

A video forensic technique for detecting frame deletion and insertion

Original

A video forensic technique for detecting frame deletion and insertion / A., Gironi; M., Fontani; Bianchi, Tiziano; A., Piva; M., Barni. - (2014), pp. 6226-6230. (2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Firenze May 2014) [10.1109/ICASSP.2014.6854801].

Availability:

This version is available at: 11583/2560955 since:

Publisher:

IEEE - INST ELECTRICAL ELECTRONICS ENGINEERS INC

Published

DOI:10.1109/ICASSP.2014.6854801

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

A VIDEO FORENSIC TECHNIQUE FOR DETECTING FRAME DELETION AND INSERTION

A. Gironi^{*} M. Fontani[†] T. Bianchi[#] A. Piva^{*} M. Barni[§]

^{*} Dept. of Information Engineering, Università di Firenze, Firenze (IT)

[†] CNIT, Università di Siena, Siena (IT)

[#] Dept. of Electronic and Telecommunications, Politecnico di Torino, Torino (IT)

[§] Dept. of Information Engineering and Mathematical Sciences, Università di Siena, Siena (IT)

ABSTRACT

We propose a method for detecting insertion and deletion of whole frames in digital videos. We start by strengthening and extending a state of the art method for double encoding detection, and propose a system that is able to locate the point in time where frames have been deleted or inserted, discerning between the two cases. The proposed method is applicable even when different codecs are used for the first and second compression, and performs well even when the second encoding is as strong as the first one.

Index Terms— Video Forensics, Video Splicing, Video Tampering, Frame Deletion, Frame Addition, Frame Removal.

1. INTRODUCTION

Digital videos (DV) are being used everyday for security purposes, and they are usually considered a more reliable source of evidence than still images. This fact is probably motivated by the considerable skills that are needed to produce a forged video, since introducing and removing objects from videos proves hard to do. On the other hand, in many cases the meaning of a video can be distorted by simply removing, replicating or inserting a group of frames. For example, such an attack proves to be extremely dangerous in contexts like video surveillance, where eliminating a group of frames can make the video totally useless.

In the last years, Video Forensics has emerged as a discipline aiming at investigating the processing history of DVs in a blind fashion [1]. Since DVs are commonly stored in a compressed form, several works have been presented for detecting whether a video has been encoded twice [2, 3, 4] or even multiple times [5]. Although being very interesting from a scientific point of view, this task does not help much in assessing the reliability of the content, since multiple compressions do not necessarily imply a video forgery.

Video Forensics also studies the integrity of DVs, trying to expose manipulations; focusing on this, it is helpful to distinguish between *intra-frame* forgeries, where the attacker alters the content of single frames (e.g. by introducing or removing objects), and *inter-frame* forgeries, where (group of) frames are entirely deleted, inserted or even replicated. Being very different, these attacks must be investigated using distinct techniques. In the literature, detection of intra-frame forgery has been investigated by adapting methods developed for images [6, 7, 8]. On the other hand, the problem addressed in this paper, that is inter-frame forgery detection, has

been faced in the literature by exploiting the frame grouping strategy adopted by all common video encoders [9, 6, 10].

The method we propose in this paper relies on the fact that creating a forged video always implies a double compression, with the first encoding being performed during acquisition and the second after tampering. Therefore, we adopt a state of the art method for double encoding detection [2] and we modify it so as to make it robust to frame removal between the two encodings; then, we design an algorithm that iteratively uses this improved method to detect whether a misalignment in the frame structure of the video occurred between the two encodings. If this is the case, we detect the point where the misalignment occurs and also classify the attack distinguishing between frame deletion and insertion. As we show in the experimental validation, the proposed approach has the advantage of being able to work when different coding algorithms are used in the first and second compression, a situation that is very common when dealing with DVs. To the best of our knowledge, this aspect was only partially investigated in [10], where different strategies for GOP structuring were simulated. Furthermore, acceptable performance are retained even when the second compression is stronger than the former. Finally, the method is computationally convenient, since a video can be analyzed even without going through the decoding chain.

In order to provide a clear explanation of our contribution, we first introduce some basic concepts of video coding, then we briefly explain the adopted method for double encoding detection, and finally we propose our approach for inter-frame forgery localization. A set of experiments are carried out to investigate the robustness of the approach under a wide range of conditions.

2. BASIC CONCEPTS OF VIDEO CODING

A digital video is basically a sequence of still pictures shot at a sufficiently high rate (typical values are 25 or 30 frame per seconds). The resulting signal can be conveniently compressed by reducing both spatial and temporal redundancy. In order to do that, common video coding algorithms like MPEG-2 [11], MPEG-4 [12] and H.264 [13] employ a block-based hybrid video coding approach, and divide pictures into different types: intra-coded pictures, referred to as I-frames, and predictive-coded pictures, commonly named P-frames and B-frames. During encoding, frames are grouped in GOPs (group of pictures) according to a structure that always starts with an I-frame and then allows a certain number of predictive frames. The total number of frames composing a GOP is called GOP size. In this paper we focus on fixed-GOP encoding, where the GOP structure and size are kept constant. When encoding a frame, the encoder divides it in macroblocks (MBs) and codes each MB separately: MBs belonging to I-frames are always encoded without making reference to

This work was partially supported by the REWIND Project funded by the Future and Emerging Technologies (FET) programme within the 7FP of the European Commission, under FET-Open grant number 268478.

other frames, while MBs belonging to predictive-coded frames may also be encoded making reference to previous frames (this is the only possibility in P-frames) or even future frames (allowed in B-frames). In the following, we will refer to MBs that are encoded without temporal prediction as intra-coded, and denote them as I-MBs; similarly, we will write P-MBs for those MBs that are encoded making reference to other frames. Finally, the encoder has the possibility to skip a MB, if this MB can be directly copied from a previous frame: these MBs are denoted as S-MBs. For a more detailed description of video coding the reader may refer to [14].

3. DOUBLE ENCODING DETECTION

Recently, the Variation of Prediction Footprint (VPF) has been presented as a valid tool for detecting whether a video has been encoded twice [2]. Since the method we propose builds on this footprint, we provide here a brief explanation of it, while details can be found in the original paper.

Suppose a video is encoded twice using a fixed GOP size G_1 for the first encoding and a fixed GOP G_2 for the second encoding, and that only I- and P- frames are used. Authors of [2] observed that when a frame originally encoded as intra is re-encoded as a P-frame, an anomalous decrease in the number of S-MBs occurs, together with an increment in the number of I-MBs. According to the notation in [2], we denote with $i(n)$ the number of intra-coded MBs used within the n -th frame, $n \in \{0, \dots, N-1\}$ with N being the total number of frames. Similarly, we denote with $s(n)$ the number of skipped MBs. Since in I-frames only I-MBs are allowed, the signal $i(n)$ presents strong peaks at multiples of G_2 ; these peaks are replaced by the average between the previous and following elements of the signal; the same is done for $s(n)$. Then, the set \mathcal{P} is defined as containing only those frames that show, simultaneously, a higher number of I-MB and a smaller number of S-MB compared to the previous and following frames. For frames in \mathcal{P} the strength of the VPF is evaluated by summing the product of the slopes as follows:

$$v(n) = |(i(n) - i(n-1))(s(n) - s(n-1))| + |i(n+1) - i(n))(s(n+1) - s(n))|,$$

while the $v(n)$ is defined to be 0 for frames not in \mathcal{P} .

The signal $v(n)$ is expected to show peaks in correspondence of those P-frames that are the re-encoded version of a previous I-frame; if G_1 was fixed, this results in *periodic* peaks in $v(n)$. Therefore, authors of [2] propose to search for the presence of such a periodicity in $v(n)$ so as to detect double encoding and, simultaneously, estimate G_1 . Classical methods like analysis in the Fourier domain are not appropriate for the case at hand, because periodicity have to be detected relying on a small number of observed periods. For this reason, authors of [2] propose to create a set \mathcal{C} of possible candidates:

$$\mathcal{C} = \{c \in \{2, \dots, N\} : \exists n_1, n_2 \in \mathcal{P}, \text{GCD}(n_1, n_2) = c\}.$$

For each element in \mathcal{C} , a composite fitness function $\phi(c) = \phi_1(c) - \phi_2(c) - \phi_3(c)$ is defined, where:

- $\phi_1(c)$ accumulates the value of $v(n)$ in integer multiples of the candidate c , formally:

$$\phi_1(c) = \sum_{i=kc} v(i), \quad \text{with } i \in \mathcal{P}, k \in \left[0, \left\lfloor \frac{N}{c} \right\rfloor\right].$$

- $\phi_2(c)$ penalizes for the absence of peaks in multiples of the candidate c :

$$\phi_2(c) = \sum_{i=kc} \beta, \quad \text{with } i \notin \mathcal{P}, k \in \left[0, \left\lfloor \frac{N}{c} \right\rfloor\right],$$

where β is the penalization factor, authors suggest $\beta = 0.1 \times \max_n \{v(n)\}$;

- $\phi_3(c)$ penalizes the presence of periodic noise, limiting to the strongest periodic component:

$$\phi_3(c) = \max_{z \in [1, c-1]} \left\{ \sum_{k=0}^{\lfloor N/z \rfloor} v(kz) \right\}.$$

As a last step, the highest value Φ obtained for $\phi(c)$ is compared to a threshold: if Φ is above the threshold, the video is classified as double encoded and G_1 is estimated as

$$\hat{G}_1 = \arg \max_{c \in \mathcal{C}} \phi(c). \quad (1)$$

4. DETECTING FRAME REMOVAL AND INSERTION

Now that we have introduced the VPF and briefly explained how it can be used to detect double encoding, we turn to show how the same effect can be exploited to detect inter-frame video tampering.

Let us suppose that a video is captured, then a set of frames is removed using some editing software, and the final video is saved (i.e., re-encoded). If we maintain the assumptions given at the beginning of Section 3, we still expect to find the VPF in the altered video but, due to the removal of frames, the peaks will no longer be periodic throughout the whole video. Instead, we expect to find a phase discontinuity located at the point where the cut took place, provided that the user did not eliminate a number of frames that is an integer multiple of G_1 . In the following, we show how to search and exploit such a discontinuity. First we introduce an improvement to the method in [2], so as to allow double encoding detection even when a group of leading frames is removed, and also estimate the number of removed frames. Then, we propose an iterative algorithm that, leveraging on the improved method, detects removal and insertion of frames throughout the video.

4.1. Increasing the robustness of the VPF

We begin by noticing that even the removal of one frame at the beginning of the video (between the two encodings) would prevent the method in [2] from working. Indeed, periodic peaks would still appear, but since the set \mathcal{C} is obtained by making use of the GCD operator, the correct value for G_1 could not be in \mathcal{C} (neglecting noisy peaks). This becomes evident if we consider that when the r leading frames are removed before re-encoding, peaks would be located at $kG_1 - r$; for $r \neq zG_1, z \in \mathbb{N}$, such numbers are not divisible by G_1 and therefore the GCD between couples of elements in this set cannot be G_1 . This limitation is clearly due to the fact that the method in [2] is explicitly thought to detect double encoding, without considering any form of frame manipulation.

In order to overcome this drawback, we propose to expose the inherent periodicity of $v(n)$ by working on its autocorrelation $R_{vv}(\tau)$, evaluated for lags $\tau = 0, \dots, N-1$. By using the very same approach for periodicity estimation presented in the previous section, but working on $R_{vv}(\tau)$ instead of $v(n)$, we obtain robustness to the removal of a set of leading frames. Furthermore, once we have an

estimate \hat{G}_1 of the period, we can also easily estimate the number of removed frames modulo \hat{G}_1 ; this quantity will be termed *shift* from now on for brevity, and it will be denoted by \hat{s} . Let us introduce the following function:

$$\psi(s) = \frac{\sum_{i=0}^{\lfloor \frac{N-s}{\hat{G}_1} \rfloor} v(i\hat{G}_1 + s)}{N}, \quad (2)$$

that is the mean of the signal $v(\cdot)$ at multiples of \hat{G}_1 displaced by s . Then, we can estimate the shift as:

$$\hat{s} = \arg \max_{s \in \{0, \dots, \hat{G}_1 - 1\}} \psi(s). \quad (3)$$

4.2. Iterative analysis for frame removal localization

The algorithm explained in Section 3 works even when the number of available frames is limited: as the authors report [2], $4 \cdot G_1$ frames are usually sufficient to obtain a correct estimate of G_1 . This fact suggests that the video could be analyzed in many sub-parts (e.g., by moving an analysis window) rather than as a whole. While this would not help in the case of a video that is just compressed twice, it would allow us to search for inconsistencies within a manipulated video. Such an approach would not be possible with the original method, since it is not invariant to frame shifting. However, by using the modified version proposed in Section 4.1, we can use a moving-window analysis to estimate the shift \hat{s} at every iteration. Let us assume the analysis window is moved by one frame at each iteration. In the case of a double compressed video (without frame removal), the shift is expected to be a periodic function of period G_1 , which decreases linearly from $G_1 - 1$ to 0 and then starts again from $G_1 - 1$, as in Fig. 1 (left). On the contrary, if a number of frames is removed that is not a multiple of G_1 , the proposed method would detect the correct value for G_1 both before and after the manipulation, but the shift would present a phase discontinuity in correspondance to the first removed frame, as in Fig. 1 (right).

Inspired by this fact, we propose the following procedure to detect these discontinuities and remove those that are due to noise. Given a video, we first compute the signal $v(n)$ as defined in Section 3. In order to get a reliable estimate of G_1 even in the presence of a manipulation, we analyze the signal $v(n)$ with a sliding window of size W , shifting it by one frame at a time. At each step we use the modified approach proposed in Section 4.1 to estimate G_1 , thus obtaining a signal $g(n)$, $n = 0, \dots, N - W$ containing the estimate of G_1 at each window position. Then, the overall estimate of G_1 for the video is defined as:

$$\tilde{G}_1 = \text{mode}(g(n)), \quad (4)$$

where $\text{mode}(\cdot)$ denotes the statistical mode of the signal. Using \tilde{G}_1 , we repeat the window-based analysis to estimate the value of the shift at each window, according to equation (3), thus obtaining the shift array $\sigma(n)$, $n = 0, \dots, N - W$ that was plotted in Figure 1. To better highlight the phase discontinuity in this signal, we remove the periodic component due to the shift of the window, and define:

$$\sigma_h(n) = \text{mod}(n + \sigma(n), \tilde{G}_1). \quad (5)$$

In the ideal case, $\sigma_h(n)$ should be a step function with value 0 until the point of the cut is reached, then it should move to the value $C \bmod G_1$, where C is the number of removed frames. Due to noise in the original signal $v(n)$, however, other peaks may be present in $\sigma_h(n)$ that are not related to manipulations. These kinds of impulsive noise can be safely mitigated by adopting a median filtering of

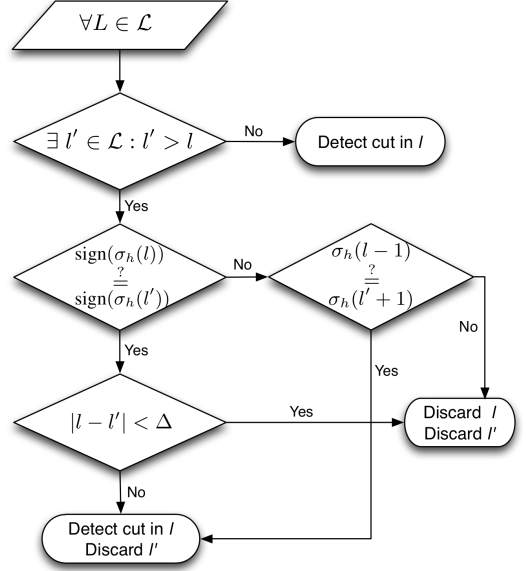


Fig. 2. Adopted algorithm for cut detection, given the set of indices where the signal $\sigma'_h(n)$ is not null.

$\sigma_h(n)$. Since we are mostly interested in where the discontinuity is located, the first order derivative $\sigma'_h(n)$ is computed from the filtered signal, and the set \mathcal{L} is defined as:

$$\mathcal{L} = \{l : \sigma'_h(l) \neq 0, l \in \{0, 1, \dots, N - W\}\}. \quad (6)$$

If the set \mathcal{L} is empty, the video is classified as not containing frame removal or insertions. If the set is not empty, a further step of analysis is carried out before classifying the video as manipulated. Indeed, we know that frame removal would cause a durable change in the step function $\sigma_h(n)$, since once the phase is broken the displacement should remain constant. This information helps us in discarding elements in \mathcal{L} that are due to noisy measurements; we propose to adopt the algorithm in Fig. 2, where the value Δ denotes the minimum allowed distance between two consecutive peaks.

4.3. Extension to localization of frame insertion

Let us now suppose that the manipulated video is obtained by inserting a group of frames coming from a different sequence, that was encoded using a constant GOP size G_1^2 different from G_1 . Under these conditions, the described algorithm is expected to detect two cutting point, one at the beginning and one at the end of the injected sequence, where the phase discontinuity changes again. Let us denote as l_1 and l_2 the frame indexes where the first and the second discontinuity are localized, respectively. In order to distinguish between two independent frame removals and a frame insertion, we can verify whether the estimate of the GOP value between l_1 and l_2 coincides with the estimate on the rest of the sequence or not. Formally, we propose to calculate:

$$\begin{aligned} \tilde{G}_1^2 &= \text{mode}(g(n)), \quad l_1 \leq n \leq l_2 \\ \tilde{G}_1^1 &= \text{mode}(g(n)), \quad n < l_1 \vee n > l_2. \end{aligned}$$

Finally, frame insertion is detected if $\tilde{G}_1^2 \neq \tilde{G}_1^1$, and \tilde{G}_1^2 is chosen as the estimate of the GOP size for the pasted sequence.

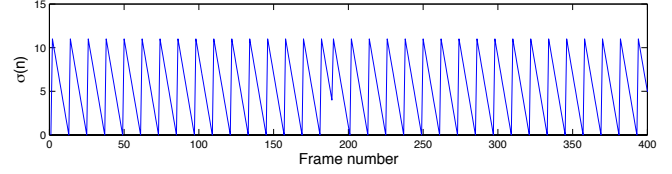
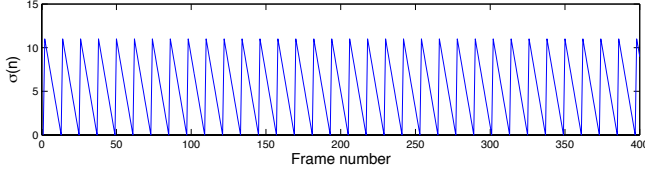


Fig. 1. An example of the shift signal $\sigma(n)$ for a double compressed sequence (left) and a manipulated sequence (right), where a group of frames were removed starting from position 190.

5. EXPERIMENTAL RESULTS

In this section we evaluate the performance of the proposed method in discriminating between double compressed and manipulated videos, and between different kinds of manipulation. Classification between single- and double-encoded videos is an embedded functionality of the method in [2], and it is not addressed here.

To generate our dataset, a group of 14 well-known YUV uncompressed videos has been downloaded¹ with CIF resolution, i.e., 352×288 pixels. Each sequence has been extended to be 1250 frames long using mirrored frames repetition, thus avoiding abrupt changes in content. Throughout our experiments, we used MPEG-2, MPEG-4 and H.264 as possible encoders to compress videos². Bitrates were taken from the set $\{100, 300, 700\}$ (CBR compression mode was used); values for G_1 were allowed to take values in $\{12, 15, 31\}$, while values for G_2 were taken from $\{10, 20, 25\}$. By taking all possible combinations of the above parameters, a total of 10206 double compressed videos were generated. To generate videos with frame deletion, each sequence was decoded after the first compression, a group of 100 frames were removed from a random point, then the second encoding was performed. This originated 10206 manipulated test sequences.

Given these two sets of videos, we run the proposed analysis, using $W = 100$, $\Delta = 50$, and using a width of 200 for the median filtering. Since the VPF is affected by the encoding settings, performance of our method also depends on them; the two most important parameters are the codec and bitrate used for the first and second compression [2]. For this reason, we plot marginalized results in Table 1 for the different couples of first/second compression codecs and compression bitrates. Accuracies were computed averaging the true negative rate and the true positive rate; a video was considered as correctly classified when the localized cutting point was no farther than W frames from the actual cutting point.

To evaluate the performance of the method for frame insertion localization, we designed the following experiment: starting from singly-compressed sequences, we selected two of them at random, decoded them, and pasted 350 frames from one sequence into the other one; finally, the spliced sequence was compressed. Since frame insertion can be detected only when the GOP sizes of the spliced sequences are different, we allowed G_1 to take values in $\{12, 15\}$ while G_2 was set to 10. The GOP size of the second encoding, that is G_2 , was chosen from $\{33, 40\}$. As stated in Section 4.3, the localization of frame insertion follows the localization of multiple cutting points. For this reason, we tested the capability of the method of distinguishing between the two manipulations. To this end, frame insertion within a video was considered as correctly localized (true positive) when the beginning of the insertion was detected no farther

C_1/C_2	MPEG-2	MPEG-4	H.264
MPEG-2	83.38 %	81.70 %	95.46 %
MPEG-4	81.83 %	79.39 %	96.25 %
H.264	76.10 %	76.19 %	88.32 %
B_1/B_2	100	300	700
100	85.44 %	89.68 %	91.27 %
300	77.95 %	86.55 %	88.80 %
700	75.93 %	81.04 %	81.94 %

Table 1. Accuracy of the method in distinguishing double compressed and manipulated (by frame-removal) videos, plotted in terms of first-vs-second coding algorithms (upper table) and first-vs-second compression bitrates (bottom table).

C_1/C_2	MPEG-2	MPEG-4	H.264
MPEG-2	77.48 %	77.91 %	89.99 %
MPEG-4	75.15 %	74.24 %	90.52 %
H.264	70.79 %	72.80 %	84.16 %
B_1/B_2	100	300	700
100	86.15 %	89.71 %	90.72 %
300	70.39 %	85.30 %	89.93 %
700	55.58 %	67.51 %	81.47 %

Table 2. Accuracy in distinguishing between videos with frame insertion and videos with frame removal.

than W frames from its actual position and G_1^2 was correctly estimated. On the other hand, true negative are obtained when no frame insertion was localized in a video in which only frame removal took place. Accuracies, obtained by averaging the true positive and true negative rates, are reported in Table 2 for different first/second employed codecs and bitrates (results are averaged across all possible encoding configuration for the pasted segment).

6. CONCLUSIONS

In this paper a method for localization of frame removal and insertion in digital videos was proposed. The key features of the scheme are: the robustness to the use of different codecs, the attainment of acceptable performance when the last compression is strong, and the possibility of distinguishing frame removal from insertion. The method is open to improvements: the localization of cutting/insertion point is not as precise as that allowed by motion-vector based schemes, and the method has not been evaluated in presence of global processing of the video, e.g. resizing of frames. Finally, we point out that the proposed method cannot detect frame manipulations when the attacker removes/inserts a whole GOP, thus re-establishing the exact periodicity of the analyzed signal.

¹Video available at <http://trace.eas.asu.edu/yuv/>, chosen sequences are: *akiyo*, *bridge-close*, *bridge-far*, *coastguard*, *container*, *foreman*, *hall*, *high-way*, *mobile*, *news*, *paris*, *silent*, *tempe*, *waterfall*.

²The libavcodec and x264 libraries were used, through FFmpeg.

7. REFERENCES

- [1] Simone Milani, Marco Fontani, Paolo Bestagini, Mauro Barni, Alessandro Piva, Marco Tagliasacchi, and Stefano Tubaro, "An overview on video forensics," *APSIPA Transactions on Signal and Information Processing*, vol. 1, 11 2012.
- [2] D. Vazquez-Padin, M. Fontani, T. Bianchi, P. Comesana, A. Piva, and M. Barni, "Detection of video double encoding with GOP size estimation," in *Information Forensics and Security (WIFS), IEEE International Workshop on*, 2012, pp. 151–156.
- [3] Junyu Xu, Yuting Su, and Qingzhong Liu, "Detection of double MPEG-2 compression based on distributions of DCT coefficients," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 01, pp. 1354001, 2013.
- [4] X. Jiang, W. Wang, T. Sun, Y.Q. Shi, and S. Wang, "Detection of double compression in MPEG-4 videos based on Markov statistics," *Signal Processing Letters, IEEE*, vol. 20, no. 5, pp. 447–450, 2013.
- [5] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, "Multiple compression detection for video sequences," in *Multimedia Signal Processing (MMSP), IEEE 14th International Workshop on*, 2012, pp. 112–117.
- [6] Yuting Su, Jing Zhang, and Jie Liu, "Exposing digital video forgery by detecting motion-compensated edge artifact," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, 2009, pp. 1–4.
- [7] Weihong Wang and Hany Farid, "Exposing digital forgeries in video by detecting double quantization," in *Proceedings of the 11th ACM workshop on Multimedia and security*, New York, NY, USA, 2009, MM&Sec '09, pp. 39–48, ACM.
- [8] D. Labartino, T. Bianchi, A. De Rosa, M. Fontani, D. Vazquez-Padin, A. Piva, and M. Barni, "Localization of forgeries in MPEG-2 video through GOP size and DQ analysis," in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, 2013, pp. 494–499.
- [9] Weihong Wang and Hany Farid, "Exposing digital forgeries in video by detecting double MPEG compression," in *MM&Sec*, 2006, pp. 37–47.
- [10] M.C. Stamm, W.S. Lin, and K.J.R. Liu, "Temporal forensics and anti-forensics for motion compensated video," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 4, pp. 1315–1329, 2012.
- [11] ISO, "Information technology - generic coding of moving pictures and associated audio information - part 2: Video," ISO/IEC IS 13818-2, International Organization for Standardization, Geneva, Switzerland, 2008.
- [12] ISO, "Information technology - coding of audio-visual objects - part 2: Visual," ISO/IEC IS 14496-2, International Organization for Standardization, Geneva, Switzerland, 2009.
- [13] ISO, "Information technology - coding of audio-visual objects - part 10: Advanced video coding (avc)," ISO/IEC IS 14496-10, International Organization for Standardization, Geneva, Switzerland, 2010.
- [14] Alan C Bovik, *Handbook of image and video processing*, Access Online via Elsevier, 2010.