $If\ (StartOrEnd \equiv' E') \Rightarrow$ Get the later point from (x, y) on the forgery line and set (x, y) to designate the new point. 3. Calculate PSNR between F_x and F_y as $PSNR_{current}$ 4. If $\left((PSNR_{current} > \max_{score} * 0.9)\right)$ • $If\ (StartOrEnd \equiv' S') \Rightarrow$ Set (x_s, y_s) to (x, y) • $If\ (StartOrEnd \equiv' E') \Rightarrow$ Set (x_e, y_e) to (x, y) • Go to Step (1) until w points are processed. 1. $If\ (StartOrEnd \equiv' S') \Rightarrow$ return (x_s, y_s) 2. $If\ (StartOrEnd \equiv' E') \Rightarrow$ return (x_e, y_e)
Output: Updated coordinates of start or end points. The third parameter determines which point will be updated.	

Figure 8. Shrinking and expanding procedure steps.

are used to create forged samples. A resolution of 320 × 240 is used to create the test video database. Another ten forged test videos are generated using movies. Table 1 lists the details of the videos in the current test data set, which is published at http://ceng2.ktu.edu.tr/%7Egulutas/test_database2.zip.

Table 1 indicates the forgery positions for each test video. For example, frames 11–61 are inserted after the 221st frame in test Video 1 to create a forged video. The test videos after Video 20 are created using various move files. Randomly chosen frames from a video are copied and inserted into the randomly chosen positions in the same video during the generation of the test data set, and forged videos are encoded with open source MPEG-4 (Level 5 Advanced Simple Profile, ASP@L5) algorithm of Xvid codec with Virtual Dub. We group the experiments into two subsections, effectiveness and comparison tests. The first section gives an idea about the runtime performance and detection capability under compression for the proposed method, whereas the second section makes a comparison with similar works using runtime performance and detection capability metrics. Precision rate (PR), recall rate (RR), and detection accuracy (DA) are evaluated to compare the method to similar detection methods reported in the literature according to detection capability. The definitions for PR, RR, and DA are given in Eq. (3), where TP, TN, FP, and FN represent "authentic is

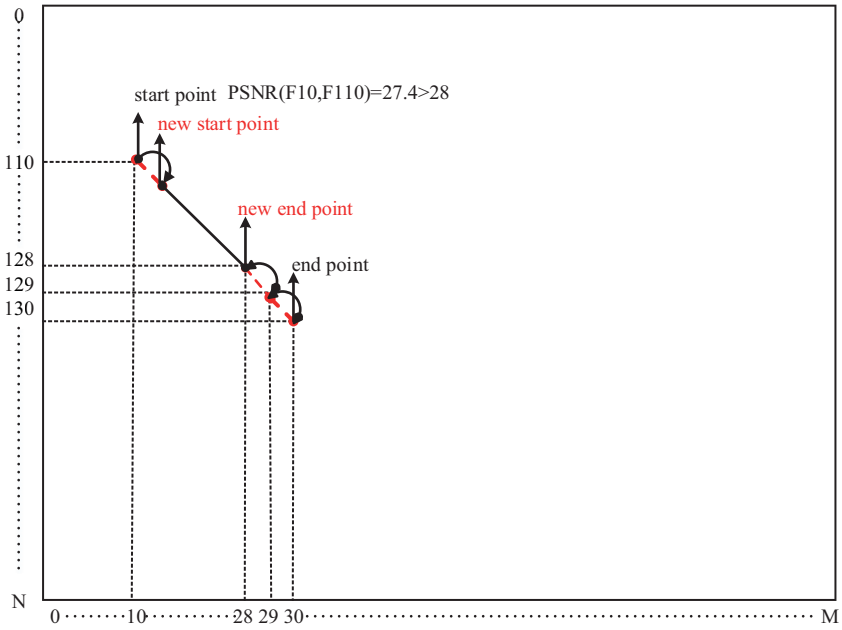


Figure 9. Shrinking procedure.

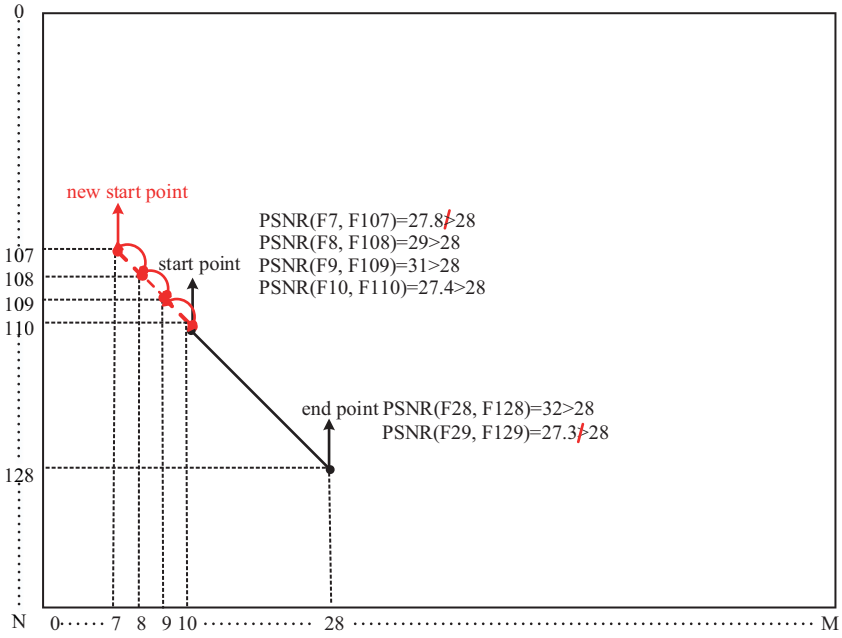


Figure 10. Expanding procedure.

detected as authentic", "forged is detected as forged", "authentic is detected as forged", and "forged is detected as authentic", respectively.

$$PR = \frac{TP}{TP + FP} \quad RR = \frac{TP}{TP + FN} \quad DA = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

A method's performance is satisfactory if it has higher precision and recall values. The total number of detections is represented by (TP + TN), and (TP + TN + FP + FN) is the total number of frames in the experiment.

Table 1. Details of the test videos.

Video name	Frame rate	Forgery operation	Total time (s)	Time (s/frame)
can_220_book	29 fps	11–61 are copied behind 221	25.5704	0.0628
can_220_flap(1)	29 fps	101–161 are copied behind 316	27.8090	0.0648
can_220_flap(2)	29 fps	61–101 are copied behind 281	20.0714	0.0540
can_220_garden(1)	29 fps	11–61 are copied behind 251	22.1294	0.0549
can_220_garden(4)	29 fps	101–141 are copied behind 181	22.0051	0.0610
can_220_man(2)	29 fps	11–61 are copied behind 161	22.0353	0.0565
can_220_road(1)	29 fps	151–221 are copied behind 241	21.9476	0.0592
can_220_road(5)	29 fps	131–191 are copied behind 221	23.2352	0.0653
can_220_room(3)	29 fps	131–171 are copied behind 261	26.1826	0.0601
can_220_street(3)	29 fps	31–71 are copied behind 211	25.2114	0.0773
can_220_street	29 fps	86–146 are copied behind 221	25.5235	0.0606
fuji_2800_busstop(4)	30 fps	6–61 are copied behind 191	25.4684	0.0717
nik_s3000_bridge	30 fps	11–41 are copied behind 281	27.9125	0.0864
fuji_2800_outdoor(4)	30 fps	31–71 are copied behind 131	25.4200	0.0820
fuji_2800_road(2)	30 fps	6–56 are copied behind 188	25.2308	0.0664
fuji_2800_road(5)	30 fps	61–101 are copied behind 264	27.8161	0.0605
fuji_2800_stair_outdoor	30 fps	41–81 are copied behind 216	27.8127	0.0752
fuji_2800_street(1)	30 fps	111–171 are copied behind 201	27.7623	0.0712
nik_s3000_bridge(1)	30 fps	121–171 are copied behind 221	27.9934	0.0710
nik_s3000_indoor_stairs	30 fps	11–51 are copied behind 106	27.8080	0.1220
Film 1	29 fps	166–293 are copied behind 331	29.2634	0.0585
Film 2	23 fps	235–305 are copied behind 572	65.6037	0.0914
Film 3	23 fps	1–55 are copied behind 100	11.1669	0.0634
Film 4	29 fps	233–342 are copied behind 355	84.3100	0.0969
Film 5	29 fps	459–519 are copied behind 735	190.5691	0.1454
Film 6	25 fps	65–230 are copied behind 631	83.5779	0.0955
Film 7	25 fps	571–632 are copied behind 731	107.7579	0.1089
Film 8	23 fps	895–935 are copied behind 41	112.6632	0.1138
Film 9	23 fps	0–236 are copied behind 855	188.8240	0.1530
Film 10	25 fps	0–115 are copied behind 601	97.1462	0.1055

Therefore, higher DA means that the method has a better detection rate. The details of the experiments are given in the following sections.

4.1. Effectiveness tests

In this section, we give the results of the experiments that were conducted to show the effectiveness of the proposed method. The first experiment reports the run time performance of the proposed method for all test videos given in Table 1, which lists the total and per frame execution time for the test videos. For example, the method reports forgery location for Video 2 within approximately 27.8 s, and the per frame duration for this video becomes approximately 0.06 s. Test videos given in Table 1 can be grouped into two parts. While the first 20 videos in Table 1 are created from the videos in the SULFA, the remaining videos are created using various movie files. The average per frame execution time for the videos in the first part (which are created using the videos in SULFA) is approximately 0.07 s. This value becomes 0.1 s for the videos in the second part. Furthermore, we give the DA, PR, and RR values for each video in the test database in Table 2. Average per

video DA, PR, and RR values in the first part (which are created using the videos in SULFA) is approximately 0.979, 0.995, and 0.978. These values become 1, 0.953, and 0.964 for the videos in the second part.

Table 2. General outline of the proposed method.

Video name	DA	PR	RR
can_220_book	0.9954	1.0000	0.9940
can_220_flap(1)	0.9565	1.0000	0.9441
can_220_flap(2)	0.9946	1.0000	0.9932
can_220_garden(1)	0.9950	1.0000	0.9934
can_220_garden(4)	0.9950	1.0000	0.9938
can_220_man(2)	0.9950	1.0000	0.9934
can_220_road(1)	0.8859	1.0000	0.8502
can_220_road(5)	0.9504	1.0000	0.9336
can_220_room(3)	0.9954	1.0000	0.9944
can_220_street(3)	0.9954	0.9045	0.9944
can_220_street	0.9266	1.0000	0.9930
fuji_2800_busstop(4)	1.0000	1.0000	1.0000
nik_s3000_bridge	0.9957	1.0000	0.9950
fuji_2800_outdoor(4)	0.9908	1.0000	0.9888
fuji_2800_road(2)	0.9954	1.0000	0.9940
fuji_2800_road(5)	1.0000	1.0000	1.000
fuji_2800_stair_outdoor	0.9957	1.0000	0.9947
fuji_2800_street(1)	0.9391	1.0000	0.9235
nik_s3000_bridge(1)	0.9957	1.0000	0.9944
nik_s3000_indoor_stairs	0.9957	1.0000	0.9947
Film 1	1.0000	0.9384	0.968
Film 2	1.0000	0.9965	0.9972
Film 3	1.0000	0.9705	0.9886
Film 4	1.0000	0.9938	0.9954
Film 5	1.0000	0.9834	0.9847
Film 6	1.0000	0.7552	0.7988
Film 7	1.0000	1.0000	1.0000
Film 8	1.0000	1.0000	1.0000
Film 9	1.0000	1.0000	1.0000
Film 10	1.0000	0.8937	0.9108

Another experiment shows the detection capability of the proposed method when the test videos are re-encoded after forgery operation. The last ten test videos given in Table 1 are re-encoded using various quality factors (QF), such as 1, 15, and 31, where 31 indicates the best compression ratio. DA, PR, and RR values are measured with these compressed test videos to show the robustness of the method against recompression.

Figure 11 indicates that the proposed method has approximately 1 and 0.996 values for DA, PR, and RR, respectively, when the forged video is resaved without any compression. If the forged test videos given in Table 1 are resaved with QF = 31 after forgery operation, DA, PR, and RR become 1, 0.953, and 0.964, respectively.

Figure 11 indicates that although DA value never changes, PR and RR values decrease when compression ratio increases. However, the method can detect forged videos with PR = 0.95 and RR = 0.96, even if the forged videos have been resaved QF = 31.

4.2. Comparison tests

In this section, we give the comparison results according to execution time and detection capability. The first experiment gives the average execution time for the proposed method and the other, as can be seen in Table 3. The average execution time results for the methods indicate that the proposed method can detect forgery operations on the test videos with less time. While the average per frame execution time for the methods in [1–4] is approximately 294.6, 140.6, 0.4, and 0.063, the proposed method can detect forgery operation on a frame within 0.07 s. Enhancing per frame execution time in video processing is important, because test videos can accommodate many frames, especially when surveillance videos are considered. Another experiment gives comparison results for the methods when their detection capabilities are considered. Figure 12 shows PR, RR, and DA values for the proposed method and other [1–4] for test videos given in Table 2. While the methods in [1–4] have approximately 0.3, 0.82, 0.94, and 0.98 PRs, respectively, the proposed method gives PR = 1. Moreover, recall rates and detection accuracy results indicate the superiority of the work compared to similar works.

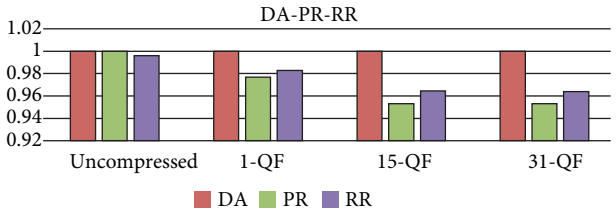


Figure 11. Detection capability of the proposed method according to different quality factors.

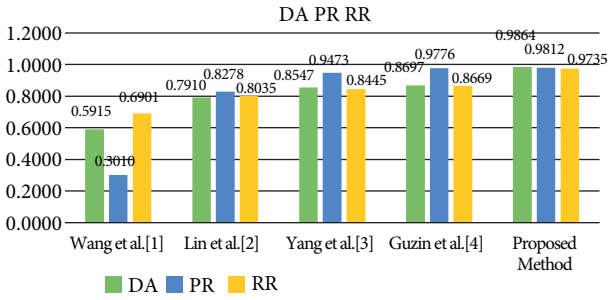


Figure 12. Comparison results according to detection capability.

Table 3. Tiny frame generation and feature construction.

Method	Execution time (s/frame)
Wang et al. [1]	294.67
Lin et al. [2]	140.60
Yang et al. [3]	0.41
Guzin et al. [4]	0.063
Proposed method	0.07

5. Conclusion

In this work, we proposed a new video forgery detection approach. The method extracts binarized DCT features from the frames and represents the similarity between them to be a correlation image. Elements of the correlation image give an idea about the similarity between corresponding frames. The method then investigates a line called forgery line on the correlation image, to determine the location of the forgery line in a coarser manner. The Hough transform is applied to the image to determine the location of the line. Furthermore, the proposed

method suggests two new procedures, shrinking and expanding, to locate the forgery line in a finer manner. The integer-valued points on the line indicate the corresponding index values of forged frames. Comparison and effectiveness tests, given in the experimental result section, show that the proposed method gives better execution time than [1–3] and better detection capability when compared to similar works [1–4].

The main difficulty in the frame duplication detection literature is to develop new methods that are resistant to compression attacks. Only one work in the literature considers this issue in their results [4]. Our results indicate that the proposed method gives better detection performance when compared to [4] under compression attacks. Finally, we aim to improve our method against frame mirroring attacks in our future work.

Acknowledgment

This study is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK, Project No. 115E214).

References

[1] Wang W, Farid H. Exposing digital forgeries in video by detecting duplication. In: ACM 2007 Workshop on Multimedia and Security; 20–21 September 2007; Dallas, TX, USA. New York, NY, USA: ACM. pp. 35-42.

[2] Lin GS, Chang JF. Detection of frame duplication forgery in videos based on spatial and temporal analysis. Int J Pattern Recogn 2012; 26: 1250017.

[3] Yang J, Huang T, Su L. Using similarity analysis to detect frame duplication forgery in videos. Multimed Tools Appl 2016; 75: 1793-1811.

[4] Ulutas G, Ustubioglu B, Ulutas M, Nabiyev VV. Frame duplication/mirroring detection method with binary features. IET Image Process 2017; 11: 333-342.

[5] Sitara K, Mehtre BM. Digital video tampering detection: An overview of passive techniques. Digit Invest 2016; 18: 8-22.

[6] Wang W, Farid H. Exposing digital forgeries in video by detecting double MPEG compression. In: ACM 2006 Workshop on Multimedia and Security; 26–27 September 2006; Geneva, Switzerland. New York, NY, USA: ACM. pp. 37-47.

[7] Wang W, Farid H. Exposing digital forgeries in video by detecting double quantization. In: ACM 2009 Workshop on Multimedia and Security; 7–8 September 2009; Princeton, NJ, USA. New York, NY, USA: ACM. pp. 39-48.

[8] Sun T, Wang W, Jiang X. Exposing video forgeries by detecting MPEG double compression. In: IEEE 2012 International Conference on Acoustics, Speech and Signal Processing; 25–30 March 2012; Kyoto, Japan. New York, NY, USA: IEEE. pp. 1389-1392.

[9] Vazquez-Padin D, Fontani M, Bianchi T, Comesana P, Piva A, Barni M. Detection of video double encoding with GOP size estimation. In: IEEE 2012 International Workshop on Information Forensics and Security; 2–5 December 2012; Costa Adeje, Spain. New York, NY, USA: IEEE. pp. 151-156.

[10] Jiang X, Wang W, Sun T, Shi YQ, Wang S. Detection of double compression in MPEG-4 videos based on Markov statistics. IEEE Signal Process Lett 2013; 20: 447-450.

[11] Huan Z, Huang F, Huang J. Detection of double compression with the same bit rate in MPEG-2 videos. In: IEEE 2014 International Conference on Signal and Information Processing; 9–13 July 2014; Xi'an, China. New York, NY, USA: IEEE. pp. 306-309.

[12] Wang W, Farid H. Exposing digital forgeries in interlaced and deinterlaced video. IEEE T Inf Foren Sec 2007; 2: 438-449.

[13] Li L, Wang X, Zhang W, Yang G, Hu G. Detecting removed object from video with stationary background. Lect Notes Comp Sci 2013; 7809: 242-252.

[14] Hsu CC, Hung TY, Lin CW. Video forgery detection using correlation of noise residue. In: IEEE 2008 Workshop on Multimedia Signal Processing; 8–10 October 2008; Brisbane, QLD, Australia. New York, NY, USA: IEEE. pp. 170-174.

[15] Su L, Huang T, Yang J. A video forgery detection algorithm based on compressive sensing. Multimed Tools Appl 2015; 74: 6641-6656.

[16] Bidokhti A, Ghaemmaghami S. Detection of regional copy/move forgery in MPEG videos using optical flow. In: IEEE 2015 International Symposium on Artificial Intelligence and Signal Processing; 3–5 March 2015; Mashad, Iran. New York, NY, USA: IEEE. pp. 13-17.

[17] Chao J, Jiang X, Sun T. A novel video inter-frame forgery model detection scheme based on optical flow consistency. Lect Notes Comp Sci 2013; 7809: 267-281.

[18] Singh VK, Pant P, Tripathi RC. Detection of frame duplication type of forgery in digital video using sub-block based features. In: EAI 2015 International Conference on Digital Forensics and Cyber Crime; 6–8 October 2015; Seoul, South Korea. Ghent, Belgium: European Alliance for Innovation, pp. 29-38.

[19] Qadir G, Yahaya S, Ho AT. Surrey University Library for Forensic Analysis (SULFA) of Video Content. Surrey, UK: University of Surrey, 2012.