VIDEO RECAPTURE DETECTION BASED ON GHOSTING ARTIFACT ANALYSIS

P. Bestagini[♯], M. Visentini-Scarzanella[♭], M. Tagliasacchi[♯], P. L. Dragotti[♭], S. Tubaro[♯]

[♯]Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, P.zza Leonardo da Vinci 32, 20133, Milano, Italy
[♭] CSP Group, EEE Department, Imperial College London
Exhibition Road, London SW7-2AZ, United Kingdom

ABSTRACT

Video forensics is becoming a popular field of research and an increasing number of forensic techniques have been proposed in the last few years. However, a simple yet effective method to fool many detectors consists in recapturing a video sequence with a camcorder. For this reason being able to detect video recapture is a topic of interest for a forensic analyst. In this paper, we first characterize the video recapture model, focusing on the common scenario of a sequence recaptured from a LCD monitor using a digital camcorder, then we propose a recapture detector for this case. The detector is based on the analysis of a characteristic ghosting artifact left by the recapture process. The presented algorithm is finally validated by means of tests on original and recaptured sequences. These tests prove that the algorithm achieves high accuracy results.

Index Terms— video forensics, recapturing, ghosting

1. INTRODUCTION

With the increasing number of multimedia sharing platforms, a growing number of video sequences is routinely acquired and uploaded for general diffusion on the Internet. However, automated mechanisms for checking the legitimacy of the uploaded content have not progressed at the same pace. Given the availability of powerful video editing software and the possibility of easily duplicating digital videos, in the last few years the need of determining the authenticity of a video sequence has become more urgent. To this end, an increasing number of techniques has been proposed in the expanding field of video forensics [1].

Among these techniques, proposed methods generally fall into two broad categories: techniques originally developed for images and applied frame-wise to videos [2], and algorithms specifically tailored to video sequences [3]. Belonging to the latter category are methods that rely on the detection of characteristic footprints left by video processing operations, such as the temporal correlation between encoded frames [4, 5]. However, these algorithms can be circumvented by hiding or removing such footprints. A well-known method to achieve this goal is to recapture the tampered video or image from a monitor [6]: in doing so, the sensitive numerical details relative to the coding and processing steps are irretrievably scrambled, and detectors relying on these footprints may fail and classify such tampered data as authentic.

The project REWIND acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number:268478.

The possibility of distinguishing between original and recaptured sequences is then of great help for a forensic analyst: a positive recapture test for a video sequence is a strong indicator of tampering activity having taken place. To this end, several methods have been proposed in the literature. An image recapture detector is described in [7], where the authors are able to automatically identify the devices used for first and second capture when sharp edges are present in the image. A detector of screenshots from interlaced videos is presented in [8]. In [9], the authors detect video recapture exploiting the Photo Response Non-Uniformity (PRNU). In [10], projected videos recaptured with a camera placed off-axis with respect to the screen are identified by detecting inconsistencies in the camera intrinsic parameters. In [11], the authors show how to detect whether a sequence has been recaptured by analyzing the high-frequency jitter introduced by, e.g., a handheld camcorder.

In this paper, we propose a novel video recapturing technique targeting the more difficult scenario of a video displayed on a Liquid Crystal Display (LCD) monitor and recaptured by a camcorder perfectly aligned with the monitor and kept fixed on a tripod. In order to detect whether a sequence has been recaptured, we exploit a characteristic ghosting artifact that is generated as a consequence of the lack of synchronization between the camera and the monitor. To the best of the authors' knowledge, this is the first work analytically describing the generative process behind ghosting artifacts, and to exploit them for automatic video recapture detection. The robustness of the proposed method is tested on a dataset of original and recaptured sequences.

The rest of the paper is organized as follows. In Section 2, the mathematical model of the recapture process and its relation to ghosting artifacts are characterized. In Section 3, our detection algorithm is presented. The evaluation results of the algorithm are presented in Section 4, while in Section 5 we draw our conclusions and discuss potential future avenues of research.

2. MODEL

In order to develop a recapture detector, we first analyze the recapture process, highlighting the configurations of practical importance. Focusing on these configurations, we mathematically characterize the generative process behind the ghosting artifacts, which constitutes the theoretical foundation of our proposed detector.

2.1. Recapture Model

Let us consider the recapturing setup depicted in Figure 1. An original video sequence $\mathbf{X} = \{X_i\},\ i = 1, ..., I$, with a nominal frame rate of $F_S = 1/T_S$ is displayed on an LCD monitor with a refresh

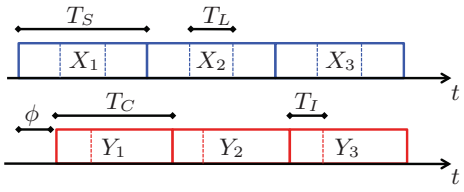


Fig. 1: Recapturing model showing frames X_i on a monitor (top) and recaptured frames Y_j (bottom). T_S is the nominal original frame length, which is multiple of the screen refresh time T_L, T_I is the integration time, T_C is the recaptured frame length, and ϕ is the phase term.

rate of $F_L = 1/T_L$. The sequence is recaptured with a camcorder to obtain $\mathbf{Y} = \{Y_j\},\ j = 1, ..., J$, with a frame rate of $F_C = 1/T_C$. Each frame Y_j is obtained by integrating the light that hits the camera sensor over a time window T_I. Since the monitor and the camera are not synchronized, the start of the recapturing process is delayed by a phase term ϕ.

In order to fully understand the practical implications of our problem formulation, let us consider inner workings of the monitor and camera system. First of all, based on current technologies it is possible to approximate the functional behavior of an LCD display as a system in which all the pixels are simultaneously refreshed at intervals of T_L, with negligible switching time for our purposes. Because of this, no evident recapture footprints such as Cathode Ray Tube scan-line artifacts are generated at refresh time. Moreover, in order to display the video at the correct frame-rate, the condition $F_L \pmod{F_S} = 0$ must in principle be verified. Indeed, the length of a displayed video frame T_S should match a multiple of the refresh time T_L. When this condition is not verified, the displayed sequence must be edited using either frame-interpolation or frame-repetition. Both techniques however alter the temporal correlation of the sequence introducing detectable artifacts. Detecting recapturing when $F_L \pmod{F_S} = 0$ is then more difficult, since there are no additional artifacts to rely on.

Regarding the camera integration window T_I, this can in principle be set arbitrarily. However, some general practical rules have to be observed: if T_I is too high, moving objects within the scene may be affected by motion blur; if T_I is too low, the resulting video will appear less smooth and also noisy, since the equivalent ISO setting of the sensor needs to be increased to capture enough light. Since low-end cameras typically do not allow the user to set this parameter, a general rule of thumb is that $T_I = T_C/2$. However, the value of T_I does not negatively affect the validity of our model or the performance of our detector.

Bearing this in mind, the j-th camcorder frame Y_j depends on what is displayed on the LCD during the integration window T_I. More specifically, we can distinguish two cases:

1. If the integration window falls either completely within the bounds of the i-th frame or between two frames of the original sequence, which are identical because of frame repetition (i.e, between X_i and $X_{i+1} = X_i$), then $Y_j = X_i$.

2. If the integration window covers the transition between two frames X_i and X_{i+1}, then Y_j is a weighted average between the two displayed frames. In particular, if within T_I the frame X_i is kept for a time T_I^0, and the frame X_{i+1} is kept for T_I^1, such that $T_I = T_I^0 + T_I^1$, the resulting camera frame can be expressed as $Y_j = \alpha X_i + (1-\alpha)X_{i+1}$, where $\alpha = T_I^0/T_I \in (0:1)$. In this case, a ghosting artifact is present on Y_j (as shown in Figure 3b).

Depending on the relative magnitude of the parameters F_S and F_C, the ghosting artifact may affect the recaptured sequence differently, as shown in Figure 2. We will now show how all cases are either of no practical relevance or can be easily solved, relying on previously proposed methods, with the exception of the scenario where $F_C = F_S$.

If $F_C \ll F_S$ (Figure 2a), the recaptured sequence consists solely of heavily ghosted frames, and some frames from the original sequence are skipped. In this scenario, $\mathbf{Y}$ is too distorted to be used as a copy of $\mathbf{X}$ by a malicious user. Conversely, in order to have $F_C \gg F_S$ (Figure 2b), the used camera should be either a high-end, high-speed camera, or the video sequence should have a very low frame-rate, which are unlikely conditions. Moreover, such a recaptured sequence would consist of many consecutive repeated frames. If $F_C \simeq F_S$ (Figures 2c and 2d), $\mathbf{Y}$ consists of a mixture of ghosted frames and original frames. The occurrence of ghosted frames forms a pattern, with a period depending on the relationship between F_S and F_C. Since ghosted frames are interpolated versions of original frames, and this interpolation appears periodically in the sequence, these sequences can be detected using an interpolation detector such as the one proposed in [12].

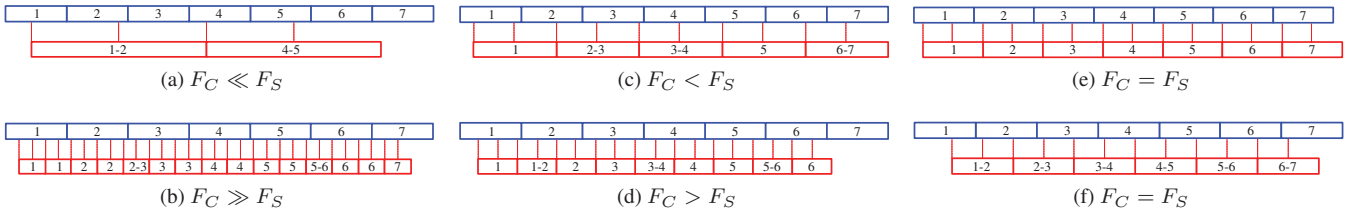
The only case of practical importance without a known solution is therefore when $F_C = F_S$. In this study, we propose a method that focuses on this challenging scenario. Indeed, depending on the time delay ϕ, the time window T_I may fall either within a single frame window X_i (Figure 2e) or between two consecutive frames X_i and X_{i+1} (Figure 2f). In the first case, we obtain $\mathbf{Y} = \mathbf{X}$, which is the only situation in which nothing can be detected. In the second scenario, the ghosting artifact is present on all recaptured frames with the same magnitude, without any non-ghosted frames perfectly recaptured from the input signal being available.

2.2. Ghosting Artifact Model

The presence of the ghosting artifact is the footprint we use to detect recapturing. In order to exploit it, we model the artifact as the outcome of a filtering operation. Let us define a mapping between the indexes i and j of original and recaptured frames respectively, such that $i = j$ and $Y_i = \alpha X_i + (1-\alpha)X_{i+1}$, where α represents the relative proportion of X_i within the integration window T_I. This is possible since we only consider the case $F_C = F_S$. Let us then consider the k-th object in X_i (e.g., the rectangle in Figure 3a) whose position moves according to the motion of a characteristic point $\mathbf{p}_i^k = (m_i^k, n_i^k)$. If the object moves in time such that it appears in $\mathbf{p}_{i+1}^k = (m_{i+1}^k, n_{i+1}^k)$ in X_{i+1} we can compute the motion vector $\mathbf{v}_i^k = \mathbf{p}_{i+1}^k - \mathbf{p}_i^k$. If we capture the screen during the transition between X_i and X_{i+1}, in the recaptured frame Y_i we observe a ghosted version of the object (e.g., the ghosted rectangle in Figure 3b). Indeed, we are averaging two pictures of the same object at two different positions related by the motion vector $\mathbf{v}_i^k$. If the motion can be locally approximated by a translation (e.g., small motion with respect to F_C), and we consider a patch P_i^k in the neighborhood of $\mathbf{p}_i^k$, the same ghosted patch can be obtained by convolution as

$$\hat{P}_i^k = P_i^k * H_i^k(\alpha), \tag{1}$$

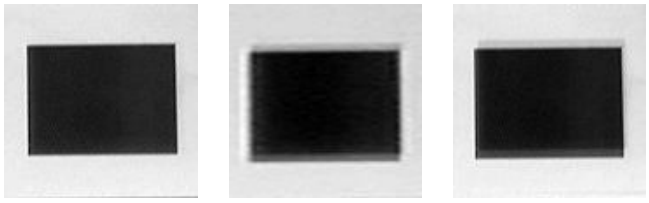
where $H_i^k(\alpha)$ is a two-dimensional filter composed by two Dirac pulses: the first pulse of magnitude α in the origin of the filter (i.e, $(0,0)$) takes into account the contribution of the patch centered in $\mathbf{p}_i^k$; the second one of magnitude $(1-\alpha)$ is located at $\mathbf{v}_i^k$, and takes into account the contribution of the patch in $\mathbf{p}_{i+1}^k$. Figure 3c shows the result of the convolution $P_i^k * H_i^k(\alpha)$.



(a) $F_C \ll F_S$ (c) $F_C < F_S$ (e) $F_C = F_S$

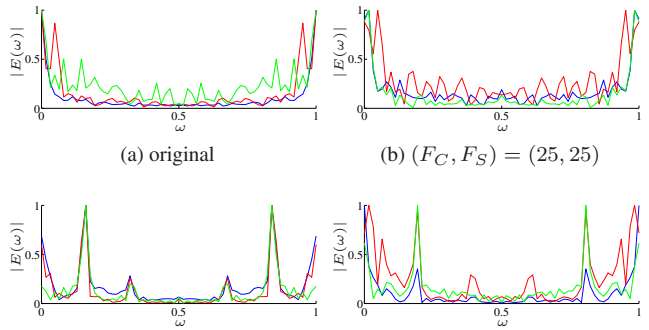
(b) $F_C \gg F_S$ (d) $F_C > F_S$ (f) $F_C = F_S$

Fig. 2: Different recapturing cases. In each figure the top row (blue) represents frames on the monitor, and the bottom row (red) frames after recapturing. Numbers show the relation between original and recaptured frames.



(a) original (b) recaptured (c) synthesized

Fig. 3: Close-up view of a patch P_i^k in an original frame X_i (a), in the recaptured frame $\alpha X_i + (1-\alpha)X_{i+1}$ (b), and its synthesized version $\hat{P}_i^k = P_i^k * H_i^k(\alpha)$ (c).



(a) original (b) $(F_C, F_S) = (25, 25)$

(c) $(F_C, F_S) = (30, 25)$ (d) $(F_C, F_S) = (25, 30)$

Fig. 4: $|E(\omega)|$ for three different sequences in the original case (a), recaptured case with $(F_C, F_S) = (25, 25)$ (b), recaptured case with $(F_C, F_S) = (30, 25)$ (c), and recaptured case with $(F_C, F_S) = (25, 30)$ (d). Peaks at high-frequencies in the last two cases means that a detector for periodic artifacts can be applied.

3. DETECTOR

Our detector targets the case $F_C = F_S$. For this reason we first need to remove the sequences affected by periodic ghosting (i.e., recaptured with $F_C \simeq F_S$) from the dataset. This can be done using the video interpolation detector proposed in [12]: first we build an estimate $\tilde{\mathbf{Y}}$ of the sequence $\mathbf{Y}$ computing its frames as $\tilde{Y}_i = (Y_{i-1} + Y_{i+1})/2$, then we compute the frame-wise error e_i summing over every pixel of $|Y_i - \tilde{Y}_i|^2$, and analyze the periodicity of this error. Figure 4 shows that in the Fourier transform $E(\omega) = \mathcal{F}(e_i)$ peaks are present on three test sequences synthetically recaptured using both $(F_C, F_S) = (25, 30)$ and $(F_C, F_S) = (30, 25)$, while they do not show up in the original sequences and in the recaptured ones using $(F_C, F_S) = (25, 25)$.

Assuming that sequences with periodic ghosting have been detected, we can run the detector for the most unfavorable scenario (i.e., $F_C = F_S$ and ghosting artifact not periodic). Our algorithm aims to blindly estimate the presence of ghosting due to frame interpolation. This kind of ghosting in the neighborhood of a patch $\hat{P}_i^k$ can be described by a convolution of a clean patch P_i^k (i.e, with no ghosting) with the ghosting filter as described in the previous section. Since the shape of this filter (i.e., the position of the delta pulses) only depends on the motion between adjacent frames, the method consists in testing the hypothesis that ghosting is present in $\hat{P}_i^k$. This can be done by: i) estimating $H_i^k(\alpha)$ from motion vectors for each key-point; ii) exploring the search space $\alpha \in (0, 1)$ to verify that such a filter can really have generated the artifact in $\hat{P}_i^k$.

The first step consists in estimating the motion vectors. For each frame Y_i a set of key-points is extracted using the method described in [13]. This method works with a multi-scale approach and returns information about the cornerness and the degree of anisotropism of every key-point. Using these values we select only the key-points that are well approximated by right angle corners. For each one, we select a patch $\hat{P}_i^k$ and search for the matching patch in Y_{i+1} to compute the motion vector $\mathbf{v}_i^k$. From this motion vector we estimate the shape of the filter $H_i^k(\alpha)$ associated to the possible ghosting artifact in $\hat{P}_i^k$.

The second step consists in verifying if $\hat{P}_i^k$ is actually affected by ghosting. To do that we select the corner in a set of templates $\mathcal{T}$, that best matches each patch as

$$T_i^k = \underset{T}{\operatorname{argmax}} \left(\operatorname{corr}(\hat{P}_i^k, T)\right),\ T \in \mathcal{T}, \tag{2}$$

where the set $\mathcal{T}$ is composed of binary patches representing right angle corners with different rotations, and $\operatorname{corr}(\cdot, \cdot)$ represents normalized cross-correlation. Notice that the use of a template is necessary because we do not have any information about the original sequence $\mathbf{X}$ and the related patches not affected by ghosting.

The next step consists in evaluating whether there is a value of α such that $\hat{P}_i^k$ can be approximated by a template corner T_i^k with a ghosting artifact caused by the convolution with $H_i^k(\alpha)$. For this purpose we compute a $\hat{P}_i^k$ approximation as

$$\overline{P}_i^k(\alpha, \beta) = T_i^k * H_i^k(\alpha) * B_i^k(\beta), \tag{3}$$

where the operator $*$ indicates convolution, $B_i^k(\beta)$ is a 2D oriented filter that introduces a motion blur of length β in the direction pointed by $\mathbf{v}_i^k$ with $\beta \in (0, |\mathbf{v}_i^k|]$, and $\alpha \in [0, 1]$. The blur is applied because corners in the original video $\mathbf{X}$ may suffer from a slight motion blur effect, while the templates that we test are perfect corners. A 2D cost function for each patch is evaluated as

$$J_i^k(\alpha, \beta) = \max\left(\operatorname{corr}\left(\hat{P}_i^k, \overline{P}_i^k(\alpha, \beta)\right)\right). \tag{4}$$

For these patches we search for the α and β values that maximize the cost function

$$(\hat{\alpha}_i^k, \hat{\beta}_i^k) = \underset{\alpha,\beta}{\operatorname{argmax}}\, J_i^k(\alpha, \beta). \tag{5}$$

If $\hat{\alpha}_i^k = 0$ or $\hat{\alpha}_i^k = 1$, $H_i^k(\alpha)$ consists of only one pulse (i.e., the other one is set to zero). This means that no ghosting affects the patch. On the other hand, if $\hat{\alpha}_i^k$ assumes any other value, we can assume that ghosting is present.

In order to detect ghosting we define

$$\begin{aligned} N_0^i &= \left| \left\{ \hat{P}_i^k : J_i^k(\hat{\alpha}_i^k, \hat{\beta}_i^k) > th \wedge (\hat{\alpha}_i^k = 1 \vee \hat{\alpha}_i^k = 0) \right\} \right|, \\ N_1^i &= \left| \left\{ \hat{P}_i^k : J_i^k(\hat{\alpha}_i^k, \hat{\beta}_i^k) > th \wedge (\hat{\alpha}_i^k \neq 1 \wedge \hat{\alpha}_i^k \neq 0) \right\} \right|, \end{aligned} \tag{6}$$

which represent the number of features in a frame Y_i whose J_i^k values are greater than the threshold th (i.e., that are well approximated by the filtered template), and that indicates the absence (N_0^i) or the presence (N_1^i) of ghosting. The threshold is selected as a percentage of the maximum achievable value of J_i^k (i.e., the maximum value of the auto-correlation of $\hat{P}_i^k$). Each frame is marked as ghosted if $N_1^i \geq N_0^i$. Since ghosting depends on the motion of the sequence, it may happen that a recaptured sequence does not show a perceivable ghosting on all its frames. For this reason the ratio N between the number of ghosted frames over a windows of W frames is computed as

$$N = \frac{|\{i : N_1^i \geq N_0^i\}|}{W}. \tag{7}$$

The final decision is then taken by thresholding the value N with the threshold $\overline{N}$, such that the sequence is considered recaptured if $N \geq \overline{N}$.

4. RESULTS

In order to test the proposed method, we prepared a dataset of original and recaptured sequences[1] (Figure 5). The dataset is composed of 18 sequences (9 original, 9 recaptured) with resolution 960×540 shot at $F_S = 25$ Frames Per Second (FPS). The recaptured sequences have been generated by displaying the original ones on a LCD monitor running at $F_L = 75$ Hz, and recapturing them at $F_C = 25$ FPS using a ϕ value that ensures the presence of ghosting and $T_I = 1/50$ seconds (i.e., case depicted in Figure 2f). Recaptured sequences have been downsized to a resolution of 960×540 pixels to match the original sequences.

The detector has been tested on all these sequences using $W = 30$. This is done to ensure the presence of frames affected by ghosting due to motion. The value N has been computed for each sequence. In order to compute the threshold used on N for recapturing detection, we performed a cross-validation analysis. This is done by using 16 sequences (8 sequences in both original and recaptured versions) as training set, and the remaining 2 sequences (1 original sequence and its recaptured version) as test set. This procedure is cycled over all the possible 9 configurations. In each case, the threshold has been computed as

$$\overline{N} = \frac{\mu_{orig} + \mu_{rec}}{2}, \tag{8}$$

where μ_{orig} is the average N for original sequences, and μ_{rec} is the average N for recaptured ones. Table 1 shows the values of μ_{orig}, μ_{rec}, the standard deviations σ_{orig} and σ_{rec}, the threshold $\overline{N}$, and accuracy values for three cases: i) the one with the maximum accuracy; ii) that with minimum accuracy; iii) and the average between all the 9 test cases. Notice that even in the worst case (i.e., that with minimum accuracy), the values μ of both recaptured and original classes indicates that recaptured sequences has an average of almost 80% of frames detected as recaptured, while original sequences has more or less only the 20%. Moreover, the threshold $\overline{N}$ is almost always set near 50%, which means that a sequence is classified as recaptured if ghosting is detected on the majority of frames.

[1]Preview available at: https://www.youtube.com/watch?v=1E2-8Dtu648.

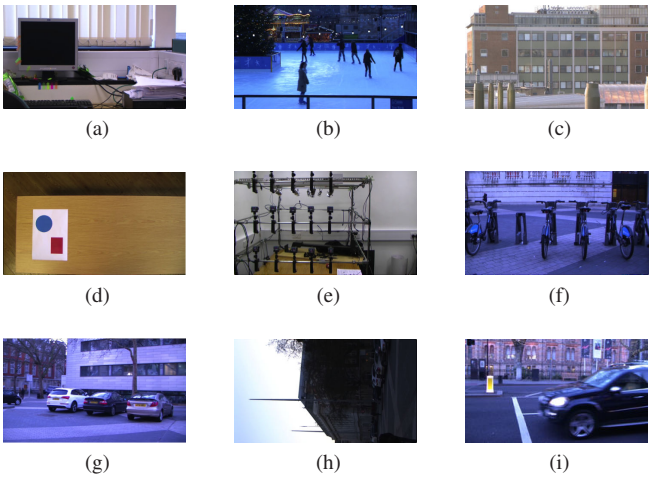


Fig. 5: Sample frames from the original tested sequences.

	μ_{orig}^{train}	μ_{rec}^{train}	σ_{orig}^{train}	σ_{rec}^{train}	$\overline{N}$	**Acc.**
max	0.18	0.81	0.14	0.26	0.49	**0.94**
min	0.17	0.77	0.14	0.25	0.47	**0.89**
mean	0.17	0.78	0.13	0.24	0.48	**0.91**

Table 1: Mean and standard deviation of N values of original and recaptured sequences in training set. The selected threshold $\overline{N}$ and the achieved accuracy are also shown. All the parameters are shown in three cases: i) maximum accuracy achieved (top row); ii) minimum accuracy achieved (middle row); average value among all the cases (bottom row).

5. CONCLUSIONS

In this paper we characterized the video recapture process and developed a detector for a case that is still not addressed in the literature. The detector is based on the analysis of characteristic ghosting artifacts left during the recapturing process. In particular, since ghosting can be modeled as the effect of a filtering operation, our novel approach exploits the motion between frames to retrieve information about the ghosting filter (i.e., the shape), and verifies if such a filter may have been used on the frames. If the answer is positive, the frames are considered recaptured. The detector is validated on a dataset of original and recaptured sequences, achieving a high level of accuracy.

This method contributes theoretically and algorithmically to the widening panorama of digital forensics. Further analysis could be devoted to the exact quantification of the phase shift between camera and monitor. Another interesting direction is the multimodal analysis of video and audio streams to further increase the detector robustness.

6. REFERENCES

[1] 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, pp. e2, 2012.

[2] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, "Source digital camcorder identification using sensor photo-response nonuniformity," in *Proc. of SPIE Electronic Imaging, Photonics West*, 2007.

[3] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, "Multiple compression detection for video sequences," in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*, sept. 2012, pp. 112 –117.

[4] 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)*, March 2012, pp. 2257 –2260.

[5] 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)*, December 2012.

[6] H. Cao and A.C. Kot, "Identification of recaptured photographs on LCD screens," in *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, March 2010, pp. 1790 –1793.

[7] T. Thongkamwitoon, H. Muammar, and P.L. Dragotti, "Identification of image acquisition chains using a dictionary of edge profiles," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, August 2012, pp. 1757 – 1761.

[8] J.-W. Lee, M.-J. Lee, T.-W. Oh, S.-J. Ryu, and H.-K. Lee, "Screenshot identification using combing artifact from interlaced video," in *Proceedings of the 12th ACM workshop on Multimedia and security*, New York, NY, USA, 2010, MM&Sec '10, pp. 49–54, ACM.

[9] D.-J. Jung, S.-J. Hyun, D.-K.and Ryu, J.-W. Lee, H.-Y. Lee, and H.-K. Lee, "Detecting re-captured videos using shot-based photo response non-uniformity," in *Proceedings of the 10th international conference on Digital-Forensics and Watermarking*, Berlin, Heidelberg, 2012, IWDW'11, pp. 281–291, Springer-Verlag.

[10] W. Wang and H. Farid, "Detecting re-projected video," in *Information Hiding*, Kaushal Solanki, Kenneth Sullivan, and Upamanyu Madhow, Eds., vol. 5284 of *Lecture Notes in Computer Science*, pp. 72–86. Springer Berlin Heidelberg, 2008.

[11] 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)*, September 2012, pp. 101 –106.

[12] 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)*, May 2013.

[13] S. Giannarou, M. Visentini-Scarzanella, and G.-Z. Yang, "Probabilistic tracking of affine-invariant anisotropic regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 130 –143, January 2013.