Local tampering detection in video sequences

Paolo Bestagini, Simone Milani, Marco Tagliasacchi, Stefano Tubaro

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
piazza Leonardo da Vinci 32
20133 Milano, Italy
bestagini/milani/tagliasa/tubaro@elet.polimi.it

***Abstract*—Video sequences are often believed to provide stronger forensic evidence than still images, e.g., when used in lawsuits. However, a wide set of powerful and easy-to-use video authoring tools is today available to anyone. Therefore, it is possible for an attacker to maliciously forge a video sequence, e.g., by removing or inserting an object in a scene. These forms of manipulation can be performed with different techniques. For example, a portion of the original video may be replaced by either a still image repeated in time or, in more complex cases, by a video sequence. Moreover, the attacker might use as source data either a spatio-temporal region of the same video, or a region taken from an external sequence. In this paper we present the analysis of the footprints left when tampering with a video sequence, and propose a detection algorithm that allows a forensic analyst to reveal video forgeries and localize them in the spatio-temporal domain. With respect to the state-of-the-art, the proposed method is completely unsupervised and proves to be robust to compression. The algorithm is validated against a dataset of forged videos available online.**

I. Introduction

In the last few years, the Web has steadily moved towards more democratic forms of information-sharing. Indeed, many websites allow users to upload and share multimedia objects, including audio, images, and video sequences. This has determined an incredible growth of user generated content easily accessible online by anyone. However, the possibility of sharing self-produced media has not been followed by the development of methods to automatically verify the authenticity of the uploaded material. For this reason, while browsing multimedia content available on the Web, it is very common to run into forged and maliciously modified objects. In many cases, forgeries can be very realistic. Indeed, newscasts and newspapers are sometimes tricked and make use of forged pictures or videos as if they were authentic.

In order to address this issue, the multimedia forensic community has proposed many forgery detection algorithms, targeting different kinds of media [1]. More specifically, if we consider visual content, many forensic solutions were proposed for still-images [2], [3], whereas just a few methods address videos [4], which is a more challenging scenario. This is also due to the common sense that videos provide stronger forensic evidence than images, since forging a video is considered significantly harder than tampering with an image. However, manipulating videos has become a relatively easy task, thanks to the increasing number of dedicated user-friendly tools. These software solutions allow people to easily forge a video sequence in a way that is realistic, hence believable. Although creating a doctored video is more time consuming than forging an image, there is an urgent necessity for forgery-detection algorithms that can automatically prove or disprove the authenticity of a video. These algorithms can be divided into two main categories: i) algorithms for *tampering detection*, i.e., methods that verify the integrity of the entire video without localizing the forgery; ii) algorithms for *tampering localization*, i.e., methods that localize the forgery in the spatio-temporal domain.

Belonging to the first category are algorithms that detect a global manipulation. As an example, in [5] the authors detect the number of compression steps applied to a video as evidence of video editing. In case double compression has been applied, [6], [7] show hot to detect the first used codec in the compression chain. In [8] and [9] methods to detect if a video sequence was re-captured from a monitor are presented. In [10], the authors propose a method to detect if a video sequence was temporally interpolated. All these methods provide useful information to determine if a sequence was modified, either maliciously or not. However, they do not provide any information on the forgery location.

On the other hand, tampering localization methods aims to exactly detect where and when a video was forged, i.e., in the spatial (pixel) and temporal (frame) domain. In this case, different detectors were proposed for different kinds of attacks. In [11] video splicing (used to remove an object from a scene) is detected analyzing noise characteristics. In [12], the authors analyze two kinds of attacks: i) spatial copy-move (obtained duplicating an object within the same scene) is detected matching Histogram of Oriented Gradients (HOG); ii) temporal copy-move (obtained copying an object from a frame to another one) is detected exploiting MPEG-2 GOP structure. In [13], the authors analyze the case of splicing attacks, that is, copying a portion of a video sequence into another. The detector exploits the differences of noise characteristics between the original and the spliced sequence, thus being very sensitive to compression. In [14], both frame duplication and region duplication are detected by means of the analysis of correlation between suspect parts. However, when region duplication is concerned, the pair of tampered frames is assumed to be known a-priori.

MMSP'13, Sept. 30 - Oct. 2, 2013, Pula (Sardinia), Italy.
978-1-4799-0125-8/13/$31.00 ©2013 IEEE.

The main drawback of many state-of-the-art algorithms lies in the insufficient validation on realistically tampered videos. Since producing manually forged sequences is time consuming, results are often presented for synthetically tampered videos (i.e., obtained by automatically copy and pasting regions in different positions), whose authenticity could be immediately determined by simple visual inspection. Only in some cases qualitative results are presented for just a few (typically 2 or 3) hand-made realistic forged sequences. In order to address this issue, in [15] the authors have recently presented a publicly available and open database of realistic forged sequences. The Surrey University Library for Forensic Analysis (SULFA) database is still at an early stage. However, the submission of additional forged video sequences is encouraged, and it may become a reference for future algorithms.

The main contribution of this paper is twofold. First, we propose a blind and automatic algorithm to detect local tampering in videos, addressing different kinds of attacks. Second, we contributed to the expansion of the SULFA database presented in [15] with additional videos, in order to validate our method on a broader set of realistic sequences and, at the same time, offer to the scientific community a precious benchmarking dataset for their future investigations.

The proposed algorithm is able to detect whether a spatio-temporal region of a sequence (i.e., a block of connected pixels in the spatio-temporal domain) was replaced by either a series of fixed images repeated in time, or a portion of the same video taken from a potentially different time interval. In the first case, the algorithm detects the attack by analyzing the footprint left on the residual computed between adjacent frames, and proves to be robust to mild compression. In the second case, the attack is detected exploiting a correlation analysis similar to [14]. However, our approach is fully automatic, and the position of tampered frames is not assumed to be known a-priori.

The rest of the paper is organized as follows. Section II illustrates in detail the kinds of attack that we aim to detect, and the footprints left in videos. Section III presents the rationale behind the proposed detection algorithm and the details of the implementation. In Section IV we report the results obtained on a realistic dataset, highlighting our contribution. Finally, in Section V we draw some conclusive remarks on this work, and present possible future research directions.

II. Problem formulation

In this paper we propose a method to detect one of the most common kinds of video forgery. That is, the attacker substitutes a part of a video, by either adding or removing something to/from a scene. Specifically, we target the challenging scenario in which a small spatio-temporal region is replaced, which is far more complex than full-frame replacement.

When applied to still-images, this attack consists in replacing a pixel region of the image using another region from either the same image or a different one (copy-move forgery). With this kind of attack it is then easy to either duplicate or remove objects from a scene (e.g., by copying and pasting part of the background over an object). However, video sequences

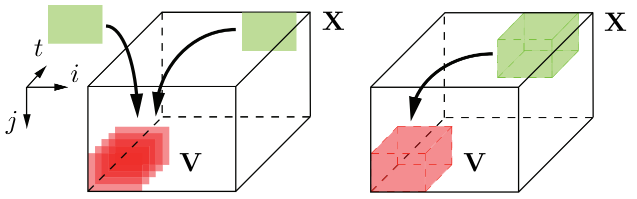


Fig. 1: Possible attack strategies. The volume $\mathbf{V}$ (red) is replaced by the repetition of the same image (green) either from the same video or from another source (on the left). Alternatively, the volume $\mathbf{V}$ (red) is replaced with a volume (green) of the same dimension (on the right).

are characterized by an additional degree of freedom due to the temporal dimension. For this reason, when applied to videos, the attack consists in replacing 3D volumes (in the spatio-temporal domain), rather than 2D regions. It is worth to notice, that typically the substitution is followed by a local filtering operation (e.g., brightness or contrast adjustment) in order to make the tampering more realistic.

Let $\mathbf{X} = \{x_{i,j}^t\}$ denote a video sequence, where $i \in [1, I]$, $j \in [1, J]$, and $t \in [1, T]$ are the spatial and temporal coordinates of pixel samples indexed by integer numbers. The attack aims to replace the original set of connected pixels represented by the volume $\mathbf{V}$, with another set of connected pixels $\hat{\mathbf{V}}$ of the same size of $\mathbf{V}$, to obtain the forged sequence $\hat{\mathbf{X}}$. In general, the shape of the volume $\mathbf{V}$ is arbitrary. For the sake of clarity, we start considering the simple case in which $\mathbf{V}$ is a box-shaped volume of samples. That is,

$$\mathbf{V} = \{x_{i,j}^t \mid i \in [i_0, i_1],\ j \in [j_0, j_1],\ t \in [t_0, t_1]\}. \quad (1)$$

The general scenario can be accommodated by considering the spatial indices to be time-dependent.

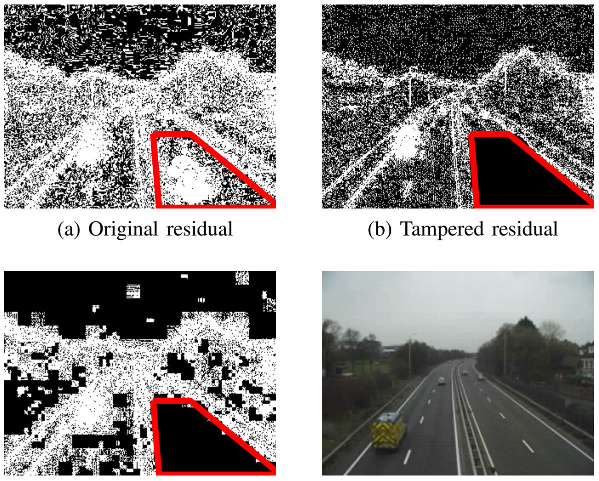
However, removing this dependency does not affect neither the problem formulation, nor the algorithm, and allows us to use a more compact notation.

In practice there are two possible choices for selecting the set of pixels $\hat{\mathbf{V}}$, which determine the nature of the attack. The first possibility is to replace the forged region with a series of images, and the second consists in replacing that region with a portion of video. Figure 1 shows these two scenarios. Let us now analyze these two possibilities.

A. Image-based attack

This method consists in pasting a fixed image over a spatial portion of a frame and repeating it in time (see left side of Figure 1). Since the image content does not move in time, this attack is generally applied to static scenes (e.g., when the video comes from a steady camera and has a fixed background). For this reason it is easy to replace $\mathbf{V}$ using either an image taken from a frame of the same video (e.g., the background), or another image (e.g., to introduce some text or additional objects).

When the image comes from the same video, $\hat{\mathbf{V}}$ is populated repeating in time a 2D region of the $\bar{t}$-th frame $\{x_{i,j}^t | \ i \in [i_0, i_1],\ j \in [j_0, j_1],\ t = \bar{t}\}$. If the image comes from another



(a) Original residual (b) Tampered residual (c) Tampered and compressed residual (d) Tampered frame

Fig. 2: Effect of image copy-move attack on the residual between two adjacent frames of an original sequence (a), its forged version (b), and its forged and compressed version (c). The map is black only for null residual. The tampered region is the area inside the red shape. The realistic tampered frame where a car has been removed comes from the database presented in [15], and it is shown in (d).

source, the concept is the same, but pixels of $\hat{\mathbf{V}}$ do not come from $\mathbf{X}$.

This attack leaves a characteristic footprints on the sequence. More specifically, since the same image is repeated in time, the difference between adjacent frames is exactly zero for pixels belonging to the area where the image was pasted. At the same time, when the video is compressed after the attack, frames differences assume many zero values also in other regions. This is due to the compression-induced redundancy, which sometimes duplicates blocks of pixels in time, e.g., due to the skip mode available in most video coding architectures. Figure 2 shows an example of the residual between adjacent frames for: a) an original; b) a forged; and c) a forged and compressed video. By exploiting this characteristic footprint it is possible to detect and localize this kind of tampering as described in Section III.

B. Video-based attack

This method consists in replacing a part of the sequence with a portion of video (see right side of Figure 1). Typically, to better integrate the duplicated region in the new part of the video, a local filtering operation is applied. This attack is typically used for scenes characterized by motion (e.g., to duplicate moving objects, or the background when the camera moves). However, since it is more difficult to realistically integrate two different videos (because of possibly different motion, illumination, etc.), this attack is commonly operated by substituting $\mathbf{V}$ with a set of pixels coming from the same video. This means populating $\hat{\mathbf{V}}$ with a set of connected pixels according to eq.(1).

This attack does not leave peculiar footprints such as those left by the image-based attack. However, since the forged region $\hat{\mathbf{V}}$ comes from the same video sequence $\mathbf{X}$, we can exploit a correlation analysis to find the duplicated region.

III. Detection algorithm

In order to detect and localize video tampering, we propose a two-step algorithm. The first step identifies videos attacked with image-based attacks, and the second step detects sequences attacked with video-based attacks. If both steps fail, the sequence is considered authentic.

A. Detection of image-based attack

To detect image-based attacks, we analyze the zero-motion video residual, obtained by taking the difference between pixels in the same spatial position on consecutive frames. Indeed, as previously shown, the residual is zero where images were spliced. For this reason we search for frames with a region of zero residual that remains constant in time. In other words, we aim to find the largest 3D bounding volume that contains only zero residual values. To achieve this goal, we propose an algorithm based on iterative morphological operations and clustering.

The morphological operation that we apply aims to compute a binary 3D map, where 1 indicates that a pixel might have been tampered with. To this purpose, let us define the video residual as the difference between adjacent frames

$$r_{i,j}^{t} = x_{i,j}^{t} - x_{i,j}^{t+1}, \tag{2}$$

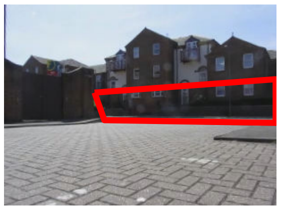
and the residual binary mask as

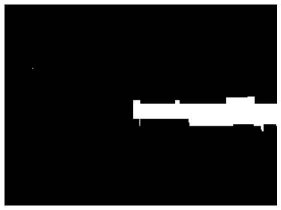
$$m_{i,j}^{t} = \begin{cases} 1 & \text{if } r_{i,j}^{t} = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

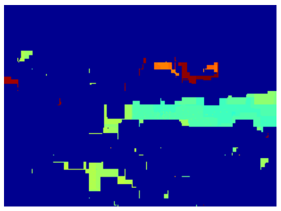
with $i \in [1, I], j \in [1, J], t \in [1, T-1]$. The binary mask maps the 0 residual values to 1, and sets everything else to 0. Let $\mathbf{M} \in \{0,1\}^{I \times J \times T-1}$ denote the 3D matrix whose elements are $m_{i,j}^{t}$. Then, we apply morphological erosion to $\mathbf{M}$ with a 3D Structuring Element (SE) $\mathbf{H}^{di,dj,dt}$ of size $di \times dj \times dt$, composed by ones, obtaining the final 3D map

$$\mathbf{E} = \{e_{i,j}^{t}\} = \mathbf{M} \ominus \mathbf{H}^{di,dj,dt}, \tag{4}$$

where $\ominus$ represents morphological erosion. In this situation, erosion acts as a filter that removes sub-volumes of $\mathbf{E}$ containing just a few values equal to 1 (i.e., small regions whose residual is equal to zero), which are more likely to be due to tampering than to compression. Indeed, compression introduces high correlation between frames, therefore the residual may assume zero value even in non-tampered regions. The size of $\mathbf{H}$ determines the minimum block of null residual that we accept as not due to compression. As a matter of fact, using a large structuring element $\mathbf{H}$ would result in deleting all traces of tampering. Conversely, using a small structuring element $\mathbf{H}$ would lead to mistaking every small volume with residual equal to 0 for a tampered area. For this reason, we start from a large value of $\mathbf{H}$ ($16 \times 16 \times 30$ in our experiments), and decrease it iteratively, until we detect a plausible tampering


(a) Tampered frame


(b) Detected mask


(c) $f^1_{i,j}$

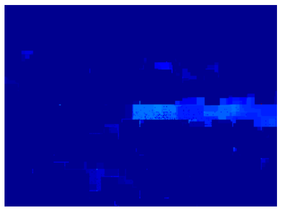

(d) $f^2_{i,j}$

Fig. 3: A tampered and compressed frame from [15] (a) whose tampered region lies inside the red shape, the 2D projection on (i, j) domain of the estimated tampering location (b), and the 2D distributions of $f^1_{i,j}$ (c) and $f^2_{i,j}$ (d).

region according to the criteria indicated below. In case none of the criteria is met, the iteration is terminated when we reach the smallest acceptable value of $\mathbf{H}$ ($4 \times 4 \times 5$ in our experiments).

In principle, each value $e^t_{i,j} = 1$ indicates tampering on that pixel position. However, in order to evaluate which pixels actually belong to a tampered area, we associate to each pair of spatial coordinates (i, j) a feature vector $F_{i,j} = [f^1_{i,j}, f^2_{i,j}]$. The two features are computed as follows:

- $f^1_{i,j}$: this feature is the cardinality of the largest set of adjacent ones in $e^t_{i,j}$ along the temporal direction. It represents the largest number of consecutive frames possibly tampered in the position (i, j).
- $f^2_{i,j}$: this feature is the t value from which the largest set of adjacent ones starts. It represents the starting frame of the possible tampering of length $f^1_{i,j}$.

By simply analyzing $F_{i,j}$ values, we can find the largest volume of possibly tampered pixels starting from the same frame. More specifically, we search for the pixel positions (i, j) with the highest $f^1_{i,j}$ values, and check if they start from the same time position given by $f^2_{i,j}$. If this volume is bigger than a given threshold (set according to the minimum tampering volume that we want to detect) we detect the presence of tampering. The tampering localization map is then built according to the pixels belonging to the detected cluster. Figure 3 shows a tampered frame, the values of $f^1_{i,j}$ and $f^2_{i,j}$, and the detected tampering mask.

B. Detection of video-based attack

The video-based attack does not leave a characteristic footprint such as that left by the image-based attack. For this reason, this kind of attack is not detected by the algorithm described in Section III-A. However, in practical situations, it is customary to replace a video region with another region from the same sequence (e.g., background copy-move to remove an object or a person). Hence, we propose a correlation method similar to that in [14], which aims to find the duplicated content. Unlike [14], we also detect which are the tampered frames, without assuming a-priori knowledge, thus moving from a semi-supervised to a fully unsupervised method.

The main idea of this step is to detect duplicated content in the 3D domain by cross-correlating small 3D blocks. Indeed, rather than simply correlating frame regions, we correlate spatio-temporal portions of $\mathbf{X}$. In order to reduce the computational complexity, yet achieving high accuracy, we resize the sequence in the spatial domain by a factor or 5, while retaining the full temporal resolution.

To this end, we first compute the residual matrix $\mathbf{R} = \{r^t_{i,j}\}$ of the downscaled sequence according to eq. (2). Analyzing $\mathbf{R}$ rather than $\mathbf{X}$ allows us to remove the effect of linear operations (e.g., brightness adjustment) that may have been applied to the duplicated block. Then, we split $\mathbf{R}$ into non overlapping 3D blocks $\mathbf{B}^n_m$ of size $di \times dj \times dt$, where n is the starting time index of a block, and $m \in [1, M]$ is the block index. We start analyzing all the blocks starting from a given time instant. If none of these blocks is detected to be duplicated (according to the method illustrated below), we analyze the next set of blocks (i.e., we increase the value n).

The detector is based on the phase-correlation between $\mathbf{B}^n_m$ and $\mathbf{R}$ as

$$\mathbf{C}^t_{i,j}(\mathbf{B}^n_m) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{B}^n_m)\mathcal{F}(\mathbf{R})^*}{|\mathcal{F}(\mathbf{B}^n_m)\mathcal{F}(\mathbf{R})^*|}\right), \tag{5}$$

where $\mathcal{F}$ is the Fourier transform operator, and * indicates the complex conjugate. This 3D correlation computes the similarity between a selected block $\mathbf{B}^n_m$ and the rest of the sequence. Notice that we make use of phase-correlation since it is computationally efficient, and in [14] the authors prove its robustness in detecting duplications. Let us define the maximum correlation value obtained for each time position as

$$c^t_{\mathbf{B}^n_m} = \max_{i,j}\left(|\mathbf{C}^t_{i,j}(\mathbf{B}^n_m)|\right). \tag{6}$$

Figure 4 shows the behavior of $c^t_{\mathbf{B}^n_m}$ for a duplicated and a non-duplicated block. The most prominent peak is due to auto-correlation, i.e., it is located in the exact time position n from which $\mathbf{B}^n_m$ starts. The position of the second highest peak represents the position of a possible duplication (e.g., at time $\tilde{n}$). If the second peak is sufficiently high (at least 0.6 times the first peak in our experiments), we associate a confidence value to the block $\mathbf{B}^n_m$ according to the max/mean ratio

$$p_{\mathbf{B}^n_m} = \frac{\max(c^t_{\mathbf{B}^n_m})}{\frac{1}{(T-1)}\sum_t c^t_{\mathbf{B}^n_m}}. \tag{7}$$

Among the M blocks starting at frame n, the one characterized by the largest value of $p_{\mathbf{B}^n_m}$ is the most likely duplicate candidate. To take the final decision, we compare this block with those starting from the frame whose time index corresponds to the second peak in $c^t_{\mathbf{B}^n_m}$ (i.e., the blocks $\mathbf{B}^{\tilde{n}}_m$, if the second

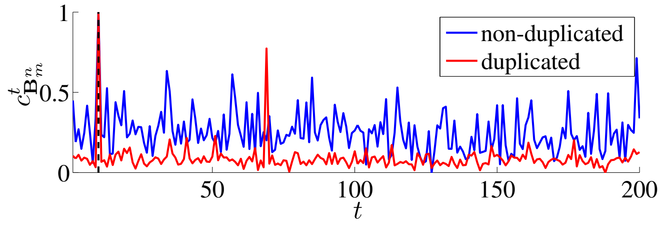


Fig. 4: Values of $c^t_{\mathbf{B}^n_m}$ normalized between 0 and 1 for a non-duplicated block (blue), and for a duplicated one (red). The dashed line represent the t value from which $\mathbf{B}^n_m$ starts. When the block is duplicated, a second prominent peak appears and $p_{\mathbf{B}^n_m}$ assumes an higher value.

peak is located at $\tilde{n}$). Duplication is detected if a matching block is detected, i.e.,

$$\arg\min_{\tilde{m}} \mathrm{MSE}(\mathbf{B}^n_m, \mathbf{B}^{\tilde{n}}_{\tilde{m}}) < \tau \tag{8}$$

where $\mathrm{MSE}(\cdot,\cdot)$ computes the mean square error between two blocks and τ is a user-defined threshold to tune the tradeoff between false positives and false negatives. Otherwise, if the condition in (8) is not satisfied, we move to the next value of n (i.e., the next set of blocks) and iterate the process until the whole sequence is scanned. If no duplication is detected, it is possible to chose a different scale value for the downsized sequence, resize it, and repeat the analysis. However, in our experiment this step was never necessary.

IV. Results

In order to validate the proposed algorithm we tested it on 120 realistic sequences with resolution of 320×240 pixels and of approximatively 300 frames each[1]. As it concerns the image-based attack, we used

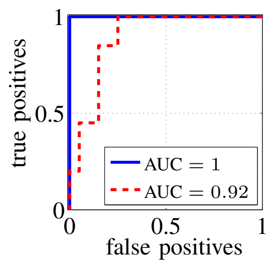
- 10 sequences from [15], divided in 5 original ones and their 5 forged versions using the image-based attack.
- 30 sequences obtained re-encoding the 10 sequences from [15] with H.264/AVC with Quantization Parameters (QP) in $\{10, 15, 20\}$ and GOP $= 150$ (i.e., a difficult scenario in which the residual is often set to zero because of the presence of many predicted frames).

As it regards the video-based attack, we used

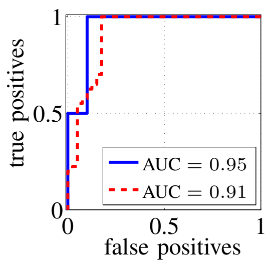
- 20 sequences, divided in 10 original and their 10 versions forged by us using the video-based attack.
- 60 sequences obtained re-encoding the 20 above mentioned sequences with H.264/AVC with QP in $\{10, 20, 30\}$ and GOP $= 150$.

Notice that we re-encoded the sequences to test the detector robustness to coding. Indeed the original and the forged videos are only slightly compressed at the origin, setting the codec to achieve high quality. For this reason we refer to the set of sequences not specifically re-compressed by us as *not re-compressed* sequences.

[1]A preview of original and forged sequences is available at: http://youtu.be/45J_092rpD0. The dataset is available at: http://www-dsp.elet.polimi.it/ispg/REWIND/forged_sequences_yuv.zip.



(a) image-based attack



(b) video-based attack

Fig. 5: ROC curves and AUC values for tampering detection on videos forged with both the image-based attack (a), and the video-based one (b). Blue line represents the ROC computed only on not re-compressed sequences, while the red curve represents the ROC curve computed on all the sequences.

QP	TP	FN	TN	FP
not re-compressed	0.75	0.25	**0.97**	0.03
10	0.71	0.29	**0.98**	0.02
15	0.58	0.42	**0.96**	0.04
20	0.44	0.56	**0.84**	0.16

TABLE I: Average tampering localization results for image-based attack and different compression rates (QP).

A. *Results on image-based attack*

To validate the detector in this scenario, we tested it on the 40 sequences described above to this purpose. Since this kind of tampering detection is based on a threshold, we built a Receiver Operating Characteristic (ROC) curve testing different threshold values. Figure 5a shows the ROC curves when the detector is used on the 10 not re-compressed sequences from [15] (blue), and when they are also re-compressed (red). We notice that the footprint left by the forgery is quite evident on not re-compressed sequences. Indeed, not re-compressed original sequences rarely exhibit a zero-valued residual. For this reason the Area Under the Curve (AUC) for this case is 1, which means perfect detection accuracy. On the other hand, as expected, if the sequences are re-compressed, the AUC decreases, since it is possible to detect original sequences as forged and vice-versa. However, results are still good, showing AUC $= 0.92$.

As it regards tampering localization, we computed on the first forged frame the amount of pixels detected as tampered for sequences correctly detected as forged. To this purpose, Table I shows the values of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) rates for different amount of compressions (i.e., different QP) averaged for different sequences. TP rate is computed as the number of pixel correctly identified as forged, normalized on the real number of forged pixels. TN rate is computed as the number of pixels detected as not-forged, normalized on the number of actually not-forged pixels. Notice that the high TN rate denotes that our detector correctly identifies not-forged pixels with high probability. On the other hand, TP rate indicates that some tampered pixels are not identified. However, for not re-compressed sequences, the 75% of forged pixels is correctly detected.

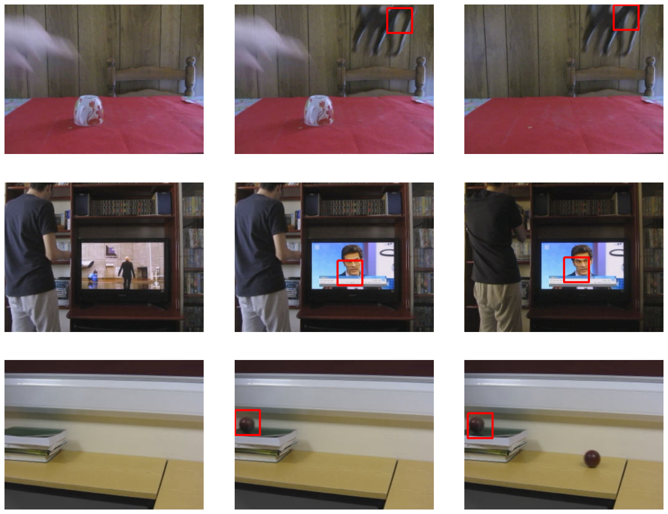


Fig. 6: Examples of block duplication detection for 3 sequences. Original (left), forged (middle), and frames with duplicated blocks (right). The red shape highlights a detected duplication.

B. Results on video-based attack

To test the detector against the video-based attack, we used the described dataset of 80 sequences. Since even this detector final decision is based on a threshold, Figure 5b shows the ROC curves for the detector tested only on the not re-compressed sequences and tested on all the sequences. Also in this case, we notice that introducing a compression step after the forgery, makes the detection more difficult. However, the AUC remains as high as 0.91 in the worst case.

As it regards temporal tampering localization, we computed for forged sequences, if the block detected as most likely duplicated was actually a duplicated block according to the ground truth. Figure 6 shows some examples. On the left the original frames of 3 sequences, in the middle the tampered version of these frames, and on the right the frames in which the duplicated block is present. The red shape identifies the spatial boundary of the detected duplicated block. When not re-compressed sequences are concerned, the detector correctly identifies a duplicated block on 90% of the sequences. If we consider also the re-compressed videos, the percentage of correctly detected blocks decreases to 87%. It is worth to notice that the tampering localization fails only when the duplicated block is not correctly identified.

Moreover, since the tampered sequences used in this paper often present small forged areas, the global correlation proposed in [14] that is less time consuming than our algorithm often fails. For this reason a comparison on these sequences would not be fair.

V. Conclusions

In this paper we presented an algorithm for local tampering detection and localization in video sequences, taking into consideration two different kinds of attacks. We also validated our algorithm on realistic forged video sequences, in order to prove that it achieves high accuracy in a real working scenario.

Some of the forged sequences that we used have been produced by us, the others come from the SULFA dataset presented in [15]. Since we recognize that producing realistic forgeries may be time consuming, we contributed to this public dataset, releasing our forged sequences too.

Future works may target the detection of other possible attacks. As an example, we should consider more complex video inpainting techniques. Moreover, we should study the possibility of developing antiforensics techniques that aims to reduce the detector accuracy.

Acknowledgment

This work was supported by the REWIND Project funded by the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme (FP7) of the European Commission, under FET-Open grant number: 268478.

References

[1] R. Poisel and S. Tjoa, “Forensics investigations of multimedia data: A review of the state-of-the-art,” in *IT Security Incident Management and IT Forensics (IMF), 2011 Sixth International Conference on*, 2011.

[2] H. T. Sencar and N. Memon, *Overview of State-of-the-art in Digital Image Forensics, Part of Indian Statistical Institute Platinum Jubilee Monograph series titled 'Statistical Science and Interdisciplinary Research,'*. World Scientific Press, 2008.

[3] A. Piva, “An overview on image forensics,” *ISRN Signal Processing*, vol. 2013, p. 22, 2013.

[4] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, “An overview on video forensics,” *APSIPA Transactions on Signal and Information Processing*, vol. 1, p. e2, 2012.

[5] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, “Multiple compression detection for video sequences,” in *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, 2012.

[6] P. Bestagini, A. Allam, S. Milani, M. Tagliasacchi, and S. Tubaro, “Video codec identification,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[7] D. Vazquez-Padin, M. Fontani, T. Bianchi, P. Comesana, A. Piva, and M. Barni, “Detection of video double encoding with GOP size estimation,” in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2012.

[8] M. Visentini-Scarzanella and P. L. Dragotti, “Video jitter analysis for automatic bootleg detection,” in *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, 2012.

[9] P. Bestagini, M. Visentini-Scarzanella, M. Tagliasacchi, P. Dragotti, and S. Tubaro, “Video recapture detection based on ghosting artifact analysis,” in *2013 IEEE International Conference on Image Processing (ICIP)*, 2013.

[10] P. Bestagini, S. Battaglia, S. Milani, M. Tagliasacchi, and S. Tubaro, “Detection of temporal interpolation in video sequences,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[11] C.-C. Hsu, T.-Y. Hung, C.-W. Lin, and C.-T. Hsu, “Video forgery detection using correlation of noise residue,” in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, 2008.

[12] A. Subramanyam and S. Emmanuel, “Video forgery detection using HOG features and compression properties,” in *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, 2012.

[13] M. Kobayashi, T. Okabe, and Y. Sato, “Detecting forgery from static-scene video based on inconsistency in noise level functions,” *IEEE Transactions on Information Forensics and Security*, vol. 5, pp. 883–892, 2010.

[14] W. Wang and H. Farid, “Exposing digital forgeries in video by detecting duplication,” in *Proceedings of the 9th workshop on Multimedia & security*, 2007.

[15] G. Qadir, S. Yahaya, and A. T. S. Ho, “Surrey university library for forensic analysis (SULFA) of video content,” in *IET Conference on Image Processing (IPR 2012)*, 2012.