Tampering Detection of Audio-Visual Content using Encrypted Watermarks

Ronaldo Rigoni*, Pedro Garcia Freitas*, and Mylène C.Q. Farias*+
*Department of Computer Science,
+Department of Electrical Engineering,
University of Brasília (UnB),
Campus Universitário Darcy Ribeiro, 70919-970 Brasília, DF - Brazil
Emails: rrigoni@gmail.com, sawp@sawp.com.br, mylene@ieee.org

***Abstract*—In this paper, we present a framework for detecting tampered information in digital videos. Using the proposed technique is possible to detect several types of tampering with a pixel granularity. The framework uses a combination of temporal and spatial watermarks that do not decrease the perceived quality of the host videos. We use a modified version of Quantization Index Modulation (QIM) algorithm to store the watermarks. Since QIM is a fragile watermarking scheme, it is possible to detect local, global, and temporal tampers and also estimate the attack type. The framework is fast, robust, and accurate.**

***Keywords*-Quantization Index Modulation, Watermarking, Video Tampering Detection**

I. Introduction

The flexibility of digital images and videos is both a blessing and a curse. Digital technologies make it possible to create high quality pictures, animations, games, and special effects with an amazing realism. Digital pictures (images and videos) can be enhanced, compressed, transmitted, translated across different standards, and displayed in a variety of devices. Then, because of the significant advances in compression and transmission techniques, it is possible to deliver high quality visual content to the end user in many different ways. As a consequence, a variety of delivery services have been created in the last years, such as direct TV broadcast satellite, digital broadcast television, and IP-based video streaming.

A very important concern for video distribution applications is content protection, which involves tampering detection [1]. Several softwares are currently available for video processing, making it easy to alter (tamper) visual digital content without leaving any clear sign of these modifications. These softwares allow unauthorized users to change and illegally distribute digital video content, causing monetary and personal losses [2], [3]. As a consequence, automatic methods for checking the authenticity and integrity of digital images and videos are, undoubtedly, very important.

Several techniques have been proposed with the goal of detecting tampering of digital content [1]. These techniques can be divided in approaches that do not require the original (no-reference) and approaches that do require the reference (full reference). Since in most transmission applications the original is not available, no reference approaches are the most adequate ones for these applications.

Most of the no-reference tampering detection techniques are specialized in detecting a single type of tamper [4], [5]. Because of this limitation, it is not possible to create a software that easily detects multiple types of tamper attacks. A compromise between the no-reference and full-reference is the reduced-reference approach. This approach embeds an "invisible" information (mark) into the content using a watermarking technique. To verify if the original content was tampered, the embedded information is extracted and its integrity is verified. In this approach, the *fragility* of the embedded mark is a key element that determines the amount of tampering which the algorithm is able to detect.

There are several reduced-reference tampering detection algorithms [6], [7], [8], [9]. The work of Amerini *et al.* is based on watermarking techniques [6]. Hou *et al.* [10] proposed a spatial tampering detection technique that consists of storing a verification bit in DCT 4×4 blocks. The method is robust and causes low degradation, but it can only detect tamper in 4×4 blocks (spatial resolution) and it does not detect temporal attacks. On the other hand, Lin *et al.* [7] uses spatial and temporal redundancy to detect tampers in videos and static image sequences. Their method inserts a mark in high correlation blocks, making it robust to geometric attacks, such as rotation and scaling. Subramanyam and Emmanuel [8] and Wang and Farid [9] also propose methods for temporal tampering detection.

In this paper, we present a framework with the goal of protecting and detecting tampered information in digital videos. The proposed framework is based on a reduced-reference watermarking technique that is able to detect temporal, local, and global attacks. The algorithm is divided in two main stages. The first stage consists of a *tampering protector* that generates a mark and embeds it into the host content. The second stage is a *tampering detector* that detects tampers using the extracted mark.

The paper is divided as follows. In Section II, we review the types of tampers that the framework is able to detect. In Section III, we present the suite of techniques used to generate the mark and embed it. In Section IV, we describe

1530-1834/14 $31.00 © 2014 IEEE
DOI 10.1109/SIBGRAPI.2014.50

the watermarking extraction and tamper detection algorithms. In Section V, we present the simulation results. Finally, in Section VI, our conclusions are presented.

II. Tampering Techniques

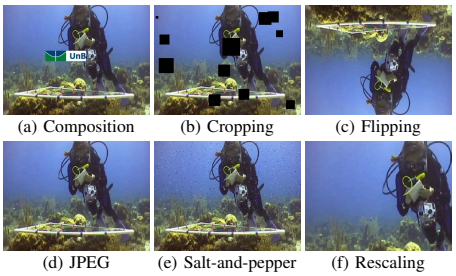
Digital tampering happens when a digital media (image, audio, video, text, etc.) is changed from its original version. These changes can be classified as intentional or unintentional. Intentional tampering has the goal of maliciously modifying the content or removing the copyright. On the other hand, unintentional tampering is a consequence of digital processing operations, such as brightness correction, format conversion, size reduction, etc. In this work, we restrict our discussion to video content. For video signals, tampering techniques can be classified as spatial and temporal changes [11], [12], [13], [7].

Spatial tampering techniques correspond to changes made to the pixels in individual frames of the video. Spatial tampering can be further classified as local or global. Local tampering corresponds to changes made to a set of pixels. Examples of local tampering include changing the color of an area, removing a block of pixels, overlapping a set of pixels, etc. Global tampering consists of modifying entire video frames. Examples include brightness adjustment, format conversion, zooming, among others. Global tampering can be applied to a single or to several video frames.

The most popular spatial tampering attacks are:

- *Composite attack*: Consists of combining two or more images to generate a new image. In Figure 1-(a), a composition is made by replacing a region of the original frame ('Diver') by another image ('Custom logo'). Note that this process may generate non-homogeneous border regions if the substitution is done in a naive way.
- *Cropping attack*: Consists of removing parts of the frame content. As showed in Figure 1-(b), this is a local tampering technique.
- *Flipping attack*: This is a global tampering technique that consists of rotating video frames. This type of attack is difficult to detect because it keeps the watermark intact. Therefore, the detection algorithm needs to be robust in order to detect the small change in the watermark. An example of the Flipping attack is shown in Figure 1-(c).
- *JPEG compression attack*: Consists of re-coding each frame of the video in JPEG format. As can be seen in Figure 1-(d), this attack is almost visually imperceptible.
- *Noise addition attack*: This is a global technique that inserts noise in video frames. Figure 1-(e) shows an example of this attack using salt-and-pepper noise. Any type of noise can be used in this type of attack.
- *Scaling*: Consists of resizing the video frame dimensions. In our simulations, this attack consisted of zooming, as shown in Figure 1-(f).

Temporal attacks consist of modifying the information about the frame position (time) [2]. They include attacks such as switching the position of two (or more) frames or changing the frame rate (removing or duplicating frames). Temporal tampering attacks are hard to detect and may be used to confuse the tampering detection algorithms. As a consequence, the robustness of the temporal tampering will determine the resilience of the watermarking tampering detection technique [7].



(a) Composition (b) Cropping (c) Flipping

(d) JPEG (e) Salt-and-pepper (f) Rescaling

Fig. 1. Spatial tampering using the fifth frame of the 'Diver' video.

The most popular temporal tampering attacks are the following:

- *Frame Duplication attack*: This is a temporal attack in which one or more frames are copied into other temporal locations of the videos. In our simulations of this attack, we copied a random number of frames and pasted them into a forward position.
- *Frame Removal attack* (or FrameRate Reduction): This is a temporal attack that reduces the number of frames per second (fps) of the video. This reduction is achieved by simply deleting some frames. In our simulations, all the videos have initially 30 fps. Our frame removal attack consists of randomly picking one of the frames in a 5 frames interval and deleting it.
- *Frame Switching attack*: Consists of choosing a random number of frame pairs and interchanging their position. This attack is only visually perceptible if the interchanged frames are far away from each other. In our simulations, the random number of pair frames were chosen in the interval between 10 and 20.

In summary, in this work we use ten types of tampering attacks: Composition, Cropping, Flipping, JPEG compression, Noise addition, Scaling, Frame Duplication, Frame Removal and Frame Switching. Although this list is not exhaustive, our framework can be extended to detect other types of attacks.

III. Tampering Protection Stage

The tampering protection stage generates the marks and embeds them into the video and audio channels. It aims to protect the digital content against tampering. The watermarking algorithm used in this approach enables a blind-detection. Fig. 2 shows a block diagram of the tampering protection stage. Notice from this figure that this stage can be roughly divided in three steps: watermarking generation, encryption, and embedding. In this section, we describe each of these steps.

A. Watermarking Generation

In order to protect both audio and video, we first need to decode the video and split it in two parts: the audio channels (A) and the video frames (F). Then, using a pseudo-random number generator with a secret key k, we generate a mark

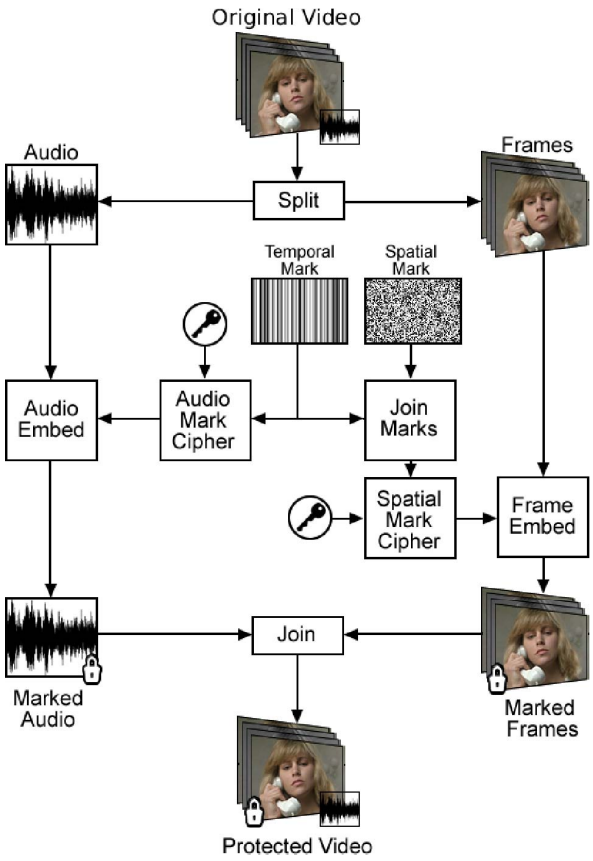


Fig. 2. Block diagram of the Tampering Protection stage.

to protect the spatial information (M_s) and another mark to protect the temporal information (M_t).

For each frame, the temporal mark M_t is a binary random sequence, which is used for temporal tampering detection. This mark is generated using the key k and it is replicated along the frame dimensions. The spatial mark M_s is a binary matrix with 5 bits of depth. M_s is used to detect spatial tampering in each of the video frames. Both marks are the same spatial size than the video frames. Notice that different marks are generated for each frame.

B. Watermarking Encryption

Both M_s and M_t are encrypted before the embedding process. The spatial mark M_s is encrypted using the equation:

$$\Psi_s[x, y] = M_s[x, y] \oplus k, \tag{1}$$

where $\oplus$ is the mathematical XOR operation, k is the encryption key, and Ψ_s is the M_s mark after encryption. Similarly, the temporal mark M_t is encrypted using the equation:

$$\Psi_t[x, y] = M_t[x, y] \oplus k, \tag{2}$$

where Ψ_t corresponds to the M_t mark after encryption.

C. Watermarking Insertion

For the video channel F, we generate a new mark by combining Ψ_t with Ψ_s in random positions. The combination process can be described by the following equation:

$$\Phi[x, y, z] = \begin{cases} \Psi_t[x, y], & \text{if } z = i \\ \Psi_s[x, y, z], & \text{otherwise} \end{cases} \tag{3}$$

where Φ is the combined mark, $i \in [0, 5]$ is a random number generated with a pseudo-random number generator using the key k. Φ is a matrix of 6 bits of depth that contains the two marks concatenated. Since the watermark Ψ_t is also embedded in the audio channel A, two copies of M_t are inserted for each frame, increasing the redundancy of M_t.

Each value of $\Phi[x, y, z]$ is embedded in the corresponding position (x, y) of the video frame. For each pixel, $\Phi[x, y, z]$ is split in three equal parts of 2 bits each. Then, they are converted to decimal, what results in 3 integer marks (Φ_1, Φ_2 and Φ_3). The three marks are inserted into the three color channels (RGB or YCbCr) of the frame.

Among the available watermarking methods, the Quantization Index Modulation (QIM) algorithm is one of the methods with the best performance [14]. The QIM algorithm inserts a mark into the host signal by quantizing it with a uniform scalar quantizer. The standard quantization operation is given by the following equation:

$$Q(x, \delta) = \text{round}\left(\frac{x}{\delta}\right),$$

where $round(.)$ denotes the mathematical operation of rounding a value to the nearest integer and δ is the step size used by the quantizer. The watermarked pixel is obtained using the following equation

$$s(x) = Q(x, \delta) + d(m), \tag{4}$$

where $d(m)$ is the *perturbation value*, which depends on the mark m to be embedded.

In this work, we use a proposed modification of the original QIM algorithm that increases the embedding capacity. Instead of using a function of the mark, the modulation function is set to $d(m) = m$. So, the integer marks Φ_1, Φ_2, and Φ_3 are inserted into the color channels using the following equation:

$$F_m[x, y, c] = Q(F_o[x, y, c], \delta) + \Phi_c[x, y], \tag{5}$$

where F_o is the original video frame, F_m is the resulting watermarked video frame, δ is the quantization step, c is the corresponding color channel, and x and y are spatial positions. For this work, we use $\delta = 4$, what allows embedding integer numbers with values between 0 and 3.

To embed the mark into the audio channel A, Ψ_t is reshaped into a unidimensional vector Ψ'_t. Then, Ψ'_t is embedded into A using the equation:

$$A_m[x] = Q(A[x], \delta) + \Psi'_t[x], \tag{6}$$

where A_m is the marked audio signal corresponding to a single video frame and A_o is the original audio channel. After A

and F are marked, the video is merged back and encoded, generating the marked video.

Notice that the embedding process provides a high temporal redundancy given that several copies of the temporal mark are inserted, both in the video and audio channels. Another important feature of the proposed technique is the ability to detect minimal changes in content. This is due to the combination of the two binary marks and the usage of the modified QIM algorithm.

IV. Tampering Detection Stage

The second stage of the proposed framework is the tampering detection stage, which is located at the receiver (decoder) side. The tampering detection should be performed after a transmission or storage of the video or if there is a suspicion of tampering. The Tampering Detection Stage is roughly divided in two parts: watermarking extraction and tampering detection, as shown in the block diagram shown in Fig. 3.

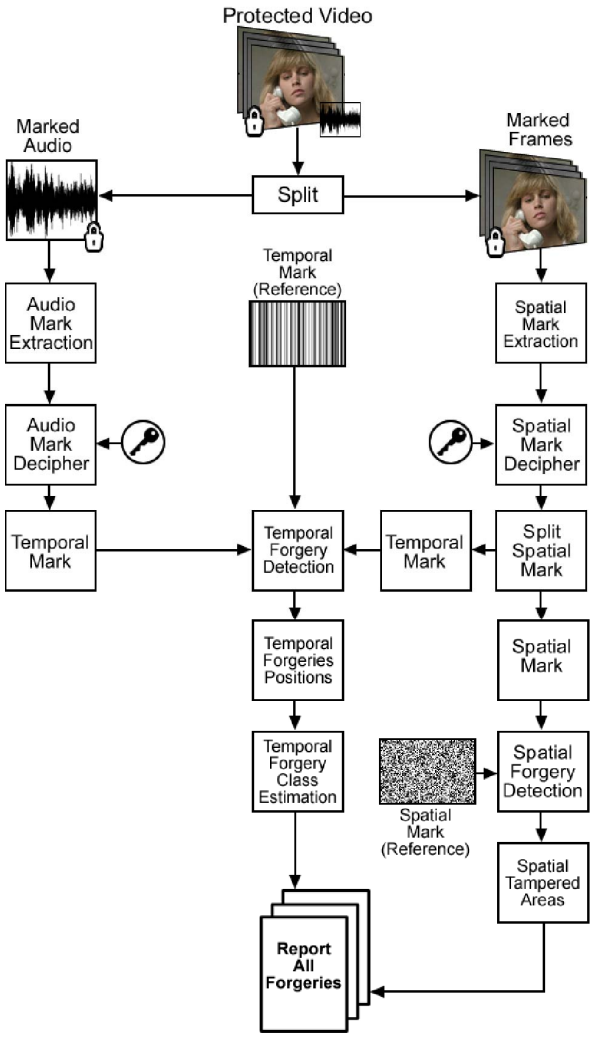


Fig. 3. Block diagram of the Tampering Detection stage.

A. Watermarking Extraction

The watermarking extraction is performed on the previously marked video. First, we split the video and audio channels and extract the marks from both channels.

From the audio channel, the mark is extracted using the following equation:

$$\hat{\Psi}'_{A,t}[x] = A_m[x] \mod \delta, \tag{7}$$

where $\hat{\Psi}'_{A,t}$ is the extracted temporal encrypted mark, $\mod$ denotes the mathematical operation modulo, and $A_m[x]$ is the marked audio. After the extraction, the unidimensional mark $\hat{\Psi}'_{A,t}$ is reshaped into the original dimensions and becomes $\hat{\Psi}_{A,t}$. Then, $\hat{\Psi}_{A,t}$ is decrypted using the following equation:

$$\hat{M}_{A,t}[x,y] = \hat{\Psi}_{A,t}[x,y] \oplus k, \tag{8}$$

where $\hat{M}_{A,t}$ is the temporal decrypted mark extracted from A.

The mark inserted in F (video frames) is a combination of the temporal and spatial marks. So, we must first extract the combination of these marks, decrypt it and, then, split them. For each frame, the extraction of the mark is performed using the following equation:

$$\sigma[x,y,c] = F_m[x,y,c] \mod \delta, \tag{9}$$

where $\sigma[x,y,c]$ is the extracted integer corresponding to the spatial position (x,y) at the color channel c. Notice also that $\sigma[x,y,c] \in [0,3]$.

Then, we transform the integer $\sigma[x,y,c]$ into binary representation, generating three variables γ_r, γ_g, and γ_b that correspond to each of the color channels, respectively. We join these binary numbers into a combined extracted mark $\hat{\Phi}[x,y]$ using the following equation:

$$\hat{\Phi}[x,y] = \gamma_r \mid\mid \gamma_g \mid\mid \gamma_b, \tag{10}$$

where $||$ denotes the concatenation operator. Notice that $\hat{\Phi}$ has depth equal to $3 \times \Delta$. For $\Delta = 2$, $\hat{\Phi}$ has six levels of depth. Each bit is independently addressed using the depth coordinate z.

Once $\hat{\Phi}[x,y,z]$ is created, the next step corresponds to the separation of the spatial and temporal marks. To accomplish this, we generate a random number $i \in [0,5]$ controlled by the same k used in the embedding process. Then, we extract the spatial and temporal marks using the following equations:

$$\begin{cases} \hat{\Psi}_{F,t}[x,y] = \Phi[x,y,l], & \text{if } l = i \\ \hat{\Psi}_{F,s}[x,y,z] = \Phi[x,y,l], & \text{otherwise.} \end{cases} \tag{11}$$

$\hat{\Psi}_{F,t}[x,y]$ receives the bit corresponding to the temporal mark. This bit is in i-th position, as explained in the embedding process. For all other cases, $\hat{\Psi}_{F,s}$ receives the other five bits of the spatial mark. Notice that $\hat{\Psi}_{F,s}$ has a depth level of 5 bits, while $\hat{\Psi}_{F,t}[x,y]$ a depth level of only 1 bit. At this point, both marks are encrypted and have the same size of the video frame.

$\hat{\Psi}_{F,s}$ is decrypted using the following equation:

$$\hat{M}_{F,s}[x,y] = \hat{\Psi}_{F,s}[x,y] \oplus k, \tag{12}$$

where $\hat{M}_{F,s}$ is the decrypted spatial mark (depth of 5 bits). Here, the same key k of the Tampering Protection stage is used. Likewise, the mark $\hat{\Psi}_{F,t}$ is decrypted using the equation:

$$\hat{M}_{F,t}[x,y] = \hat{\Psi}_{F,t}[x,y] \oplus k, \tag{13}$$

where $\hat{M}_{F,t}$ is the temporal decrypted mark.

After the temporal and spatial marks are extracted and decrypted, we generate the original marks M_s and M_t using the same key k, as described in Section III-A. The detection process is divided in Spatial Detection and Temporal Detection, which we describe next.

B. Spatial Detection

In spatial tampering detection, our goal is to detect local and global attacks. For each frame, local detection is performed by comparing the extracted mark ($\hat{M}_{F,s}$) with the original mark (M_s) using the following equation:

$$\rho_{F,s}[x,y] = \begin{cases} \text{true}, & \text{if } \hat{M}_{F,s}[x,y,z] \neq M_s[x,y] \\ \text{false}, & \text{otherwise} \end{cases} \tag{14}$$

where $\rho_{F,s}$ is a boolean *Spatial Detection Matrix* (SDM). In this matrix, each position that has 'true' values corresponds to a 'tampered' pixel. Notice that using this approach we are able to detect the exact location of a tampered area.

When the locations of the tampered areas are detected, we analyze the SDM to find the percentage of tampered area in relation to the total area of the video. The idea here is to try to characterize the changes to the video in order to classify the type of attack. For example, if this percentage is greater than a threshold, $\tau_{F,s}$, this frame is classified as "globally" attacked. Otherwise, it is classified as "locally" attacked. In our simulations, we used $\tau_{F,s} = 85\%$.

Notice that both local and global tampering damage the temporal mark inserted in the video frame. It is a problem. If temporal mark is lost, the temporal detection may fail. To avoid this problem, we use the temporal mark that is replicated in the audio channel. It allows the detector to differentiate temporal, local, or global attacks, as described next.

C. Temporal Detection

The temporal detection algorithm uses the temporal marks from F and A. First, it estimates the *Temporal Detection Matrix* (TDM) using the following equation:

$$\rho_{F,t}[x,y] = \begin{cases} \text{true}, & \text{if } \hat{M}_{F,t}[x,y] \neq M_t[x,y] \\ \text{false}, & \text{otherwise} \end{cases} \tag{15}$$

where M_t is the original temporal mark (reference) and $\rho_{F,t}$ is a boolean matrix. When $\rho_{F,t}$ contains 'true' values, a watermark loss occurred and the corresponding frame is probably temporally tampered. To detect if a frame is temporally tampered, we compare the percentage of 'true' values in TDM with a threshold $\tau_{F,t}$. For this work, we used $\tau_{F,t} = 85\%$. If this percentage is less than $\tau_{F,s}$ this frame is classified as NOT temporally attacked.

Notice that the 'true' percentage is not a very accurate criteria to determine temporal tampering because the spatial tampering may damage the temporal mark. To double check if the frame is temporally attacked, we can use the audio mark when $\rho_{F,t}$ is higher than $\tau_{F,t}$. The mark ($\hat{M}_{A,t}$) is compared with the original mark using the following equation:

$$\rho_{A,t}[x,y] = \begin{cases} \text{true}, & \text{if } \hat{M}_{A,t}[x,y] \neq M_t[x,y] \\ \text{false}, & \text{otherwise} \end{cases} \tag{16}$$

where $\rho_{A,t}$ is the TDM of the audio mark. From $\rho_{A,t}$, we compare the percentage of 'true' values with the threshold $\tau_{A,t}$. When the lost percentage in $\rho_{A,t}$ is smaller than $\tau_{A,t}$, the frame is NOT temporally tampered. For this work, we used $\tau_{A,t} = 15\%$. Therefore, the detection using $\rho_{F,t}$ indicates that a spatial tampering damaged the temporal mark embedded in the frame. If the audio and video comparison criteria indicates a temporal watermarking loss, the frame is classified as temporally attacked.

Next, we create a queue Θ to estimate the temporal attack type. This queue stores the temporal detection result. If the frame is classified as 'temporally attacked', Θ stores a copy of the extracted temporal mark. Otherwise, Θ stores $\varnothing$ (null) in the corresponding frame position.

1) Temporal Tampering Position: Once the temporal attack location is known, we can estimate the attack type. To do this, we first generate the original temporal mark and store it in a queue Ω. Then, we analyze Θ using Ω as a reference.

The first analysis consists of searching all occurrences in Ω that not happen in Θ. If mismatches are found, we compute the distances among all tampering attacks. If all occurrences have the same distance, this attack is classified as a FrameRate reduction. Otherwise, it is classified as Frame Deletion attack.

The second analysis consists in verifying if all elements of Ω are in Θ and if the indexing order is kept. Also, we verify if Θ contains elements that does not exist in Ω. If both conditions are satisfied, this means that Θ contains more elements than the original marks Ω. This indicates that new frames were inserted into the marked video. So, we classify this attack as a Frame Insertion attack.

The third analysis consists in searching all Ω elements that occur more than once in Θ. This means that one or more frames are copied and pasted into other video positions. So, this attack is classified as Frame Duplication attack.

Fourth, for each element marked as tampered at position i in Θ we find its correct position j in Ω. Then, we verify if the element $\Theta[j]$ is equal to $\Omega[i]$. If they are equal, then we classify this attack as a Frame Switch attack.

V. Simulations and Results

In this work, we use a set of fifteen copyright-free videos downloaded from the ReefVid database, which is maintained by the University of Queensland, Australia [15]. The videos used in our tests have different temporal and spatial resolutions. They also have different spatial and temporal characteristics. Some have a higher spatial activity (texture), while others have higher temporal activity (movement). This diversity is an important requirement when choosing a video database for testing tampering attacks.

The first test consists of analyzing the quality of the *marked* videos. Although our goal is to protect the video content from any tampering, it is important that the overall quality of the video is not decreased by the watermarking algorithm. This analysis was made by comparing the frames of marked and original videos using two popular full reference metrics: the Peak Signal-to-Noise Ratio (PSNR) and the Structural SIMilarity (SSIM) index [16]. The values obtained with these metrics are shown in Table I. The PSNR values obtained are all well above 45dB, while the SSIM values are all above 0.999. These values indicate that the marked videos have a very good quality with visually imperceptible distortions.

TABLE I
PSNR AND SSIM VALUES CALCULATED BETWEEN ORIGINAL AND MARKED TEST VIDEOS.

Video	Frames	PSNR	SSIM
Canoe	413	45.53459	0.99966
Diver	337	45.40895	0.99968
Coral	314	45.51629	0.99967
Fish	612	45.56291	0.99964
Seaweed	191	45.44542	0.99937
Beach	803	45.59478	0.99955
Rock	19	45.37220	0.99967
Sky	273	45.45621	0.99966
Birds	102	45.38711	0.99967
Deepsea	58	45.41717	0.99958
Aquamarine	1123	45.51627	0.99940
Rocks	751	45.57977	0.99973
Bill	900	45.54493	0.99936
Alga	576	45.55015	0.99966
Sunset	1938	45.64242	0.99970

Next, we tested ten types of attacks (three temporal and seven spatial), as explained in Section II. For the spatial attacks, we performed the same tests for all the fifteen videos. However, due to the large amount of data to be analyzed, Fig. 4 shows the results only for the video 'Sunset'. This figure shows the proportion of attacked (tampered) and detected regions for all frames. The attacks correspond to the areas modified using several types of tampering. The black bars are the proportions of areas detected as tampered, while the white bars are the proportions of the corresponding attacks. It is worth pointing out that less than 25% of the video is attacked and, therefore, the white (and black) bars do not reach the 100% limit. When white and black bars have the same size, the proposed framework is able to detect all tampering attacks.

Fig.5 shows four spatial attacks and their detections. In this figure, the red highlights illustrates the tampered regions. The first line shows Composition, Cropping, Flipping, and Salt-and-pepper (Noise) attacks. The second line shows their corresponding detections. Thus, Fig. 5-(a) shows the Composition attack and its detection for a frame of the video 'Cannoe'. The frames shown in Fig. 5-(b) illustrates the Cropping attack and its corresponding detection for the video 'Diver'. Fig. 5-(c) shows an example of a 180 degrees Flipping attack and its detection for the same video. For this attack, most of the frame was detected as being tampered since the majority of the spatial positions of the mark has been altered. Finally, Fig. 5-(d) shows a Salt-and-pepper noise attack and its detection

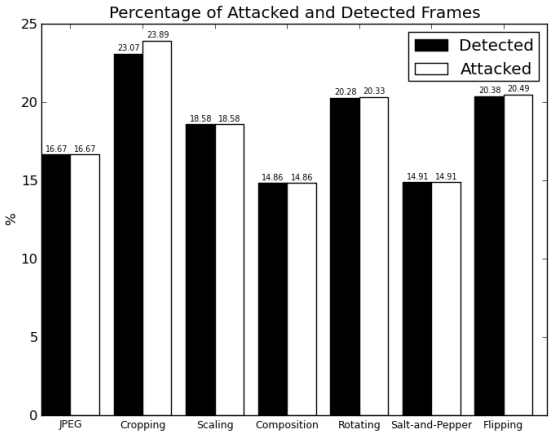


Fig. 4. Global spatial tampering detected for all frames.

for the video 'Rock'.

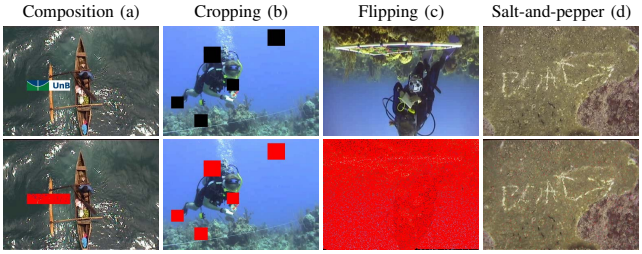


Fig. 5. Spatial attacks (first line) and their detected areas (second line) of five tested videos.

Table II shows the average percentage efficiency of seven spatial attacks for all tested videos, computed as the ratio between the amount of attacked areas and the amount of detected areas. From this table, we notice that the framework has 96% of successful detections for the Composition attack. The false-negative percentage average is 4%, since the inserted image by the Composition attack has always the same size. The Flipping, Rotating, and Scaling attacks have a false-negative percentage average of 12%, since these are global attacks in which the watermark is always lost in the same way. For the JPEG compression attack, we notice that the compression factor used in the attack is always 95%. Therefore, the efficiency percentage average is around 93% for all videos. The Salt-and-pepper (Noise) attack is applied randomly to the frames. So, the attack positions may overlap, causing the detection to fail in some overlapping positions. This attack has a false-negative percentage average of 4%. A similar situation of overlapping positions occurs for the Cropping attack, causing a false-negative percentage average of 12%.

We also performed three types of temporal attacks. To compare all attacks over the time, the framework requires a buffer. So, first, we load the video frames into the memory. The test results for two types of attacks are shown in Fig. 6

TABLE II
AVERAGE PERCENTAGE EFFICIENCY OF SPATIAL DETECTIONS PER AREA FOR ALL VIDEOS (%).

Video	Composition	Flipping	JPEG Compression	Rotating	Salt-and-pepper	Scaling	Cropping
Canoe	96.57532	88.89811	93.65289	88.86159	94.37949	88.84412	88.88089
Diver	96.53922	88.79136	93.52155	88.85033	94.41239	88.85801	88.65209
Coral	96.66685	88.95969	93.20273	88.86600	94.38747	88.85448	88.74452
Fish	96.62966	88.90802	92.99444	88.86235	94.22194	88.83048	88.81851
Seaweed	96.53025	88.86941	93.49474	88.86328	94.52187	88.83860	88.77541
Beach	96.65479	88.88090	93.54141	88.86420	94.40788	88.84218	88.79031
Rock	96.82041	88.92413	93.22163	88.88834	94.61698	88.83916	86.91857
Sky	96.66859	88.88664	92.80452	88.86852	94.46368	88.86577	88.98801
Birds	96.59843	88.90845	93.60656	88.86725	94.24075	88.85656	88.94274
Deepsea	96.65206	88.87395	93.66885	88.86175	94.40545	88.80427	88.55688
Aquamarine	96.62299	88.81509	93.52120	88.85723	94.29038	88.83239	88.83305
Rocks	96.62892	88.81875	93.48457	88.86136	94.34785	88.79764	88.68016
Bill	96.68402	88.81955	92.89044	88.85585	94.37704	88.86138	88.71293
Alga	96.54274	88.88281	93.49796	88.87459	94.37877	88.86033	88.87534
Sunset	96.65064	88.84509	92.69742	88.86846	94.42169	88.86054	88.83518
Total	96.63099	88.87213	93.32006	88.86474	94.39158	88.84306	88.66697

and Fig. 7. The Fig. 6 shows a Frame Duplication attack of the video 'Coral'. On the first line of the figure, the 28-th frame was copied and pasted between the 28-th frame and 29-th frame. The detection result is shown on the second line of the figure. Fig. 7 shows the result for a Switch attack of the video 'Sunset'. On the first line, the 163-th and 529-th frames of the video have their position interchanged. On the second line of the figure, the detection is presented. The last tested attack is the FrameRate reduction. Since the error report is similar to Fig. 6 and 7, we do not present this result in this paper.

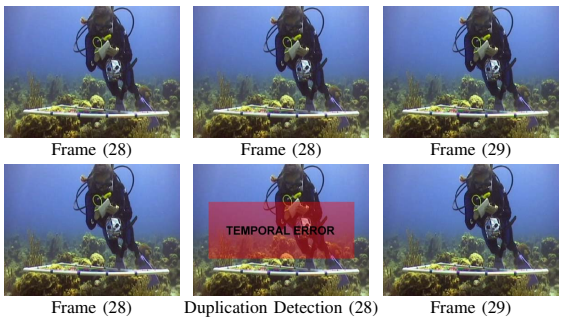


Fig. 6. Frame duplication attack and its detection for video 'Coral'.

Fig. 7. Temporal Frame Switch attack for video 'Sunset' and its detection.

The average efficiency and the accuracy of the temporal detection for all videos are shown in Table III. For the Frame Switch attack, an overlapping of attack positions also occurred. In other words, one frame was switched more than once and ended up back in its original position. For FrameRate and Duplication attacks, the framework is able to detect almost all tampered frames.

While Table III shows the average efficiency percentage for all videos, Fig.8 shows the exact number of detected and attacked frames of the 'Sunset' video. For the attacks Frame Duplication and FrameRate reduction, there were no false-positive, i.e. the detector was able to detect 100% of all test attacks. For the Frame Switch attack, there were only 7.8% of false-negatives for a video with 1938 frames.

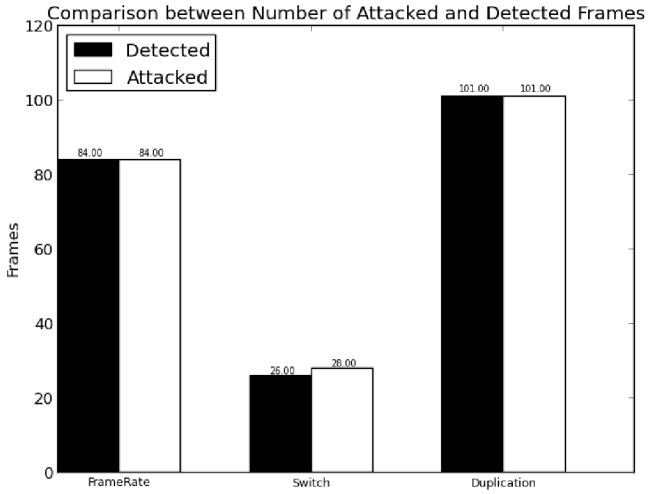


Fig. 8. Number of Temporal Detected and Attacked Frames of the 'Sunset' video.

In Table IV, we compare the spatial and temporal efficiency of the proposed technique with seven methods available in the literature [10], [17], [2], [18], [8], [6], [9]. In this table, the efficiency is computed as the ratio between the amount of tampered frames and the amount of detected tampered frames. The table shows the percentage of the cases in which an attacked frame was detected as being tampered, independent

TABLE III
AVERAGE PERCENTAGE EFFICIENCY OF TEMPORAL DETECTIONS FOR ALL VIDEOS (%).

Video	Switch	FrameRate	Duplication
Canoe	100.0	100.0	100.0
Diver	83.3	100.0	100.0
Coral	91.6	100.0	100.0
Fish	91.6	100.0	94.7
Seaweed	100.0	100.0	100.0
Beach	90.0	100.0	100.0
Rock	80.0	100.0	100.0
Sky	63.1	100.0	100.0
Birds	71.4	100.0	100.0
Deepsea	100.0	100.0	100.0
Aquamarine	100.0	100.0	100.0
Rocks	100.0	100.0	100.0
Bill	100.0	97.4	100.0
Alga	94.1	96.0	100.0
Sunset	92.8	100.0	100.0
Total	90.5	99.5	99.6

TABLE IV
MEAN EFFICIENCY OF SPATIAL DETECTIONS PER FRAME FOR SOME METHODS (%).

Method	Spatial Attack	Temporal Attack
Zhi-yu & Xiang [10]	50.49	-
Hsu *et al.* [17]	98.34	-
Lin *et al.* [2]	82.05	-
Pan & Lyu [18]	100	-
Subramanyam & Emmanuel [8]	85.01	99.5
Amerini *et al.** [6]	98.17	-
Wang & Farid* [9]	-	100
Proposed Framework	97.86	96.53

of the size of the attacked areas.

The proposed method is able to detect a larger number of tamper attacks than methods available in the literature. As shown in Table IV, our method achieves an average detection rate of 97.86% for Spatial Attacks and 96.53% for Temporal Attacks. While [8] and [9] present better results for temporal attacks, for spatial attacks [8] has a worse performance and [9] is not designed to detect them. On the other hand, [6] and [17] have better results for spatial attacks, but are not designed to detect temporal attacks. In summary, most of methods in the literature are designed to detect only a specific type of forgery, while the proposed method uses a single technique to detect several types of attacks and obtain a better or comparable performance.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a framework with the goal of detecting temporal, global, and local tampering in digital videos. The proposed framework is based on a simple and fast watermarking algorithm that does not damage the video quality. It allows identifying spatial and temporal tampering location with a pixel granularity. The combination of spatial and temporal marks increases the sensitivity and the robustness of the proposed technique, when compared to other algorithms found in the literature. Also, the replication of the temporal mark in the audio and video channels allows us to identify temporal attacks even if all frame content is lost. The framework is also able to estimate the type of tampering attack using an analysis of the tampering vector. Overall, the algorithm has good performance, presenting a high accuracy, efficiency, and a low percentage of false-positives.

Future works include the protection of the audio channel itself. Furthermore, a better criteria for picking the thresholds $\tau_{F,s}$, $\tau_{F,t}$, and $\tau_{A,t}$ is needed, given that these thresholds are used as criteria for classifying when a video is tampered or not. In this paper, we choose these values empirically, based on simulation results.

REFERENCES

[1] "Digital image forensics: a booklet for beginners," *Multimedia Tools and Applications*, vol. 51, 2011.

[2] E. Lin, A. Eskicioglu, R. Lagendijk, and E. Delp, "Advances in digital video content protection," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 171–183, 2005.

[3] F. Hartung, "Digital watermaking and fingerprinting of uncompressed and compressed video," Ph.D. dissertation, Universitat, Erlangen, Nurnberg, 2000.

[4] T.-T. Ng, S.-F. Chang, and Q. Sun, "Blind detection of photomontage using higher order statistics," in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 5, may 2004, pp. V–688 – V–691 Vol.5.

[5] F. Peng and X. lan Wang, "Digital image forgery forensics by using blur estimation and abnormal hue detection," in *Photonics and Optoelectronic (SOPO), 2010 Symposium on*, june 2010, pp. 1 –4.

[6] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy–move attack detection and transformation recovery," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 3, pp. 1099–1110, 2011.

[7] E. Lin *et al.*, "Video and image watermark synchronization," 2005.

[8] A. Subramanyam and S. Emmanuel, "Video forgery detection using hog features and compression properties," in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*. IEEE, 2012, pp. 89–94.

[9] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting duplication," in *Proceedings of the 9th workshop on Multimedia & security*. ACM, 2007, pp. 35–42.

[10] H. Zhi-yu and T. Xiang-hong, "Integrity authentication scheme of color video based on the fragile watermarking," in *Electronics, Communications and Control (ICECC), 2011 International Conference on*, sept. 2011, pp. 4354 –4358.

[11] I. Cox and J.-P. Linnartz, "Some general methods for tampering with watermarks," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 4, pp. 587 –593, may 1998.

[12] B. Mobasseri, M. Sieffert, and R. Simard, "Content authentication and tamper detection in digital video," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1, 2000, pp. 458 –461 vol.1.

[13] P. Yin and H. H. Yu, "Classification of video tampering methods and countermeasures using digital watermarking," pp. 239–246, 2001. [Online]. Available: +http://dx.doi.org/10.1117/12.448208

[14] B. Chen and G. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *Information Theory, IEEE Transactions on*, vol. 47, no. 4, pp. 1423–1443, 2001.

[15] P. Mumby. (2013, Apr.) Reefvid: A resource of free coral reef video clips for educational use. [Online]. Available: http://www.reefvid.org/

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[17] C.-C. Hsu, T.-Y. Hung, C.-W. Lin, and C.-T. Hsu, "Video forgery detection using correlation of noise residue," in *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*. IEEE, 2008, pp. 170–174.

[18] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *Information Forensics and Security, IEEE Transactions on*, vol. 5, no. 4, pp. 857–867, 2010.